

KOKUM

# NETWORK PROGRAMMING

---

## *User Manual & API Reference*

Remote databases. Live connections.  
HTTP clients. Services. Shared development.

**Kokum 0.29.0** |

First networking edition • 9 September 2026

A practical guide with small, source-aligned examples

**TigerDB Solutions Private Limited**

Kokum programming language created by Anil Jagtap

[kokum.dev](https://kokum.dev)

## READ THIS FIRST

# About this book

This standalone manual brings Kokum's network features together in one place. It is both a learning path and a working reference: start a local service, call it from Kokum, connect remote data, consume a WebSocket stream, and deploy the result through the correct runtime boundary.

| Edition baseline       | Value                                                 |
|------------------------|-------------------------------------------------------|
| Language and runtime   | Kokum 0.29.0; development line 0.29.0-phase4.17.7     |
| Source baseline        | Kokum_0_29_0_Phase4_17_7_Final_HTTP.zip               |
| Development host       | Linux: compiler, IDE and Remote FlatDB server         |
| Application execution  | KokumVM on Linux; Windows/macOS VM-only builds        |
| Manual validation host | Linux x86_64; source build and local network fixtures |

## What “working example” means

**Local lab** examples include their endpoint, configuration and run command. **GUI examples** require the named Designer components and connected events; a PUBLIC type declaration by itself does not construct a component. **External-service examples** require your own TigerDB or PostgreSQL server, credentials, schema and trust files.

The manual's console labs were executed locally. The two GUI networking examples were built and exercised through the real native form bridge. Database-provider examples were source-checked; provider regression fixtures were also run. A live customer database, a public Internet endpoint, a physical multi-user LAN and native Windows/macOS execution were not tested for this book. See the validation appendix.

**IMPORTANT** The authenticated network IDE is a trusted-LAN service using HTTP and plaintext users.csv passwords. It must not be exposed publicly. This restriction does not turn the separate backend-service host into an IDE server.

Source basis: [K01], [K02], [K03], [K14], [V1].

CONTENTS

Learning path and chapters

|                                                               |    |
|---------------------------------------------------------------|----|
| 01 Choose the right networking facility                       | 5  |
| 02 Prepare the Linux development host                         | 7  |
| 03 Use the local browser IDE                                  | 9  |
| 04 Start the passive database server                          | 13 |
| 05 Use the DATABASE component model                           | 18 |
| 06 Build a local echo application                             | 24 |
| 07 Understand the native URL contract                         | 30 |
| 08 Follow redirects with a bounded policy                     | 39 |
| 09 Launch independent HTTP tasks                              | 42 |
| 10 Host exported Kokum functions as an API                    | 47 |
| 11 Keep the UI transport separate from application networking | 52 |
| 12 Use the right security boundary for each facility          | 54 |
| APPENDICES API reference, diagnostics, validation and sources | 58 |

How to navigate

Chapter entries are clickable in Word. The document also uses real heading styles for the Navigation Pane. Page numbers refer to the printed page sequence, including the cover.

Readers new to networking should follow Chapters 1–7 in order. Readers already building an application can jump directly to the relevant API, then consult the configuration, security and troubleshooting chapters before deployment.

## CONVENTIONS

## Reading and running the examples

| Convention                    | Meaning                                                                                                           |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------|
| Kokum source                  | Save as UTF-8 with a .kokum extension. Code blocks contain ordinary editable text, not screenshots.               |
| Shell commands                | Run in a Linux terminal unless a block is explicitly labelled PowerShell.                                         |
| KOKUM_HOME                    | Absolute path to your extracted and built Kokum source directory.                                                 |
| network-lab                   | A new working directory you create for this book's examples.                                                      |
| Square brackets in signatures | Optional arguments, not characters to type into a call.                                                           |
| Names and case                | Keywords and HTTP option names are case-insensitive. Use the shown case for MAP keys and application identifiers. |
| Semicolon at line end         | Kokum continuation marker. Keep the source line break after it.                                                   |
| Example credentials           | Demo-only values. Never reuse a real account password in a lab.                                                   |

## One-time terminal setup

## LINUX SHELL

```
export KOKUM_HOME="$HOME/kokum-source"
export PATH="$KOKUM_HOME:$PATH"
mkdir -p "$HOME/network-lab"
cd "$HOME/network-lab"
```

Replace the first path with the actual Phase 4.17.7 source root. Keep this terminal in network-lab for the console, TLS and FlatDB client examples. The book states when a different directory is required.

## Core source syntax

## KOKUM BASICS

```
FUNCTION Main AS INTEGER
  LOCAL settings AS MAP
  settings = {"enabled": .T., "limit": 10}
  ? JSONENCODE(settings)
  RETURN 0
ENDFUNC
```

Kokum uses .T./F. logical values, NULL for no value, arrays such as [10, 20], maps such as {"name": "Kokum"}, and TRY/CATCH/FINALLY for cleanup. Avoid copying a typographic quote into code.

## 01 • NETWORK FEATURE MAP

# Choose the right networking facility

| Facility                  | Public entry point           | Use it for                                           |
|---------------------------|------------------------------|------------------------------------------------------|
| HTTP/HTTPS client         | URLGET / URLPOST             | Request/response APIs, JSON, text and form bodies    |
| WebSocket client          | WebSocketClient / WEBSOCKET  | Long-lived text or binary streams                    |
| Remote FlatDB             | KCONNECT / KQUERY / KEXECUTE | Shared single-file relational data on a Linux server |
| TigerDB provider          | DATABASE component           | TigerDB server or Router access                      |
| PostgreSQL provider       | DATABASE component           | PostgreSQL access through libpq                      |
| Backend services          | kokumvm serve                | Exported Kokum handlers behind bounded HTTP routes   |
| Local IDE and Web GUI     | Tokenized loopback hosts     | Browser UI on the same computer as the runtime       |
| Authenticated network IDE | kokum-ide server             | Multiple trusted-LAN users on one Linux installation |

## Do not confuse the similarly named pieces

Remote FlatDB is not the TigerDB database server. A WebSocket Client component is not a public WebSocket server. The Web GUI bridge is not the outbound application WebSocket connection. A browser connecting to the LAN IDE is not running the compiler or opening database sockets itself.

The HTTP function names are **URLGET** and **URLPOST**. HTTPGET, HTTPPOST, URLGETASYNC and URLPOSTASYNC are not native aliases in this release. Async behavior is composed with Kokum's existing task language.

**SCOPE** No public raw TCP/UDP, SMTP/IMAP, FTP/SFTP, HTTP/2/3, SSE, generic WebSocket server or automatic proxy-discovery API is documented in this source line. A CHANNEL is an in-process synchronization object, not a network socket.

Source basis: [K02], [K04]–[K13].

## 01 • NETWORK FEATURE MAP

## Where the connection actually runs

| Layer                 | What it owns                                                  | What it must not own                                               |
|-----------------------|---------------------------------------------------------------|--------------------------------------------------------------------|
| Browser frontend      | Visible controls, user input, selected result data            | Database passwords, private keys or native connection handles      |
| KokumVM owner session | DATABASE/WEBSOCKET values, GUI state, task results            | Another session's mutable native objects                           |
| Task worker           | Snapshot arguments, request-local HTTP resources              | Live GUI widgets or a transferred WEBSOCKET handle                 |
| Backend worker        | Persistent VM session and its request callbacks               | Unbounded work or an arbitrary client-selected upstream            |
| Linux LAN IDE host    | Compiler, per-user workspaces, launched application processes | A promise of operating-system isolation between untrusted programs |

### The loopback rule

127.0.0.1 always means the machine that makes the connection. When a developer clicks Run in the network IDE, the application executes on the Linux server. `URLGET("http://127.0.0.1:18080/hello")` therefore calls a service on that server—not a service on the developer's laptop.

### A practical network inventory

| Endpoint              | Book convention                                       |
|-----------------------|-------------------------------------------------------|
| Local IDE / Web GUI   | Automatic loopback port and private launch token      |
| LAN IDE               | Private IPv4 address, TCP 9440 in the example         |
| Remote FlatDB         | TCP 19437 in the lab; 9437 is the normal example port |
| Kokum HTTP lab        | 127.0.0.1:18080                                       |
| TLS test helper       | 127.0.0.1:18443                                       |
| WebSocket echo helper | 127.0.0.1:18765                                       |

Ports in this book are local lab choices, not mandatory constants. Change both ends together when a port is already in use. Never substitute a public address into the LAN IDE configuration.

Source basis: [K03], [K08], [K11], [K12], [K13].

## 02 • WORKSTATION AND PROJECTS

# Prepare the Linux development host

## Start from the matching release

Use the complete Phase 4.17.7 tree. The source directory name and DEVELOPMENT\_LINE identify the baseline more reliably than historical paragraphs retained in older README and help files.

### BUILD AND INSPECT

```
cd "$KOKUM_HOME"
cat DEVELOPMENT_LINE
make -j4
make flatdb
./kokumc --version
./kokumvm --version
./kokum-ide --help
```

The source build needs the existing C toolchain and development libraries for OpenSSL, SQLite, PostgreSQL/libpq and PCRE2, plus their platform dependencies. Use the release build instructions for your Linux distribution. Node.js and libcurl are not native HTTP-client dependencies.

## Optional tools used only by this book's labs

Python 3 is used for local TLS and WebSocket fixtures. The WebSocket fixture uses the pinned Python websockets 16.0 package. OpenSSL's command-line utility creates a disposable local certificate. curl is a convenient independent test client, not a Kokum runtime dependency.

## Validate without installing system services

### AFTER SAVING THE CHAPTER 7 GET EXAMPLE

```
cd "$HOME/network-lab"
kokumc check get.kokum
kokumc run-bytecode get.kokum
```

A successful semantic check proves that the program is accepted by the compiler. It does not prove that the endpoint exists, that a certificate is trusted, or that a database account has permission. The run step is meaningful only after its lab service is started.

**CLEAN LAB** Use fresh example directories and disposable records. The FlatDB CRUD example changes only its manual\_people table and removes its own row. Do not run demonstration DDL against a production database.

Source basis: [K01], [K14], [R6], [V1].

## 02 • WORKSTATION AND PROJECTS

## Compile once, deploy deliberately

### SINGLE-SOURCE PROGRAM

```
kokumc check get.kokum
kokumc compile get.kokum -o get.kbc
kokumvm get.kbc
kokumvm verify get.kbc
```

For a multi-module application, use a manifest. Put the source in `src/main.kokum` and make its `MODULE` name match `entry_module`. The example below is a console project; a Designer project has additional web, form and resource declarations.

#### kokum.toml

```
[project]
name="network.demo"
version="1.0.0"
kind="console"
entry_module="network.demo"
entry="Main"

[bundle]
output="dist/network-demo.kapp"

[[module]]
name="network.demo"
version="1.0.0"
source="src/main.kokum"
```

#### src/main.kokum

```
MODULE network.demo VERSION "1.0.0"
EXPORT FUNCTION Main
FUNCTION Main AS INTEGER
    ? "Network project ready"
    RETURN 0
ENDFUNC
```

### BUILD A BUNDLE OR LINUX STANDALONE

```
kokumc bundle kokum.toml
kokumvm dist/network-demo.kapp
kokumc build --standalone kokum.toml -o dist/network-demo
```

A `.kapp` carries compiled code and declared resources. External configuration and secrets travel separately. A standalone executable embeds its VM: rebuild it when updating the runtime. A portable `.kbc/.kapp` can instead run under a newer compatible VM.

Source basis: [K01], [K07], [K14].

## 03 • LOCAL AND NETWORK IDE

# Use the local browser IDE

**LOCAL MODE: ALTERNATIVE LAUNCHES**

```
kokum-ide "$HOME/network-lab"  
kokum-ide "$HOME/network-lab" --no-browser  
kokum-ide --no-browser --port 9441
```

Local mode listens only on 127.0.0.1 and issues a fresh tokenized URL. Open the complete URL printed by the process. A bare port URL or a token copied from an earlier launch is not equivalent.

**Development workflow**

Create or open a Console or Web GUI project. Save the source, run Check, then Build or Run. In a Web GUI project, add Database and WebSocket Client objects from the Application Components tray; they are nonvisual components rather than visible controls.

The native runtime performs database and outgoing network I/O. The browser receives form state and selected result data through a separate authenticated bridge. Do not embed access tokens in web/app.js or other browser resources.

**HTTP assistance in the editor**

URLGET and URLPOST have signatures, hover help, snippets and option completion. The HTTP reference panel covers the twenty request options and fourteen response fields. Local advisory diagnostics cover unknown or duplicate options, literal type/range errors, unsafe TLS/auth settings and incompatible authentication choices.

Ctrl/Cmd+Space remains available for explicit completion. Suggestions distinguish the GET options argument from the POST body and nested header maps. Dynamic expressions are still validated at runtime; the editor does not call your endpoint while analysing source.

**SECURITY** The local IDE's token, Host/Origin validation and self-hosted assets are part of its boundary. Do not remove those checks to make a remote URL work; use the separate authenticated LAN mode.

Source basis: [K03], [K09], [K11], [K13].

## 03 • LOCAL AND NETWORK IDE

## Configure authenticated LAN access

Select a stable private IPv4 address assigned to the Linux host. In this example it is 192.168.10.20, and approved clients are in 192.168.10.0/24. Replace both values with your actual LAN settings. Wildcard and public listener addresses are rejected.

**CREATE A PRIVATE BASE DIRECTORY**

```
mkdir -p "$HOME/kokum-lan"
chmod 700 "$HOME/kokum-lan"
```

**ide-lan.conf — REPLACE base\_directory WITH YOUR REAL ABSOLUTE PATH**

```
[server]
format_version=2
mode=lan
listen_address=192.168.10.20
listen_port=9440
advertised_host=192.168.10.20
allowed_client_cidr=192.168.10.0/24
allow_loopback=true
base_directory=/home/yourname/kokum-lan
users_file=users.csv
workspace_directory=workspaces
tvm_path=auto
enable_project_check=true
enable_project_build=true
enable_project_run=true
enable_project_deploy=true
trace=false
```

Use the complete config/kokum-ide-lan.conf.example when tuning session, quota, client and request limits. Keep configuration files private. The base directory must already exist and must not be a symlink.

**CHECK, THEN START**

```
kokum-ide server --config ide-lan.conf --check-config
kokum-ide server --config ide-lan.conf
```

From another approved LAN computer, open <http://192.168.10.20:9440/> and sign in. This is **server**, not serve: kokum-ide server hosts the IDE; kokumvm serve hosts an application API.

**LAN ONLY** Do not port-forward this service, bind it to a public interface, or publish it through a public reverse proxy. HTTP and plaintext account storage remain deliberate limitations of this IDE mode.

Source basis: [K03].

## 03 • LOCAL AND NETWORK IDE

## Manage users, sessions and workspaces

**users.csv — SAVE UNDER base\_directory**

```
username,password
admin,replace-this-lab-admin-password
learner1,replace-this-lab-user-password
```

**FILE PERMISSIONS**

```
chmod 600 "$HOME/kokum-lan/users.csv"
```

The header must be exactly username,password. admin is the reserved administrator. Other accounts are developer accounts. Usernames are matched case-insensitively; duplicate names are rejected. Passwords retain their exact contents. Ordinary CSV quoting is required for commas or quotation marks in a field.

The file must be a regular private file, not a symlink, with no group/other permissions. Anyone who can read this file can read every password. Use lab-only passwords and do not reuse important credentials.

### Administrator operations

The reserved administrator can open /admin/users to create a developer, change a password or delete a developer. There is no self-registration route. The admin account cannot be deleted. Password changes and deletions invalidate the affected user's active sessions.

Authenticated sessions use a random in-memory session and an HttpOnly, SameSite=Strict cookie. Idle/absolute expiry, per-user session limits, client-address binding and login throttling are configurable. These controls do not encrypt HTTP traffic.

### Workspace layout

**SERVER-SIDE TREE**

```
kokum-lan/
  users.csv
  ide-server.log
  workspaces/
    learner1/
      projects/
      state/
      cache/
      temp/
      trash/
```

Each user receives a contained IDE project tree. This is a workspace-management boundary, not a full operating-system sandbox for arbitrary hostile code. Use the service only with trusted users, and consider disabling Run for a group that should not execute code on the host.

Source basis: [K03].

## 03 • LOCAL AND NETWORK IDE

## Check, Build, Run and Deploy over the LAN

| Operation        | Current behavior                                                                                     |
|------------------|------------------------------------------------------------------------------------------------------|
| Check            | Checks the authenticated user's project within its contained root.                                   |
| Build            | Builds server-side project artifacts.                                                                |
| Run              | Starts the selected .kapp in a separate KokumVM process on the Linux host.                           |
| Deploy           | Creates a Linux standalone executable at the server-side output location.                            |
| Browser download | Deploy does not itself transmit the executable to the client browser.                                |
| Web GUI preview  | A launched local Web GUI prints its loopback URL; LAN preview routing is not supplied by this phase. |

The enable\_project\_check/build/run/deploy options default to true. Set a capability to false to hide it in the IDE and reject its corresponding backend operation. The current implementation supersedes older help paragraphs that say all remote operations are disabled.

### Understand paths before debugging

A remote Run has the project root as its working directory. A relative OPEN("upstream.json", "r") therefore reads a file on the server. The browser's Downloads folder and the client's local drive are not automatically part of that project.

KOKUM\_WEB\_NO\_BROWSER=1 prevents a launched Web GUI from opening a browser on the Linux server. KOKUM\_WEB\_PRINT\_URL=1 keeps its launch URL in the VM output. A server loopback URL is not directly usable on another computer.

### Classroom deployment pattern

Keep one Linux toolchain installation. Give each participant an account and isolated project directory. Supply approved sample projects and endpoints on the same LAN. Collect artifacts through your separately approved file-distribution process; do not assume the IDE's Deploy action performs a download.

### Failure checks

Check the operation capability, actual server project path, configured tvn\_path, source permissions, disk quota and output pane. For GUI network calls, distinguish application errors from the missing remote preview route.

Source basis: [K03], especially docs/LAN\_PROJECT\_OPERATIONS.md and the current LAN configuration.

## 04 • REMOTE FLATDB

# Start the passive database server

Remote FlatDB is Kokum's Linux-hosted single-file relational service. KokumVM contains its client; there is no separate kokum-flatdb-client product binary. A matching shared key authenticates a connection, after which requests and replies use authenticated encryption.

**AT REST** The .kfdb database file is not encrypted at rest. Network encryption does not protect a copied database file. Protect the host disk, permissions and backups separately.

**TERMINAL 1 — INITIAL CREATION**

```
mkdir -p "$HOME/flatdb-lab"
chmod 700 "$HOME/flatdb-lab"
cd "$HOME/flatdb-lab"
openssl rand -out manual.kkey 64
chmod 600 manual.kkey
"$KOKUM_HOME/kokum-flatdb-server" \
  --database "$HOME/flatdb-lab/manual.kfdb" \
  --key manual.kkey --listen 127.0.0.1 \
  --port 19437 --create
```

The key must be beside the database; a key from another directory is rejected. The key file's basename and binary contents must match the client copy exactly. Keep this server running while you execute the client examples.

**LATER STARTS — OMIT --create**

```
"$KOKUM_HOME/kokum-flatdb-server" \
  --database "$HOME/flatdb-lab/manual.kfdb" \
  --key manual.kkey --listen 127.0.0.1 --port 19437
```

For a real LAN database, bind the intended server interface and restrict access with the host/network firewall. Unlike the IDE, the FlatDB server can accept a wildcard listener; that does not make unrestricted exposure a safe default. This book's loopback lab does not require root.

Source basis: [K04].

## 04 • REMOTE FLATDB

## Place the configuration and shared key correctly

**TERMINAL 2 — COPY THE SAME FILE, DO NOT GENERATE ANOTHER KEY**

```
cp "$HOME/flatdb-lab/manual.kkey" "$HOME/network-lab/"
chmod 600 "$HOME/network-lab/manual.kkey"
cd "$HOME/network-lab"
```

**kdb.conf**

```
[connection]
host=127.0.0.1
port=19437
database=manual.kfdb
key_file=manual.kkey
connect_timeout_ms=3000
request_timeout_ms=5000
max_message_bytes=16777216
```

database is the server's exact filename, not its full filesystem path. INI values are unquoted. key\_file is relative to the selected configuration directory when it is not absolute. A file named manual.kkey.txt is not the same key filename.

### Connection syntax

**INSIDE A KOKUM ROUTINE**

```
KCONNECT USING kdb.conf
IF .NOT. KCONNECTED()
    ? "Remote database is unavailable."
ENDIF
KDISCONNECT()
```

The equivalent function form is KCONNECT("kdb.conf"). A failed remote authentication does not provide a detailed reason. Local file/format errors can still be reported locally. Catch runtime errors as well as checking KCONNECTED().

### Application-relative lookup

A relative configuration path is tried beside the active source, bytecode or bundle before the historical working-directory fallback. KokumVM's executable directory is a further fallback where module origin is unavailable. Prefer one clearly placed application copy instead of relying on several competing kdb.conf files.

**DEPLOYMENT TREE**

```
application/
  application.kapp
  kdb.conf
  manual.kkey
```

**KEY CHECK** Compare sha256sum manual.kkey on Linux, or Get-FileHash on Windows, without printing the key. Equal hashes are useful only when the basenames also match and you are comparing the intended files.

Source basis: [K04].

## 04 • REMOTE FLATDB

## A complete, repeatable CRUD program

## flatdb.kokum — LOCAL LAB

```

FUNCTION Main AS INTEGER
  LOCAL rows AS ARRAY
  TRY
    KCONNECT USING kdb.conf
    IF .NOT. KCONNECTED()
      ? "Remote database is unavailable."
      RETURN 1
    ENDIF
    KEXECUTE("CREATE TABLE IF NOT EXISTS manual_people (" + ;
      "id INTEGER PRIMARY KEY, name TEXT, city TEXT)")
    KEXECUTE("CREATE INDEX IF NOT EXISTS manual_city " + ;
      "ON manual_people(city)")
    KEXECUTE("DELETE FROM manual_people WHERE id=101")
    KEXECUTE("INSERT INTO manual_people VALUES " + ;
      "(101, 'Kokum reader', 'Pune')")
    rows = KQUERY("SELECT id,name,city FROM manual_people " + ;
      "WHERE id=101")
    ? JSONENCODE(rows)
    KEXECUTE("UPDATE manual_people SET city='Mumbai' " + ;
      "WHERE id=101")
    KEXECUTE("DELETE FROM manual_people WHERE id=101")
  CATCH error
    ? "Database operation failed: " + STR(error)
    RETURN 1
  FINALLY
    KDISCONNECT()
  ENDTRY
  RETURN 0
ENDFUNC

```

## RUN FROM network-lab WHILE THE SERVER IS ACTIVE

```
kokumc run-bytecode flatdb.kokum
```

Expected query output: [{"id":101,"name":"Kokum reader","city":"Pune"}]. The example creates its table/index only when needed, replaces only its own ID 101, updates that record, then deletes it. FINALLY closes the built-in connection even on a failure.

**DISPOSABLE DATA** The preparatory DELETE makes this lab repeatable; it is not a general upsert pattern and is not a transaction. Use a dedicated demonstration database.

Source basis: [K02], [K04], [V1].

## 04 • REMOTE FLATDB

## SQL, results and safe input boundaries

| Operation     | Contract                                                                                            |
|---------------|-----------------------------------------------------------------------------------------------------|
| KEXECUTE(sql) | Exactly one STRING argument; returns INTEGER affected-row count.                                    |
| KQUERY(sql)   | Exactly one STRING argument; returns ARRAY of row MAPs.                                             |
| KCONNECTED()  | Reports the owner context's built-in connection state.                                              |
| KDISCONNECT() | Closes that connection and returns LOGICAL.                                                         |
| SQL scope     | CREATE TABLE, CREATE INDEX/UNIQUE INDEX, INSERT, SELECT, UPDATE, DELETE; one statement per request. |

KEXECUTE(sql, [values]) and KQUERY(sql, [values]) are **not valid overloads** in this release. Positional parameter arrays belong to DATABASE.Execute/Query and the other provider methods in Chapter 5. Do not copy parameter-array examples from those APIs into the K-prefixed FlatDB calls.

### Use fixed SQL for the first application

#### SMALL QUERY FRAGMENT

```
rows = KQUERY( ;
  "SELECT id,name FROM manual_people " + ;
  "WHERE city='Pune' ORDER BY id LIMIT 20")
```

For dynamic input, validate values against a narrow business rule before constructing SQL. Do not concatenate unrestricted browser text, identifiers or expressions. For applications requiring general untrusted text binding, prefer the parameterized DATABASE provider API or design an application-specific validation layer.

### Transactions and result size

Remote FlatDB does not expose BEGIN/COMMIT/ROLLBACK through this SQL allowlist. A sequence of KEXECUTE calls is not an atomic multi-statement transaction. The server serializes conflicting writes, but that is not an application transaction API.

Use LIMIT and select only the columns you need. KQUERY materializes its rows rather than exposing a streaming cursor. Use unique output names in a SELECT so MAP keys are unambiguous. Keep request\_timeout\_ms and max\_message\_bytes deliberate; a larger limit is not a fix for an unbounded query.

Source basis: [K02], [K04].

## 04 • REMOTE FLATDB

## Own each connection and close it reliably

**flatdb\_async.kokum — RUN AFTER THE CRUD LAB**

```

ASYNC FUNCTION ReadPeople AS ARRAY
  LOCAL rows AS ARRAY
  TRY
    KCONNECT USING kdb.conf
    IF .NOT. KCONNECTED()
      RETURN []
    ENDIF
    rows = KQUERY("SELECT id,name FROM manual_people LIMIT 10")
    RETURN rows
  FINALLY
    KDISCONNECT()
  ENDTRY
ENDFUNC

FUNCTION Main AS INTEGER
  LOCAL task AS TASK
  task = ReadPeople()
  ? JSONENCODE(AWAIT task)
  RETURN 0
ENDFUNC

```

Run with `kokumc run-bytecode flatdb_async.kokum`. The earlier CRUD program leaves an empty table, so the expected result is `[]`. The task opens and closes its own built-in session; it does not borrow an owner thread's open connection. This example uses `[]` as its application-specific unavailable result, so a production API should return an explicit status envelope when “no rows” and “unavailable” must differ.

### Multiple clients and revocation

Multiple clients may use the same key. They receive separate encrypted sessions, but the initial authorization model gives all valid key holders equivalent database access; there are no per-user roles or read-only keys. To revoke copies, stop the server, replace the shared key, distribute the new exact file only to approved clients and restart. Terminate old sessions rather than assuming an established session is instantly revoked.

### Operational habits

Keep the server key private and outside the browser resource tree. For a simple consistent backup, stop the server cleanly and copy the database together with any related deployment configuration under controlled access. Do not copy a live main file casually while writes are in flight.

Source basis: [K04], [K11], [V1].

## 05 • REMOTE DATABASE PROVIDERS

# Use the DATABASE component model

TigerDB and PostgreSQL are available through the provider-neutral DATABASE type. Add the relevant nonvisual component in the Designer and assign an ID such as dbTiger or dbPostgres. The project creates and binds the object to its owner VM session; a bare PUBLIC dbPostgres AS DATABASE declaration does not connect to a server.

| Method                                         | Returned shape                                    |
|------------------------------------------------|---------------------------------------------------|
| Open() / Close()                               | Manage the configured connection.                 |
| Execute(sql [, parameters])                    | Run DDL or a write; provider execution result.    |
| Query(sql [, parameters])                      | ARRAY of row MAPs.                                |
| QueryOne(sql [, parameters])                   | One row MAP, or NULL.                             |
| QueryValue(sql [, parameters])                 | One scalar, or NULL.                              |
| QueryTable(sql [, parameters] [, headings])    | Two-dimensional array for TableView.              |
| Begin() / Commit() / Rollback()                | Transaction control on the same connection.       |
| CreateTableFromJson(name, data [, schemaOnly]) | Infer a new table and optionally insert the data. |

## External settings

Use db\_env.conf for environment-specific settings. The runtime checks KOKUM\_DB\_ENV, then the legacy TLANG\_DB\_ENV, then db\_env.conf beside the .kapp/standalone application. \${APP\_DIR}, \${PROJECT\_DIR}, \${USER\_DATA\_DIR} and \${ENV:NAME} are supported substitutions; they do not invoke a shell.

### OVERRIDE THE EXTERNAL CONFIGURATION

```
KOKUM_DB_ENV=/path/to/private/db_env.conf \
kokumvm dist/application.kapp
```

The component section's module must name the slots module that owns the generated PUBLIC variable. Keep connection settings and secrets out of browser resources. Retain the managed declaration blocks instead of adding duplicate PUBLIC declarations by hand.

Source basis: [K05], [K06], [K07].

## 05 • REMOTE DATABASE PROVIDERS

## Connect to TigerDB or its Router

**db\_env.conf — ENVIRONMENT-DEPENDENT EXAMPLE**

```
[database:dbTiger]
provider=tigerdb
module=manual.tiger
endpoint=server
host=tigerdb.internal.example
port=9192
database=training
username=kokum_app
password=${ENV:KOKUM_DBTIGER_PASSWORD}
tls=true
verify_tls=true
ca_file=${APP_DIR}/tls/company-ca.pem
tls_server_name=tigerdb.internal.example
auto_open=false
connect_timeout_ms=10000
query_timeout_ms=30000
max_message_bytes=8388608
```

Supply your real endpoint, account, database and CA. The .example name is a placeholder, not an operating service. Typical ports are 9191/9192 for plain/TLS server deployments and 9292/9293 for Routers, but the actual server configuration is authoritative.

### Router access

**CHANGE ONLY THE RELEVANT CONFIGURATION FIELDS**

```
endpoint=router
host=router.internal.example
port=9293
```

The ordinary DATABASE API does not change when you select a Router. Mutual TLS is available through `client_certificate` and `client_key`. Certificate identity can be set with `tls_server_name` when deliberately connecting by IP to a certificate naming a DNS service.

**PROJECT BOUNDARY** Kokum contains a TigerDB client provider, not TigerDB server, SQL-engine, storage, WAL, recovery or administration implementation. The provider can only execute what the connected TigerDB deployment supports.

Source basis: [K05], [K07].

## 05 • REMOTE DATABASE PROVIDERS

## Read and write through TigerDB

Prerequisites: a Web GUI project with dbTiger, a Label named lblStatus, a Button named btnQuery and a connected clicked slot. The configured training database must contain customers(id, name), with any desired test rows. Keep the generated component declaration; insert this handler in the module configured in db\_env.conf.

**TIGERDB BUTTON HANDLER — SOURCE-CHECKED**

```
PROCEDURE btnQuery_clicked(sender, event, form)
  LOCAL rows AS ARRAY
  TRY
    dbTiger.Open()
    dbTiger.Ping()
    rows = dbTiger.Query( ;
      "SELECT id,name FROM customers WHERE id = ?", [101])
    form.lblStatus.Text = JSONENCODE(rows)
  CATCH error
    form.lblStatus.Text = "Database request failed"
  FINALLY
    dbTiger.Close()
  ENDTRY
ENDPROC
```

### Parameter arrays and transactions

**FRAGMENT — dbTiger MUST ALREADY BE OPEN**

```
dbTiger.Begin()
TRY
  dbTiger.Execute( ;
    "UPDATE customers SET name = ? WHERE id = ?", ;
    ["Kokum reader", 101])
  dbTiger.Commit()
CATCH error
  dbTiger.Rollback()
ENDTRY
```

TigerDB's provider converts parameter values into escaped SQL literals while respecting quoted text and comments. This is not a reusable server-side prepared-statement object. PostgreSQL uses a different parameter transport under the same method signature.

QueryOne returns NULL when no row is found; check before indexing. QueryTable(sql, NULL, .T.) includes a heading row for a TableView. UseDatabase("sales") selects another logical database. Ping() and Raw({"action":"ping"}) are TigerDB-specific diagnostics; normal applications should prefer the shared API.

**NO BLIND RETRY** A failed reply does not prove the server did not apply a write. Reconnect and reconcile application state before deciding whether to retry a non-idempotent operation.

Source basis: [K05], [V1]. Live application-server execution requires your TigerDB environment.

## 05 • REMOTE DATABASE PROVIDERS

## Configure PostgreSQL with explicit TLS policy

**db\_env.conf — ENVIRONMENT-DEPENDENT EXAMPLE**

```
[database:dbPostgres]
provider=postgresql
module=manual.providers
host=postgres.internal.example
port=5432
database=training
username=kokum_app
password=${ENV:KOKUM_DBPOSTGRES_PASSWORD}
schema=public
sslmode=verify-full
sslrootcert=${APP_DIR}/tls/company-ca.pem
sslcert=
sslkey=
application_name=KokumNetworkManual
auto_open=false
connect_timeout_ms=10000
query_timeout_ms=30000
read_only=false
```

PostgreSQL support uses libpq. The canonical provider name is postgresql; postgres and pgsq are accepted configuration aliases. Supply the real host, database, account and trust file. The client machine must also have the required libpq runtime dependencies.

### TLS mode is not the same as TLS verification

Some historical examples use sslmode=prefer. That mode can fall back to an unencrypted connection and does not authenticate the server's identity. For a security-sensitive deployment, verify-full with the correct root CA checks both trust and hostname; correct the CA or certificate name instead of disabling verification. [R5]

sslcert and sslkey optionally supply a client certificate and private key. These are separate from sslrootcert, which establishes trust in the server. Do not copy a private key to a browser asset directory.

### Server prerequisites

The database must already exist, the role must be permitted by the server's authentication configuration, and its privileges must match the example. Kokum does not automatically create PostgreSQL roles or bypass server authentication. Use a dedicated least-privilege application role.

Source basis: [K06], [K07], [R5].

## 05 • REMOTE DATABASE PROVIDERS

## Run a small PostgreSQL query

Add PostgreSQL as `dbPostgres`, a Label `lblStatus` and a Button `btnQuery`. Connect clicked to `btnQuery_clicked`. The following query needs no application table; it returns two bound constants through PostgreSQL. The component's module must match your actual slots module.

**POSTGRESQL HANDLER — SOURCE-CHECKED**

```
PROCEDURE btnQuery_clicked(sender, event, form)
  LOCAL row
  TRY
    dbPostgres.Open()
    row = dbPostgres.QueryOne( ;
      "SELECT ?::integer AS id, ?::text AS name", ;
      [101, "Kokum reader"])
    form.lblStatus.Text = row["name"]
  CATCH error
    form.lblStatus.Text = "Database request failed"
  FINALLY
    dbPostgres.Close()
  ENDTRY
ENDPROC
```

Expected `lblStatus` after a successful live connection: Kokum reader. The provider converts ? placeholders into \$1, \$2 and sends their values through `PQexecParams`. Question marks inside quoted strings, identifiers, comments or dollar-quoted text are not placeholders.

### Useful query shapes

**FRAGMENTS — REQUIRE AN OPEN CONNECTION AND customers TABLE**

```
rows = dbPostgres.Query( ;
  "SELECT id,name FROM customers ORDER BY id LIMIT 20")
row = dbPostgres.QueryOne( ;
  "SELECT id,name FROM customers WHERE id = ?", [101])
count = dbPostgres.QueryValue("SELECT COUNT(*) FROM customers")
tblCustomers = dbPostgres.QueryTable( ;
  "SELECT id,name FROM customers ORDER BY id LIMIT 20", NULL, .T.)
```

Begin, Commit and Rollback use the same connection. Do not share a live DATABASE object across unrelated task runtimes. A provider-neutral method name does not imply that SQL types, functions and server behavior are identical across TigerDB and PostgreSQL.

Source basis: [K06], [K07], [V1]. No live PostgreSQL server was used for this book.

## 05 • REMOTE DATABASE PROVIDERS

## Turn network JSON into a new table

For a MAP, an ARRAY of MAP records or a JSON string representing those shapes, the database API can infer a new table. The destination must be an initialized DATABASE component, not the KCONNECT built-in Remote FlatDB session.

**FRAGMENT — dbPostgres OPEN; manual\_import MUST NOT EXIST**

```
LOCAL payload AS ARRAY
payload = [ ;
  {"name": "Kokum", "active": .T., "count": 2}, ;
  {"name": "Network lab", "active": .F., "count": 1} ;
]
CREATE TABLE manual_import ;
  ON DATA dbPostgres ;
  FROM payload
```

Kokum inserts a first primary-key column named `rec_no` and assigns values in source order. Input must not contain `rec_no` in any ASCII letter case. This operation creates a table; it is not an append, upsert or automatic schema migration.

### Fetching first, importing second

**INTEGRATION FRAGMENT — approvedUrl MUST RETURN OBJECT RECORDS**

```
LOCAL response AS MAP
response = URLGET(approvedUrl, {"MaxResponseBytes": 1048576})
IF response["Ok"] AND response["JsonError"] == ""
  CREATE TABLE manual_api_import ;
    ON DATA dbPostgres ;
    FROM response["Json"]
ENDIF
```

Validate the response's application schema before importing. The first record establishes field names and order. Later records may omit fields but may not introduce new fields; incompatible types are rejected before database changes. All-null fields become text. Nested maps/arrays map to PostgreSQL JSONB, or the corresponding provider representation.

### Schema-only and failure behavior

**SCHEMA ONLY — NO DATA ROWS INSERTED**

```
dbPostgres.CreateTableFromJson("manual_schema", payload, .T.)
```

The method form is equivalent to the language statement. Creation/insertion is transactional for supported DATABASE providers; existing tables are an error. This does not add a transaction API to KEXECUTE/KQUERY. An HTTP success alone does not prove that the JSON is an acceptable import shape.

Source basis: [K05], [K06], docs/CREATE\_TABLE\_FROM\_JSON.md, [K09].

## 06 • WEBSOCKET CLIENTS

# Build a local echo application

A WebSocket stays open for messages in both directions. Kokum's WebSocketClient is a native, nonvisual component; its worker handles I/O while your owner session consumes queued events. It supports ws:// and verified wss://. There is no generic application WebSocket server API in this release. [K08, R4]

## Start the disposable echo endpoint

### OPTIONAL PYTHON LAB DEPENDENCY; NOT A KOKUM DEPENDENCY

```
cd "$HOME/network-lab"
python3 -m venv .venv
. .venv/bin/activate
python -m pip install "websockets==16.0"
python ws_echo.py
```

### ws\_echo.py — LOCAL ECHO SERVER

```
from websockets.sync.server import serve

def echo(connection):
    for message in connection:
        connection.send(message)

with serve(echo, "127.0.0.1", 18765,
           compression=None, max_size=1048576) as server:
    server.serve_forever()
```

## Create the GUI components

| ID / type                                   | Setting or event                                                                                           |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------|
| wsFeed / WebSocketClient                    | Auto Connect=false; URL=ws://127.0.0.1:18765/echo; Subprotocols empty; Reconnect=false for the first test. |
| timerPoll / Timer                           | Interval=100–250 ms; Running=false; Repeat=true.                                                           |
| btnConnect, btnSend, btnDisconnect / Button | Connect clicked to the same-named procedures shown next.                                                   |
| txtLog / TextArea; lblStatus / Label        | A read-only log and a status label. Connect timerTick to timerPoll_timerTick.                              |

Keep the project's existing MODULE declaration and managed PUBLIC block. If you build from the shipped WebSocket project, remove its old loaded binding and authentication placeholder handler before using this local echo example.

Source basis: [K08], [R6], [V1].

**06 • WEBSOCKET CLIENTS**

## Connect and send from ordinary slots

Insert these procedures into the existing slots module and export them. The Designer supplies wsFeed AS WEBSOCKET and txtLog AS STRING through its managed declarations; do not declare those variables again outside the managed block.

**ADD THE EXPORTS ONCE**

```
EXPORT PROCEDURE btnConnect_clicked
EXPORT PROCEDURE btnSend_clicked
EXPORT PROCEDURE timerPoll_timerTick
EXPORT PROCEDURE btnDisconnect_clicked
```

**CONNECT AND SEND HANDLERS**

```
PROCEDURE btnConnect_clicked(sender, event, form)
  TRY
    wsFeed.SetUrl("ws://127.0.0.1:18765/echo")
    wsFeed.Connect()
    form.timerPoll.Running = .T.
    form.lblStatus.Text = "Connected"
  CATCH error
    form.lblStatus.Text = "Connection failed: " + STR(error)
  ENDTRY
ENDPROC

PROCEDURE btnSend_clicked(sender, event, form)
  wsFeed.SendJSON({"message": "Hello from Kokum"})
ENDPROC
```

Connect() establishes the configured connection and may block the invoking slot until the connect/TLS/upgrade budget completes. Network frame I/O then runs on the native worker. A reconnect strategy is discussed later; the first echo test keeps automatic reconnection disabled.

### Run the project

**USE THE BUNDLE NAME IN YOUR MANIFEST**

```
kokumc project check kokum.toml
kokumc bundle kokum.toml
kokumvm dist/your-application.kapp
```

Start ws\_echo.py before clicking Connect. Click Send, then observe the echoed JSON in txtLog after the Timer runs. Click Disconnect to close cleanly. The echo server does not require any authentication headers or application subscription format.

**DEVELOPMENT DIAGNOSTICS** The connection handler includes STR(error) for a local lab. A deployed public-facing UI should show a generic message and keep detailed diagnostics in trusted logs.

Source basis: [K08], [V1].

## 06 • WEBSOCKET CLIENTS

## Drain events on the owner session

## TIMER AND DISCONNECT HANDLERS

```

PROCEDURE timerPoll_timerTick(sender, event, form)
    LOCAL batch AS ARRAY
    LOCAL item AS MAP
    batch = wsFeed.Drain(50)
    FOR EACH item IN batch
        IF item["type"] == "text"
            txtLog = txtLog + item["text"] + "\n"
        ENDIF
    NEXT
ENDPROC

PROCEDURE btnDisconnect_clicked(sender, event, form)
    form.timerPoll.Running = .F.
    wsFeed.Close(1000, "Done", 3000)
    form.lblStatus.Text = "Disconnected"
ENDPROC

```

A bounded `Drain(50)` prevents an unbounded receive loop from monopolizing a UI timer tick. Incoming frames can arrive faster than your interface can render them. Aggregate repeated values, update a graph in batches, and cap the displayed log rather than appending forever.

### Expected echo

**txtLog CONTAINS THIS MESSAGE AFTER SEND**

```
{"message":"Hello from Kokum"}
```

The timer is invoked by the GUI runtime on its owner session, not by the WebSocket worker. Only that owner session changes `txtLog` or a form property. The WEBSOCKET value itself cannot be transferred into an ASYNC task.

### Closing the form

Before disposing the form/session, stop its timer and close the WebSocket deliberately. Avoid leaving a callback that assumes a now-closed form still exists. `Disconnect()` is an alias for a normal close. A user-requested close does not trigger automatic reconnect.

**RESOURCE OWNERSHIP** Use the component as the persistent connection owner. A new PUBLIC variable of type WEBSOCKET is not a constructor and does not create a usable client on its own.

Source basis: [K08], [K13], [V1].

## 06 • WEBSOCKET CLIENTS

## Message APIs and event envelopes

| Call                                     | Behavior                                                             |
|------------------------------------------|----------------------------------------------------------------------|
| Receive([timeoutMs]) / Poll([timeoutMs]) | Next event MAP, or NULL. Omitted timeout is nonblocking.             |
| ReceiveText([timeoutMs])                 | Next matching text payload or NULL; other event kinds remain queued. |
| ReceiveJSON([timeoutMs])                 | Decode a matching text payload as JSON; handle parse failures.       |
| Drain([maximumCount])                    | ARRAY of queued event MAPs, useful for bounded GUI work.             |
| ClearMessages()                          | Discard currently queued incoming events.                            |
| SendText(text) / SendJSON(value)         | Queue a text frame; JSON is encoded by Kokum.                        |
| SendBinary(bytes)                        | Queue ARRAY integers 0..255 as binary payload.                       |
| Ping([payload])                          | Send a control ping; observe pong events.                            |

### FRAGMENTS — AN OPEN COMPONENT IS REQUIRED

```
wsFeed.SendText("hello")
wsFeed.SendJSON({"action": "subscribe", "topic": "prices"})
wsFeed.SendBinary([1, 2, 255])
wsFeed.Ping("health")
```

| Event type   | Additional fields   |
|--------------|---------------------|
| connected    | protocol            |
| text         | text, size          |
| binary       | data, size          |
| pong         | text/data, size     |
| closed       | code, reason, clean |
| error        | message             |
| reconnecting | attempt, delayMs    |

Every event includes type and timestamp. Use the shown lower-case event keys. Check event type before interpreting payload. A clean close and a transport error are not interchangeable. WebSocket client frames are masked; that protocol masking is not encryption. [R4]

Source basis: [K08].

## 06 • WEBSOCKET CLIENTS

## Authenticate and verify a WSS endpoint

Keep Auto Connect disabled until authentication and query settings have been configured. Put real secrets in trusted native configuration, never browser JavaScript. The following are alternative service-dependent helpers, not a requirement to send every credential at once.

### CONFIGURATION FRAGMENT — BEFORE CONNECT

```
wsFeed.SetBearerToken(accessToken)
wsFeed.SetFeedToken(feedToken)
wsFeed.SetApiKey(apiKey)
wsFeed.SetHeader("X-Client-Code", clientCode)
wsFeed.SetCookie("session=demo")
wsFeed.SetQueryParameter("client", "kokum")
wsFeed.Connect()
```

| Helper                       | Default header                    |
|------------------------------|-----------------------------------|
| SetBearerToken(token)        | Authorization: Bearer <token>     |
| SetFeedToken(token)          | x-feed-token                      |
| SetApiKey(value)             | x-api-key                         |
| SetBasicAuth(user, password) | Authorization with Basic encoding |

Bearer/feed/API-key helpers can take an alternate header name. Reserved handshake fields such as Host, Upgrade and Sec-WebSocket-Key cannot be overridden. Do not assume the URL API's HTTPS-only authentication-helper rule is also enforced by WebSocket helpers: select a verified wss:// endpoint yourself.

### TLS settings belong to the component

Verify TLS defaults on. A private CA file establishes trust; TLS Server Name selects an explicit certificate identity/SNI when required. Client Certificate and Client Key optionally provide mutual TLS. The client supports TLS 1.2 or newer. Fix trust or identity errors rather than turning off verification.

### Authentication after opening

#### ONLY WHEN THE REMOTE SERVICE DEFINES THIS MESSAGE FORMAT

```
wsFeed.SendJSON({"action": "authenticate", "token": accessToken})
```

The server, not Kokum, defines authentication messages, token expiry, subscriptions and acknowledgements. A successfully queued SendJSON is not proof that the service accepted the login or subscription.

Source basis: [K08], [R4].

## 06 • WEBSOCKET CLIENTS

## Bound queues and reconnect intentionally

**CONFIGURE WHILE DISCONNECTED; INSPECT DURING OPERATION**

```
wsFeed.SetReconnect(.T., 1000, 10)
? wsFeed.State
? wsFeed.PendingMessages
? wsFeed.PendingSends
? wsFeed.DroppedMessages
? JSONENCODE(wsFeed.Snapshot())
```

The arguments enable reconnection, select a base delay in milliseconds and set the maximum attempts; zero attempts means unlimited. Backoff is bounded and reconnecting events report the attempt and delay. A manual Close/Disconnect does not restart the client.

| Native default                | Value       |
|-------------------------------|-------------|
| Connect timeout               | 10,000 ms   |
| I/O timeout                   | 30,000 ms   |
| Maximum assembled message     | 8 MiB       |
| Incoming/outgoing queue limit | 1,024       |
| Automatic ping interval       | 0: disabled |
| Base reconnect delay          | 1,000 ms    |

Designer templates can store explicit values that differ from native defaults. Inspect the component rather than assuming every existing form uses these settings. Set finite message and queue limits suitable for the workload.

### After reconnect

Treat reconnection as a new application session unless the service explicitly guarantees continuity. Send authentication/subscription messages again only when the service protocol requires them; detect duplicate deliveries and gaps at the application layer. Kokum does not implement your feed's sequence-number or exactly-once contract.

### Health and diagnostics

Snapshot exposes status and counters, not private keys, authentication headers or native socket handles. Track queue depth, dropped messages, LastError, close reason and reconnect count. A continuously increasing DroppedMessages counter means the consumer cannot keep up; enlarge neither the log nor the queue without a memory plan.

Source basis: [K08].

## 07 • HTTP GET AND POST

# Understand the native URL contract

**SIGNATURES — NOT SOURCE TO EXECUTE**

```
URLGET(url [, options]) -> MAP
URLPOST(url, body [, options]) -> MAP
```

Both native built-ins support HTTP and verified HTTPS. They are synchronous calls at this boundary; Chapter 9 shows how to run them with tasks. The same transport and response parser are shared by interpreter/IR execution, bytecode and KokumVM. A browser fetch call is not involved.

**Start with an endpoint you control**

The next three pages define a small Kokum HTTP service. Save its complete lab.kokum source, then save server.conf beside it and start it on 127.0.0.1:18080. All basic GET/POST examples use this service, so a public test website is unnecessary.

| Route                 | What the lab demonstrates                                |
|-----------------------|----------------------------------------------------------|
| GET /hello?name=Kokum | Decoded query input and a JSON response                  |
| POST /echo            | The received method, raw body and parsed JSON            |
| GET /redirect         | A same-origin 302 redirect to /hello                     |
| GET /slow             | A 150 ms handler delay for timeout demonstrations        |
| GET /missing          | A valid 404 JSON response                                |
| GET /badjson          | A 200 response claiming JSON but containing invalid JSON |
| GET /health           | The built-in backend health endpoint                     |

**LAB BOUNDARY** These endpoints are unauthenticated and deliberately local. They are examples of the backend host, not a production API. The TLS and authentication labs use a separate local helper later in the book.

Use reasonable time and response-size limits even for a local endpoint. A valid non-2xx HTTP reply returns a response MAP with Ok=.F.; transport, TLS, invalid-option and malformed-protocol failures are exceptions.

Source basis: [K09], [K12].

## 07 • HTTP GET AND POST

## Lab service: declarations, greeting and echo

**lab.kokum — PART 1 OF 2**

```

MODULE manual.lab VERSION "1.0.0"
EXPORT FUNCTION Main
EXPORT FUNCTION Hello
EXPORT FUNCTION Echo
EXPORT FUNCTION Redirect
EXPORT FUNCTION Slow
EXPORT FUNCTION Missing
EXPORT FUNCTION BadJson

FUNCTION Main AS INTEGER
    RETURN 0
ENDFUNC

FUNCTION Hello(request AS MAP) AS MAP
    LOCAL name AS STRING
    name = "reader"
    IF HASKEY(request["Query"], "name")
        name = request["Query"]["name"]
    ENDIF
    RETURN {"Json": {"message": "Hello, " + name}}
ENDFUNC

FUNCTION Echo(request AS MAP) AS MAP
    RETURN {"Json": {
        "method": request["Method"], ;
        "body": request["Body"], ;
        "json": request["Json"] ;
    }}
ENDFUNC

```

The MODULE identifies this source as manual.lab. Each handler is exported because kokumvm serve resolves exported callbacks. Main is the application entry and does not itself start a listener; the serve command and route configuration supply that behavior.

The greeting safely defaults to reader when the name query key is missing. Echo returns only its demonstration body fields; it does not echo Authorization or Cookie values. JSON under the Json response key is serialized by the backend host with the appropriate content type.

**COPY BOTH PARTS** Append the next page's functions to the same lab.kokum file. Do not create a second file with another MODULE declaration for this single-source lab.

Source basis: [K12], [V1].

## 07 • HTTP GET AND POST

## Lab service: redirects and controlled failures

**lab.kokum — PART 2 OF 2**

```

FUNCTION Redirect(request AS MAP) AS MAP
    RETURN {"Status": 302, ;
        "Headers": {"Location": "/hello?name=redirect"}}
ENDFUNC

FUNCTION Slow(request AS MAP) AS MAP
    SLEEP(150)
    RETURN {"Json": {"ready": .T.}}
ENDFUNC

FUNCTION Missing(request AS MAP) AS MAP
    RETURN {"Status": 404, "Json": {"error": "not found"}}
ENDFUNC

FUNCTION BadJson(request AS MAP) AS MAP
    RETURN {"ContentType": "application/json", "Body": "{broken}"}
ENDFUNC

```

### Why these handlers exist

Redirect returns a Location field for the client's redirect policy. Slow sleeps for 150 ms so a much shorter request deadline can be observed. Missing is still a syntactically valid HTTP reply. BadJson deliberately separates JSON-content failure from HTTP-framing failure.

Do not raise every non-2xx result as though no response arrived. Likewise, do not assume response["Json"] is populated just because the status is 200. The later client examples inspect Status, Ok and JsonError independently.

### Source check

**RUN IN network-lab**

```

kokumc check lab.kokum
kokumc compile lab.kokum -o lab.kbc

```

Compilation alone does not open port 18080. The next page starts the exported handlers using the route file. Stop the server with Ctrl+C after finishing the exercises.

Source basis: [K09], [K12], [V1].

## 07 • HTTP GET AND POST

## Lab route file and startup

**server.conf**

```
[server]
listen=127.0.0.1
port=18080
workers=4
queue_capacity=32
max_request_bytes=1048576
max_body_bytes=786432
read_timeout_ms=5000
write_timeout_ms=5000
graceful_shutdown_ms=5000
health_path=/health
trace=false
[route hello]
method=GET
path=/hello
handler=Hello
parameters=1
[route echo]
method=POST
path=/echo
handler=Echo
parameters=1
[route redirect]
method=GET
path=/redirect
handler=Redirect
parameters=1
[route slow]
method=GET
path=/slow
handler=Slow
parameters=1
[route missing]
method=GET
path=/missing
handler=Missing
parameters=1
[route badjson]
method=GET
path=/badjson
handler=BadJson
parameters=1
```

**TERMINAL 1 — KEEP RUNNING**

```
kokumvm serve lab.kbc --config server.conf
```

In a second terminal run `curl http://127.0.0.1:18080/health`, or begin the Kokum GET example. Every route is intentionally fixed; no request is treated as an arbitrary file path or destination URL.

Source basis: [K12], [V1].

## 07 • HTTP GET AND POST

## GET with query parameters and JSON results

**get.kokum — LOCAL LAB**

```

FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  TRY
    response = URLGET( ;
      "http://127.0.0.1:18080/hello", ;
      {"Query": {"name": "Kokum reader"}, ;
      "TimeoutMs": 3000} ;
    )
    ? response["Status"]
    ? response["Json"]["message"]
  CATCH error
    ? "Request failed: " + STR(error)
  RETURN 1
ENDTRY
RETURN 0
ENDFUNC

```

**TERMINAL 2**

```
kokumc run-bytecode get.kokum
```

**EXPECTED OUTPUT**

```

200
Hello, Kokum reader

```

Query values are percent-encoded and appended to the initial URL. Arrays become repeated parameters. A literal space is encoded rather than inserted into the request target. A URL fragment is not transmitted to the server.

**ADAPTABLE FRAGMENT — approvedUrl IS AN APPROVED ENDPOINT**

```

response = URLGET(approvedUrl, { ;
  "Query": {"page": 2, "tag": ["database", "web API"]}, ;
  "Headers": {"Accept": "application/json"}, ;
  "TimeoutMs": 5000, "MaxResponseBytes": 1048576 ;
})

```

Header names and values are checked for invalid characters and CR/LF injection. Host, Content-Length, Transfer-Encoding and Connection remain transport-controlled. Do not place secrets in a query string: URLs can appear in application, proxy and access logs.

Source basis: [K09], [V1].

## 07 • HTTP GET AND POST

## POST a MAP or validated JSON string

**post\_json.kokum — LOCAL LAB**

```

FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  response = URLPOST( ;
    "http://127.0.0.1:18080/echo", ;
    {"name": "Kokum", "count": 2, "active": .T.}, ;
    {"TimeoutMs": 3000} ;
  )
  IF response["Ok"]
    ? response["Json"]["json"]["name"]
    ? response["Json"]["json"]["count"]
  ELSE
    ? "HTTP failure: " + STR(response["Status"])
  ENDIF
  RETURN 0
ENDFUNC

```

The MAP is encoded as JSON automatically. Expected output is Kokum, then 2. An ARRAY body is also JSON by default. The echo service returns the decoded request under its own json key, so response["Json"]["json"] accesses that nested payload.

### A string containing JSON needs an explicit mode

**VALIDATED RAW JSON FRAGMENT**

```

response = URLPOST( ;
  "http://127.0.0.1:18080/echo", ;
  "{ \"name\": \"Kokum\" }", ;
  {"BodyType": "json"} ;
)

```

Without BodyType="json", a STRING is plain text. With JSON mode, malformed input raises KUR1008 before a network connection is opened. Avoid JSONENCODE followed by an implicit text POST unless the API intentionally expects text.

### Return status is application-specific

The echo route returns 200; an actual resource-creation API may return 201 or another valid status. Check the documented success codes and response schema of that API. A queued or completed request is not an application-level guarantee that a transaction committed.

Source basis: [K09], [V1].

07 • HTTP GET AND POST

# Text, URL-encoded forms, XML and empty bodies

post\_bodies.kokum — LOCAL LAB

```
FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  LOCAL endpoint AS STRING
  endpoint = "http://127.0.0.1:18080/echo"
  response = URLPOST(endpoint, "Hello from Kokum")
  ? response["Json"]["body"]
  response = URLPOST(endpoint, ;
    {"name": "Kokum reader", "tag": ["one", "two"]}, ;
    {"BodyType": "form"})
  ? response["Json"]["body"]
  response = URLPOST(endpoint, "<hello>Kokum</hello>", ;
    {"ContentType": "application/xml; charset=utf-8"})
  ? response["Json"]["body"]
  response = URLPOST(endpoint, NULL)
  ? response["Status"]
  RETURN 0
ENDFUNC
```

Run kokumc run-bytecode post\_bodies.kokum. The text and XML are echoed unchanged. The form result contains name=Kokum%20reader&tag=one&tag=two. NULL produces a zero-length request body, not the JSON text null.

| Body mode                  | Accepted input / default content type                              |
|----------------------------|--------------------------------------------------------------------|
| Automatic STRING / text    | STRING or NULL; text/plain; charset=utf-8                          |
| Automatic MAP/ARRAY / json | JSON body; application/json; charset=utf-8                         |
| form                       | MAP or NULL; application/x-www-form-urlencoded; charset=utf-8      |
| ContentType override       | Useful for textual XML or another explicitly agreed representation |

Do not supply ContentType and Headers["Content-Type"] together. Kokum knows the body length and sends Content-Length; it does not send chunked uploads. The body preparation limit is 64 MiB. Multipart, file-stream uploads, response decompression and streaming downloads are not part of this API.

Source basis: [K09], [V1].

## 07 • HTTP GET AND POST

## Verified HTTPS without disabling certificate checks

The local TLS fixture is listed in Appendix D. Save `tls_lab.py` in `network-lab`. Generate this disposable certificate/key there; the certificate explicitly names `localhost` and `127.0.0.1`. It is trusted only by the examples that pass its file as `CAFile`.

**TERMINAL 1 — PRIVATE, DISPOSABLE LOCAL TLS LAB**

```
openssl req -x509 -newkey rsa:2048 -sha256 -nodes \
  -days 7 -subj /CN=localhost \
  -addext "subjectAltName=DNS:localhost,IP:127.0.0.1" \
  -keyout lab-key.pem -out lab-cert.pem
chmod 600 lab-key.pem
python3 tls_lab.py
```

**https.kokum — TERMINAL 2 IN THE SAME DIRECTORY**

```
FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  TRY
    response = URLPOST( ;
      "https://127.0.0.1:18443/echo", ;
      {"name": "Kokum"}, ;
      {"CAFile": "lab-cert.pem", "TimeoutMs": 3000} ;
    )
    ? response["Status"]
    ? response["Json"]["received"]["name"]
  CATCH error
    ? STR(error)
  RETURN 1
ENDTRY
RETURN 0
ENDFUNC
```

Run `kokumc run-bytecode https.kokum`. Expected output: 200, then Kokum. Verification stays enabled. The helper accepts a small JSON body and returns a JSON envelope; it is not a general HTTPS application server.

`CAFile` is a PEM trust file, not a private key or client certificate. `TLSservername` is an optional SNI/identity override when the address used for TCP differs from the intended certificate name. It does not rewrite the TCP destination.

**PRODUCTION** Replace the lab trust file with your organization's CA or the intended public trust configuration. `URLGET/URLPOST` do not expose client-certificate options in this release; `WebSocket` and database TLS settings are separate APIs.

Source basis: [K09], [R7], [V1].

## 07 • HTTP GET AND POST

## Separate HTTP failure, JSON failure and transport failure

errors.kokum — LOCAL LAB

```

FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  response = URLGET("http://127.0.0.1:18080/missing")
  ? response["Status"]
  ? response["Ok"]
  response = URLGET("http://127.0.0.1:18080/badjson")
  ? response["Json"] == NULL
  ? LEN(response["JsonError"]) > 0
  TRY
    response = URLGET("http://127.0.0.1:18080/slow", ;
      {"TimeoutMs": 50, "ReadTimeoutMs": 50})
  CATCH error
    ? STR(error)
  ENDTRY
  RETURN 0
ENDFUNC

```

### Read the output correctly

The first request prints 404 and .F.; it did receive a valid response. The second request prints .T. twice: the decoded Json is NULL and JsonError is nonempty. The final request raises a timeout diagnostic because the 150 ms endpoint exceeds a 50 ms budget. Exact timeout code can depend on which budget expires first.

### Application response validation

Check response["Ok"] or your permitted status set, then JsonError, then the required JSON shape and keys. A valid JSON literal null also decodes to NULL but has an empty JsonError. Do not collapse those cases into one generic "HTTP failed" condition.

### Protocol safety

The shared parser supports Content-Length, chunked and connection-close body framing. It rejects malformed status lines, conflicting/invalid framing, prohibited trailers and oversized or truncated responses. An unsupported coding is not automatically decoded. HTTP 101 is not a way to obtain a WebSocket through URLGET.

Source basis: [K09], [K14], [V1].

# Follow redirects with a bounded policy

redirect.kokum — LOCAL LAB

```
FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  response = URLGET("http://127.0.0.1:18080/redirect")
  ? response["Status"]
  ? response["RedirectCount"]
  ? response["Json"]["message"]
  response = URLGET("http://127.0.0.1:18080/redirect", ;
    {"FollowRedirects": .F.})
  ? response["Status"]
  ? response["Headers"]["location"]
  RETURN 0
ENDFUNC
```

| Status    | GET behavior    | POST behavior                                   |
|-----------|-----------------|-------------------------------------------------|
| 301 / 302 | Continue as GET | Change to GET; discard body and related headers |
| 303       | Continue as GET | Change to GET; discard body and related headers |
| 307 / 308 | Continue as GET | Keep POST and exact prepared body bytes         |

FollowRedirects defaults to .T. and MaxRedirects to 5, with an allowed range of 0..20. MaxRedirects=0 is not the same as FollowRedirects=.F.: the former errors when a follow would occur; the latter returns the original response untouched.

Relative Location references are resolved against the current URL. Query options are applied only once, to the initial request. The returned Url and RedirectCount describe the final result; only the final body is retained. One total deadline covers admission and the whole redirect chain, not a fresh deadline at each hop.

Source basis: [K09], [K10], [R1], [R3], [V1].

## 08 • REDIRECTS, AUTH AND COOKIES

## Keep credentials at their intended origin

An origin consists of scheme, host and effective port. A different port, subdomain or scheme is a different origin, even on the same machine. Kokum strips sensitive and unknown/custom headers on an origin change and does not restore them on a later return to the first origin.

| Boundary                 | Kokum rule                                                                                                                               |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| HTTPS → HTTP             | Always blocked, regardless of other redirect flags.                                                                                      |
| Cross-origin credentials | Authorization, Cookie, Proxy-Authorization, API-key and unknown/custom headers are removed.                                              |
| Cross-origin POST replay | Blocked by default even for an empty POST; AllowCrossOriginBody permits an explicitly approved body, not credentials or a TLS downgrade. |
| TLS identity override    | TLSServerName is cleared on an origin change; the new URL's identity is checked.                                                         |
| Same-origin redirect     | Explicit credentials/cookies can remain; normal method/body rules still apply.                                                           |

### Authentication helpers

#### ALTERNATIVE FRAGMENTS — DO NOT COMBINE AUTH METHODS

```
response = URLGET(approvedHttpsUrl, ;
  {"BearerToken": accessToken, "TimeoutMs": 5000})
response = URLPOST(approvedHttpsUrl, {"action": "check"}, ;
  {"BasicAuth": {"Username": userName, "Password": password}})
```

BasicAuth, a nonempty BearerToken and a raw Authorization header are mutually exclusive. These helpers are preemptive; a 401 response does not trigger a hidden retry. They do not obtain or refresh an OAuth token. HTTPS is required by default; AllowInsecureAuth is an explicit plaintext-test opt-in.

**NOT A DESTINATION SANDBOX** Redirect protection does not validate arbitrary user-selected destinations against your private network policy. Secrets placed in query strings, allowed bodies or ostensibly harmless headers remain the application's responsibility.

Source basis: [K10], [R1], [R3].

## 08 • REDIRECTS, AUTH AND COOKIES

## Try explicit credentials and cookie omission locally

**auth.kokum — USE THE LOCAL TLS FIXTURE**

```

FUNCTION Main AS INTEGER
  LOCAL response AS MAP
  LOCAL options AS MAP
  options = {"CAFile": "lab-cert.pem", ;
    "BearerToken": "demo-token", "Cookie": "session=demo"}
  response = URLGET("https://127.0.0.1:18443/echo", options)
  ? response["Json"]["hasAuth"]
  ? response["Json"]["hasCookie"]
  options["CookiePolicy"] = "omit"
  response = URLGET("https://127.0.0.1:18443/echo", options)
  ? response["Json"]["hasCookie"]
  response = URLGET("https://127.0.0.1:18443/echo", ;
    {"CAFile": "lab-cert.pem", "BasicAuth": ;
      {"Username": "demo", "Password": "demo-only"}})
  ? response["Json"]["hasAuth"]
  RETURN 0
ENDFUNC

```

**EXPECTED OUTPUT**

```

.T.
.T.
.F.
.T.

```

The helper reports only whether the headers exist; it does not authenticate a real account and never echoes secret values. Keep the disposable lab certificate and demo credentials isolated from production configuration.

### Two cookie policies

manual sends only a cookie explicitly supplied for this call. omit suppresses outgoing Cookie and Cookie2, but it does not bypass input validation or hide response headers. No automatic jar, disk persistence or shared per-user cookie state is created.

Use either the Cookie option or Headers["Cookie"], not both. Repeated Set-Cookie response fields remain in response["HeaderValues"]["set-cookie"]. Those fields are not automatically converted into request cookies. Do not copy an entire Set-Cookie value, with Path/Secure/HttpOnly attributes, into Cookie.

**RAW HEADERS** The HTTPS-only guard applies to helper-generated Basic/Bearer credentials. Explicit raw Authorization headers and manually supplied cookies remain caller-controlled and would be unencrypted on HTTP.

Source basis: [K09], [K10], [V1].

## 09 • ASYNC AND CONCURRENCY

# Launch independent HTTP tasks

**async.kokum — LOCAL LAB**

```

ASYNC FUNCTION Fetch(url AS STRING) AS MAP
    RETURN URLGET(url, {"TimeoutMs": 3000})
ENDFUNC

FUNCTION Main AS INTEGER
    LOCAL first AS TASK
    LOCAL second AS TASK
    LOCAL response AS MAP
    TRY
        first = Fetch("http://127.0.0.1:18080/hello?name=one")
        second = Fetch("http://127.0.0.1:18080/hello?name=two")
        response = AWAIT first
        ? response["Json"]["message"]
        response = AWAIT second
        ? response["Json"]["message"]
    CATCH error
        ? STR(error)
    RETURN 1
ENDTRY
RETURN 0
ENDFUNC

```

**USE A VM-BACKED PATH FOR ACTUAL OVERLAP**

```
kokumc run-bytecode async.kokum
```

Calling an ASYNC FUNCTION returns TASK. Its declared AS MAP type describes the result after AWAIT, not the immediate call value. Launch both tasks before awaiting either to permit overlap. Main itself is not ASYNC.

Arguments, nested maps and arrays cross the task boundary as snapshots. Each request owns its headers, cookies, body, socket and TLS state. Retrieved results are materialized into the owner runtime. Repeated Result()/AWAIT accesses reuse the owner's cached value; they do not send the request again.

**EXECUTION MODEL** KokumVM and run-bytecode use the worker engine. Ordinary AST/IR reference ASYNC calls execute inline; source requiring named-thread lowering may be routed through bytecode even under other CLI modes. Do not use AST/IR timing to benchmark parallel HTTP.

Source basis: [K11], [K14], [V1].

09 • ASYNC AND CONCURRENCY

# Distinguish task waits from HTTP deadlines

task\_timeout.kokum — LOCAL LAB

```
ASYNC FUNCTION FetchSlow AS MAP
  RETURN URLGET("http://127.0.0.1:18080/slow", ;
    {"TimeoutMs": 3000})
ENDFUNC

FUNCTION Main AS INTEGER
  LOCAL pending AS TASK
  LOCAL response AS MAP
  pending = FetchSlow()
  TRY
    response = pending.Await(10)
  CATCH error
    ? "Wait ended; the request may still be running."
  ENDTRY
  response = AWAIT pending
  ? response["Json"]["ready"]
  RETURN 0
ENDFUNC
```

The first wait may time out; the second retrieves the same request’s eventual result. A wait timeout does not cancel the request, reverse its side effects or grant permission to send a POST again. Retain the TASK while its result matters.

| Budget                  | What it limits                                                              |
|-------------------------|-----------------------------------------------------------------------------|
| Task submission / queue | Configured by the worker pool; earlier queue waiting is not HTTP TimeoutMs. |
| TimeoutMs               | HTTP admission plus the transport and redirects after preparation.          |
| ConnectTimeoutMs        | TCP/TLS connection budget, constrained by total remaining time.             |
| ReadTimeoutMs           | Read inactivity, not an independent unlimited whole-response period.        |
| pending.Await(ms)       | Only this caller’s wait for a TASK result.                                  |

System DNS resolution is blocking and cannot be interrupted exactly at the HTTP deadline. Snapshot copying, body preparation and result conversion are not preemptive deadline work. These budgets are not hard real-time cancellation guarantees.

Source basis: [K09], [K11], [V1].

09 • ASYNC AND CONCURRENCY

# Named RUN jobs and process-wide admission

named\_run.kokum — LOCAL LAB

```
FUNCTION Fetch(url AS STRING) AS MAP
  RETURN URLGET(url, {"TimeoutMs": 3000})
ENDFUNC

FUNCTION Main AS INTEGER
  LOCAL result AS MAP
  LOCAL url AS STRING
  url = "http://127.0.0.1:18080/hello"
  RUN Fetch(url) AS fetchJob NOWAIT :result
  WAIT FOR fetchJob
  ? result["Status"]
  ? result["Json"]["message"]
  RETURN 0
ENDFUNC
```

Keep RUN ... AS ... on one source line as shown. WAIT FOR fetchJob collects the named result; the ordinary function need not be declared ASYNC. Named-job inspection includes THREADSTATUS, THREADDONE, THREADERROR and THREADID.

SET BEFORE STARTING THE VM

```
export KOKUM_TASK_WORKERS=4
export KOKUM_TASK_QUEUE_CAPACITY=256
export KOKUM_TASK_SUBMIT_TIMEOUT_MS=5000
export KOKUM_URL_MAX_CONCURRENT=16
```

| Setting                      | Default / supported range             |
|------------------------------|---------------------------------------|
| KOKUM_TASK_WORKERS           | 4 / 1..64                             |
| KOKUM_TASK_QUEUE_CAPACITY    | max(64, workers × 64) / 1..65536      |
| KOKUM_TASK_SUBMIT_TIMEOUT_MS | 5000 / 0..3600000                     |
| KOKUM_URL_MAX_CONCURRENT     | 16 / 1..1024; configuration read once |

One process-wide HTTP slot covers the full request and redirect chain across synchronous calls, task workers and backend sessions. Admission waiting consumes TimeoutMs. KUR7001 means the slot could not be acquired before expiry and no request was sent. It is not a FIFO scheduling guarantee or a bound on all queued/retained data.

Source basis: [K11], [V1].

## 09 • ASYNC AND CONCURRENCY

## Keep HTTP work out of a responsive GUI slot

Create a Button `btnFetch`, Timer `timerHttp` and Label `lblStatus`. Use a stopped repeating timer with a 100–250 ms interval. Connect `clicked` and `timerTick` to the procedures below. Keep your existing `MODULE` declaration and add the two exports once.

**GUI HTTP — DECLARATIONS AND START HANDLER**

```
EXPORT PROCEDURE btnFetch_clicked
EXPORT PROCEDURE timerHttp_timerTick
PUBLIC pending AS TASK

ASYNC FUNCTION GetGreeting AS MAP
  RETURN URLGET("http://127.0.0.1:18080/hello", ;
    {"TimeoutMs": 3000})
ENDFUNC

PROCEDURE btnFetch_clicked(sender, event, form)
  IF pending != NULL
    RETURN
  ENDIF
  TRY
    pending = GetGreeting()
    form.timerHttp.Running = .T.
    form.lblStatus.Text = "Loading"
  CATCH error
    form.lblStatus.Text = "Cannot start request"
  ENDTRY
ENDPROC
```

The click handler starts work and returns. It refuses a second click while an earlier task is retained. The next page's timer checks completion before reading `Result`, so the GUI does not perform a blocking `AWAIT` during a click or timer tick.

### Task ownership

Pass ordinary data into a worker; do not pass form, widget or `WebSocket` component values. The worker returns a transferable result. The owner session decides how to show it, whether to discard it after a form change, and when it is safe to release the task.

**LIFECYCLE** Closing a form does not imply HTTP cancellation. Keep a deliberate application lifetime for outstanding tasks. A form-specific timer must not later update controls that were disposed.

Source basis: [K11], [K13], [V1].

## 09 • ASYNC AND CONCURRENCY

## Complete the GUI request on its owning session

### GUI HTTP — TIMER HANDLER

```

PROCEDURE timerHttp_timerTick(sender, event, form)
  LOCAL response AS MAP
  IF pending == NULL
    RETURN
  ENDIF
  IF .NOT. pending.IsCompleted
    RETURN
  ENDIF
  TRY
    response = pending.Result()
    form.lblStatus.Text = STR(response["Status"])
  CATCH error
    form.lblStatus.Text = "Request failed"
  FINALLY
    pending = NULL
    form.timerHttp.Running = .F.
  ENDTRY
ENDPROC

```

The success path displays the HTTP status. In a real application, validate Ok, JsonError and the response schema before replacing UI data. The failure path intentionally hides transport details from the user-facing label.

### Bound a growing application

A concurrency limit does not bound completed MAPs held in an array or queued task snapshots containing large bodies. Launch a small fixed fan-out, drain results, release handles, and keep response-size limits appropriate to the screen.

Backend and task workers are separate pools. Blocking AWAIT inside a backend handler occupies that inbound worker while outgoing work proceeds. Increasing either pool blindly can increase memory and socket pressure rather than reduce latency.

### Shared state is not shared I/O ownership

MUTEX, ATOMICINTEGER and CHANNEL can coordinate in-process tasks, but do not make a DATABASE or WEBSOCKET handle freely transferable. Use value messages and stable ownership boundaries. Open a task's own Remote FlatDB session instead of borrowing another task's native resources.

Source basis: [K11], [V1].

## 10 • BACKEND SERVICES

# Host exported Kokum functions as an API

**APPLICATION SERVER ENTRY POINTS**

```
kokumvm serve application.kapp --config server.conf
kokumvm serve service.kbc --config server.conf \
  --ready-file service.ready
```

The host accepts HTTP connections, queues them in a bounded accepted-request queue and dispatches routes to persistent owner-thread KokumVM sessions. Each worker resolves exported callbacks once and reuses its session. A full accepted queue returns a structured 503 response instead of creating unbounded workers.

**Configuration controls**

| Setting                            | Purpose                                                            |
|------------------------------------|--------------------------------------------------------------------|
| listen / port                      | Bind the intended interface and TCP port.                          |
| workers / queue_capacity / backlog | Bound execution and admission resources.                           |
| max_request_bytes / max_body_bytes | Limit complete requests and bodies.                                |
| read_timeout_ms / write_timeout_ms | Bound socket I/O waits.                                            |
| graceful_shutdown_ms               | Drain policy; not forced termination of a running Kokum statement. |
| startup_handler / shutdown_handler | Exported zero-argument routines, once per worker session.          |
| health_path / trace                | Basic health endpoint and controlled diagnostics.                  |

A route supplies method, path and handler, plus optional module and an explicit parameters count of zero or one. Exact paths and :parameter segments are supported. The host can route methods beyond GET/POST, but the outgoing URL client still exposes only those two verbs.

**SEPARATE SECURITY MODEL** The backend host is HTTP-only and does not provide built-in application authentication. A public API needs a reviewed TLS termination, authorization and operational boundary. This is not permission to expose the LAN IDE.

Source basis: [K12].

## 10 • BACKEND SERVICES

## Understand request and response MAPs

| Request field                            | Meaning                                                  |
|------------------------------------------|----------------------------------------------------------|
| Method / Path / Target                   | Method, path without query, and original request target. |
| QueryString / Query                      | Raw query and decoded query MAP.                         |
| Params                                   | Decoded values from route segments such as :id.          |
| Headers                                  | Lower-case header names to values.                       |
| Body / Json                              | Raw request STRING and parsed JSON value when available. |
| RequestId                                | Server-generated correlation ID.                         |
| HttpVersion / RemoteAddress / RemotePort | Protocol and immediate network peer information.         |

### ROUTE FRAGMENT

```
[route customer]
method=GET
path=/customers/:id
handler=Customer
parameters=1
```

### HANDLER FRAGMENT — VALIDATE id BEFORE A DATABASE OPERATION

```
EXPORT FUNCTION Customer
FUNCTION Customer(request AS MAP) AS MAP
    RETURN {"Status": 200, "Json": { ;
        "id": request["Params"]["id"], ;
        "requestId": request["RequestId"] ;
    }}
ENDFUNC
```

A response MAP can contain Status, Headers, ContentType, Body and Json; Json takes precedence over Body. Returning a STRING gives text; other JSON-safe values can be serialized as JSON. NULL produces an empty 200 response. Protected framing headers cannot be overridden.

Malformed application/json requests are rejected before a handler runs. The host accepts Content-Length request bodies, not chunked requests. It uses one request per connection; keep-alive, pipelining, multipart, streaming, static hosting, middleware and WebSocket upgrades are not part of this backend foundation.

Source basis: [K12].

## 10 • BACKEND SERVICES

## A service with a Remote FlatDB connection per worker

**service\_db.kokum — LOCAL LAB**

```

MODULE manual.dbservice VERSION "1.0.0"
EXPORT FUNCTION Main
EXPORT FUNCTION Startup
EXPORT FUNCTION Shutdown
EXPORT FUNCTION People

FUNCTION Main AS INTEGER
    RETURN 0
ENDFUNC

FUNCTION Startup AS INTEGER
    KCONNECT USING kdb.conf
    IF KCONNECTED()
        KEXECUTE("CREATE TABLE IF NOT EXISTS manual_people (" + ;
            "id INTEGER PRIMARY KEY, name TEXT, city TEXT)")
    ENDIF
    RETURN 0
ENDFUNC

FUNCTION Shutdown AS INTEGER
    KDISCONNECT()
    RETURN 0
ENDFUNC

FUNCTION People(request AS MAP) AS MAP
    IF .NOT. KCONNECTED()
        KCONNECT USING kdb.conf
    ENDIF
    IF .NOT. KCONNECTED()
        RETURN {"Status": 503, ;
            "Json": {"error": "database unavailable"}}
    ENDIF
    TRY
        RETURN {"Json": KQUERY( ;
            "SELECT id,name FROM manual_people LIMIT 20")}
    CATCH error
        RETURN {"Status": 503, ;
            "Json": {"error": "database unavailable"}}
    ENDTRY
ENDFUNC

```

Each worker calls Startup once and owns its own KCONNECT session. People performs a bounded read and returns generic unavailable JSON on a problem. Startup does not make the entire server fail when the database is temporarily unavailable; the handler can make a later connection attempt.

Source basis: [K04], [K12], [V1].

## 10 • BACKEND SERVICES

## Configure and run the database-backed service

**db-service.conf**

```
[server]
listen=127.0.0.1
port=18082
workers=2
startup_handler=Startup
shutdown_handler=Shutdown

[route people]
method=GET
path=/people
handler=People
parameters=1
```

**KEEP kdb.conf AND manual.kkey BESIDE THE APPLICATION**

```
cd "$HOME/network-lab"
kokumc compile service_db.kokum -o service_db.kbc
kokumvm serve service_db.kbc --config db-service.conf
```

**INDEPENDENT TEST CLIENT**

```
curl http://127.0.0.1:18082/people
```

Keep the Remote FlatDB server running from Chapter 4. After the CRUD lab, the table is empty, so /people returns []. Without an available database connection, the handler returns a 503 envelope rather than exposing authentication, SQL or network internals.

### Persistent-worker behavior

Do not use one global connection shared outside the owning worker. Each worker has its own startup and shutdown calls, so startup logic should be idempotent and safe to run workers times. CREATE TABLE IF NOT EXISTS satisfies that requirement for this small example.

### Shut down cleanly

SIGINT/SIGTERM stops new acceptance and drains queued/active work under the host policy. Active handlers are not killed mid-statement; long-running work can extend shutdown. Set finite network budgets and make shutdown handlers small. A ready file is removed during normal shutdown.

Source basis: [K04], [K12], [V1].

## 10 • BACKEND SERVICES

## Run the supplied two-service gateway

The complete source includes examples/kokum\_029\_url\_services. Its ordinary kokum.http.tasks module exports GetAsync and PostAsync. These are source-library functions, not new built-ins. Use its exact manifest to link the imports.

**BUILD BOTH APPLICATIONS**

```
cd "$KOKUM_HOME/examples/kokum_029_url_services"
"$KOKUM_HOME/kokumc" bundle upstream.toml
"$KOKUM_HOME/kokumc" bundle kokum.toml
```

**TERMINAL 1 — STAY IN THE EXAMPLE DIRECTORY**

```
"$KOKUM_HOME/kokumvm" serve dist/upstream.kapp \
  --config upstream.conf
```

**TERMINAL 2 — SAME DIRECTORY**

```
KOKUM_TASK_WORKERS=4 KOKUM_URL_MAX_CONCURRENT=3 \
  "$KOKUM_HOME/kokumvm" serve dist/gateway.kapp \
  --config gateway.conf
```

**TERMINAL 3**

```
curl http://127.0.0.1:8080/summary
curl -H "Content-Type: application/json" \
  --data '{"sku":"KOKUM-BOOK","quantity":2}' \
  http://127.0.0.1:8080/orders
```

The gateway reads upstream.json from its working directory once in each worker startup. Copy that operator-controlled file and gateway.conf alongside a deployed bundle and restart after changes. It is not automatically embedded in the application.

The gateway fans out two GETs, forwards a JSON POST exactly once, applies explicit time/size limits, disables redirects and does not forward incoming authorization/cookie/API-key headers. It returns generic 502/504 errors with a server-generated request ID.

**DEMONSTRATION ONLY** The upstream order endpoint echoes JSON; it does not persist orders, place trades or take payments. This is not service discovery, a circuit breaker, a retry engine or a public authenticated proxy.

Source basis: [K11], [K12].

## 11 • BROWSER BRIDGES AND DEPLOYMENT

# Keep the UI transport separate from application networking

A Web GUI uses a tokenized loopback host and a form bridge. Browser events are validated, dispatched to exported Kokum slots and converted into patches. An outgoing URL request or WebSocket feed is a different connection owned by KokumVM.

| Connection                | Purpose                                                | Boundary                                                |
|---------------------------|--------------------------------------------------------|---------------------------------------------------------|
| Browser → local Web GUI   | Form initialization, events and patches                | Loopback host, launch token, exact Host/Origin checks   |
| Browser → LAN IDE         | Authenticated editing and permitted project operations | Private interface, account/session, contained workspace |
| KokumVM → remote API/feed | URLGET/URLPOST or WebSocket client                     | Explicit endpoint, TLS/auth policy and limits           |
| KokumVM → database        | Remote FlatDB or DATABASE provider                     | Provider configuration and owner-session connection     |

## Application event flow

### CONCEPTUAL FLOW — NOT A GENERAL REMOTE-CALL API

```
browser event
  -> validated form bridge
  -> owning Kokum slot
  -> task or database operation
  -> owner-session result handling
  -> selected UI patch
```

The frontend is not granted arbitrary native filesystem or network access by the bridge. Database configuration is kept outside served resources; WebSocket native configuration and authentication are not exposed as ordinary browser component state. Application code can still leak a secret by explicitly inserting it into a label, returned JSON or log.

## Launch a packaged Web GUI locally

### OPEN THE COMPLETE TOKENIZED URL THAT IS PRINTED

```
kokumvm web dist/application.kapp --no-browser
```

Do not weaken token/origin checks or reuse the IDE token for an application. The local Web GUI host is not converted into a remote multi-user application server by changing a URL in the browser.

Source basis: [K07], [K08], [K13].

11 • BROWSER BRIDGES AND DEPLOYMENT

# Move code and external configuration together

| File or setting           | Where it belongs                                                                           |
|---------------------------|--------------------------------------------------------------------------------------------|
| .kbc / .kapp              | Application deployment directory; run with matching KokumVM.                               |
| Standalone executable     | Built for the intended host platform; embeds its runtime.                                  |
| kdb.conf + matching .kkey | Beside the FlatDB client application; relative key path follows the selected config.       |
| db_env.conf               | Private external provider configuration, or explicit KOKUM_DB_ENV path.                    |
| server.conf               | Beside the backend application, or an explicit config path.                                |
| upstream.json             | The gateway's working directory in the supplied demo.                                      |
| CA files / client keys    | Native-only private configuration; use deliberate absolute or supported substituted paths. |

## Windows and macOS clients

Build the application on Linux, copy its portable bundle or bytecode and necessary external files, then run it with the target KokumVM. Do not copy a Linux standalone binary and expect Windows or macOS to execute it. Compiler and IDE binaries remain Linux-only in this line.

### WINDOWS POWERSHELL EXAMPLES

```
.\kokumvm.exe .\application.kapp
Test-NetConnection 192.168.10.20 -Port 9437
Get-FileHash .\manual.kkey -Algorithm SHA256
```

## Compatibility and capabilities

URLGET/URLPOST preserve built-in IDs 105/106 and URL ABI 1.0. Network-connect capability metadata survives compilation and bundling; metadata alone is not an SSRF/destination sandbox. Existing bytecode/bundle formats were retained through Phase 4.17.7.

**VALIDATION SCOPE** The final HTTP release includes portable fixtures and native CI entry points. Linux-host Windows/macOS adapter simulations are not native operating-system certification. Test the actual target VM, trust files, DNS and deployment layout.

Source basis: [K04], [K07], [K11], [K14].

## 12 • SECURITY AND OPERATIONS

# Use the right security boundary for each facility

| Facility             | Protection present                                                     | Responsibility still yours                                             |
|----------------------|------------------------------------------------------------------------|------------------------------------------------------------------------|
| URL HTTPS            | Chain/identity verification by default; redirect credential protection | Approved destinations, secret handling, response schema, authorization |
| WebSocket WSS        | TLS trust/identity; bounded native queues                              | Select WSS for secrets, service login/subscription, reconnect recovery |
| Remote FlatDB        | Shared-key authentication and encrypted transport                      | Key distribution, equal-access model, unencrypted database/backups     |
| TigerDB / PostgreSQL | Configured provider TLS and server login                               | Correct trust mode, privileges, transaction/retry decisions            |
| LAN IDE              | Private binding, accounts/sessions and workspaces                      | Trusted LAN/users; plaintext password file and HTTP restrictions       |
| Backend service      | Bounded HTTP parsing, routes, worker ownership                         | Application authentication, public TLS edge, destination/rate policy   |

## Use data, not authority, across a boundary

Never forward an arbitrary browser header MAP to an upstream. Choose each outbound header explicitly. Treat URLs, SQL identifiers, route parameters and filesystem paths as separate input types requiring their own validation. Keep secrets out of queries, browser resources and raw error responses.

## Do not confuse availability with identity

A listening TCP port does not prove that the expected service is there. A TLS session without verified identity is not proof of the intended peer. A health endpoint does not prove that a dependent database transaction succeeded. Log these layers separately.

**PUBLIC NETWORKS** Only expose an application backend after supplying the missing production controls. The network IDE remains prohibited from public exposure in this release, even if another component uses a public-facing proxy.

Source basis: [K03]–[K13], [R5], [R8].

12 • SECURITY AND OPERATIONS

# Operate within bounded resources

## Set budgets from the application outward

Choose a user-visible operation deadline, a bounded number of upstream calls, finite body/response sizes, worker/queue limits and a shutdown policy. The total request path can include task queue waiting, HTTP admission, DNS, TLS, I/O, JSON conversion and database work. No single TimeoutMs setting covers all those layers preemptively.

| Observe                                   | Why it matters                                                          |
|-------------------------------------------|-------------------------------------------------------------------------|
| Status and failure code                   | Separate HTTP error replies from transport/TLS/admission failures.      |
| ElapsedMs and operation wall time         | Reveal queue/preparation time not captured by one transport field.      |
| Task queue and HTTP admission             | Show whether work is waiting rather than executing.                     |
| WebSocket queue/dropped counters          | Reveal an undersized or slow message consumer.                          |
| Database latency and connection state     | Distinguish server pressure, stale sessions and application SQL issues. |
| Backend health, request IDs and readiness | Support startup checks, correlation and clean shutdown.                 |

## Logging

Keep public error messages generic; keep necessary diagnostic codes in a trusted log. Avoid serializing entire options/request objects containing credentials. Backend trace can include request targets, so leave it off when query values may be sensitive. A WebSocket Snapshot is safer for counters than an application’s arbitrary header MAP.

## Performance expectations

The HTTP client opens request-local connections; it does not provide a persistent connection pool. Concurrent I/O can improve throughput when waiting dominates, but additional workers consume resources and can overload an upstream. Measure a representative workload instead of extrapolating loopback benchmark numbers.

## Repeatable verification

STRICT ADDS REQUIRED BROWSER TESTING

```
cd "$KOKUM_HOME"
make -j1 phase4-17-7-verify
make -j1 phase4-17-7-strict-verify
```

Source basis: [K08], [K09], [K11], [K12], [K14].

## 12 • SECURITY AND OPERATIONS

## Troubleshoot HTTP and TLS in layers

| Symptom                     | First checks / next action                                                                                 |
|-----------------------------|------------------------------------------------------------------------------------------------------------|
| Unknown HTTPGET/HTTPPOST    | Use URLGET/URLPOST. Confirm the running compiler/VM belongs to Phase 4.17.7.                               |
| KUR1008 before I/O          | Check option types, BodyType, raw JSON, header conflicts and mutually exclusive auth inputs.               |
| DNS / connect failure       | Resolve the intended host on the machine running KokumVM; confirm the port and listener.                   |
| KUR3001                     | The trust file/store could not be loaded. Check path, readability and PEM format.                          |
| KUR3005                     | Certificate chain or identity mismatch. Correct CA, SAN/name or TLSServerName; do not bypass verification. |
| 404/401/500 with Ok=.F.     | A response arrived. Check route, application authentication and server-side semantics.                     |
| Json=NULL                   | Inspect JsonError and ContentType; valid JSON null differs from malformed claimed JSON.                    |
| KUR6004 / KUR6005           | Downgrade or cross-origin POST replay was blocked. Review the destination chain and body sensitivity.      |
| KUR7001                     | HTTP admission expired before sending. Bound fan-out, inspect capacity and operation budgets.              |
| Repeated POST after timeout | Stop blind retries. Reconcile with the server because the first write may already have executed.           |

### Minimal isolation recipe

First run the local /hello lab. Then test the failing host with one plain request or verified HTTPS request using a tiny response limit and no custom headers. Add query, body, authentication and redirects individually. Preserve the first error and the exact runtime version.

**PRIVATE DATA** Do not paste a real token, cookie, password or private key into a bug report. Include a redacted configuration, the diagnostic code, expected behavior and a minimal reproducible endpoint description.

Source basis: [K09], [K10], [K11].

## 12 • SECURITY AND OPERATIONS

## Troubleshoot databases, WebSockets and the IDE

| Symptom                                    | What to check                                                                                                                                |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| FlatDB KCONNECTED() is false               | TCP reachability; exact database filename; exact key basename and binary hash; app-relative kdb.conf. Remote denial is intentionally silent. |
| FlatDB second-argument error               | KEXECUTE/KQUERY accept one SQL STRING. Do not pass a parameter array.                                                                        |
| Works in IDE, fails after deployment       | Copy external config/key/CA files and check runtime lookup; distinguish app directory from working directory.                                |
| DATABASE or WEBSOCKET is NULL              | The component was not instantiated/bound. A declaration alone is not a constructor; check Designer ID/module/project build.                  |
| PostgreSQL TLS unexpectedly optional       | Replace an unintended sslmode=prefer policy with verified settings and the correct server trust configuration.                               |
| WebSocket not connected                    | Start the endpoint first, inspect LastError, URL, TLS, subprotocol and service authentication. Confirm Connect completed.                    |
| Feed connects but no data                  | Check subscription acknowledgement, event type, queue depth and that the Timer/Receive consumer is running.                                  |
| LAN IDE rejects bind config                | Use one assigned private IPv4 address and a valid private CIDR; no wildcard/public listener.                                                 |
| LAN Run cannot reach laptop service        | The application runs on the Linux server. Its 127.0.0.1 is not the client laptop.                                                            |
| Web GUI loopback URL fails from LAN client | Remote preview routing is not implemented. This does not imply the app's outgoing network request failed.                                    |

### Safe recovery

Reconnect a broken read session with fresh owner-local state. Do not automatically replay writes. Close or drain outstanding work deliberately, check file permissions and transport limits, and verify the correct artifact is being run before changing server policy.

Source basis: [K03]–[K08], [K11].

## APPENDIX A • HTTP REFERENCE

# Request options: headers, queries and budgets

Option names are case-insensitive. Values must have the documented runtime type. These are options for URLGET/URLPOST, not for WEBSOCKET components or server.conf.

| Option                     | Default | Contract                                                                                               |
|----------------------------|---------|--------------------------------------------------------------------------------------------------------|
| Headers / MAP              | {}      | Validated caller headers. Host, Content-Length, Transfer-Encoding and Connection cannot be overridden. |
| Query / MAP                | {}      | Percent-encode scalars; arrays repeat a parameter. Apply only to the initial URL.                      |
| ConnectTimeoutMs / INTEGER | 5000    | TCP/TLS budget constrained by total time. Blocking DNS is not interruptible.                           |
| TimeoutMs / INTEGER        | 30000   | Admission + transport/redirect budget after preparation; excludes earlier task queue waiting.          |
| ReadTimeoutMs / INTEGER    | 30000   | Read inactivity budget, also constrained by remaining total time.                                      |
| MaxResponseBytes / INTEGER | 8388608 | Maximum retained response body; select a smaller value for bounded APIs.                               |
| FollowRedirects / LOGICAL  | .T.     | Follow supported redirects with method and origin protection.                                          |
| MaxRedirects / INTEGER     | 5       | 0..20 followed transitions, not including the initial request.                                         |
| VerifyTLS / LOGICAL        | .T.     | Verify certificate chain and server identity.                                                          |
| CAFile / STRING            | empty   | Optional PEM trust file for HTTPS.                                                                     |

## Budget selection example

### OPTIONS MAP FRAGMENT

```
{ "TimeoutMs": 5000, "ConnectTimeoutMs": 2000, ;
  "ReadTimeoutMs": 2000, "MaxResponseBytes": 262144 }
```

Keep positive finite timeout values. A smaller connect/read budget never extends the total budget. Intermediate redirect responses also pass through bounded parsing; the final response is not the only point at which a size/protocol failure can occur.

Source basis: [K09], [K10], [K11].

## APPENDIX A • HTTP REFERENCE

## Request options: bodies, credentials and redirect policy

| Option                         | Default      | Contract                                                                                                |
|--------------------------------|--------------|---------------------------------------------------------------------------------------------------------|
| TLSServerName / STRING         | empty        | Optional SNI/identity override; cleared on cross-origin redirect.                                       |
| ContentType / STRING           | automatic    | Explicit POST media type; conflicts with Headers["Content-Type"].                                       |
| BodyType / STRING              | automatic    | text, json or form; match the documented input shape.                                                   |
| BearerToken / STRING           | empty        | Nonempty validated token generates Authorization. Does not acquire/refresh tokens.                      |
| BasicAuth / MAP                | NULL         | Username and Password STRING values; do not combine with another Authorization source.                  |
| Cookie / STRING                | empty        | Explicit request cookie only; conflicts with Headers["Cookie"].                                         |
| UserAgent / STRING             | Kokum/0.29.0 | Replaces the default User-Agent.                                                                        |
| CookiePolicy / STRING          | manual       | manual or omit. No automatic cookie jar.                                                                |
| AllowInsecureAuth / LOGICAL    | .F.          | Opt in to HTTP transmission for Basic/Bearer helper tests only.                                         |
| AllowCrossOriginBody / LOGICAL | .F.          | Permit explicitly approved cross-origin POST replay; never a downgrade or credential-header forwarding. |

## Body-selection rules

Automatic STRING is text, MAP/ARRAY is JSON and NULL is empty. Explicit text requires STRING/NULL; json accepts STRING/MAP/ARRAY/NULL; form accepts MAP/NULL. Top-level arbitrary numbers or booleans are not automatic body modes. The native prepared-body limit is 64 MiB.

## No hidden session state

These options configure one call. There is no persistent authenticated HTTP session object, cookie store, token refresh, retry engine or connection pool behind URLGET/URLPOST. Compose any application policy explicitly and preserve the existing security boundaries.

Source basis: [K09], [K10].

APPENDIX A • HTTP REFERENCE

# The fourteen response fields

| Field         | Meaning                                                                              |
|---------------|--------------------------------------------------------------------------------------|
| Ok            | LOGICAL true for HTTP status 200..299.                                               |
| Status        | INTEGER final HTTP status code.                                                      |
| Reason        | Reason phrase from the final status line.                                            |
| Url           | Final canonical URL after followed redirects.                                        |
| HttpVersion   | HTTP/1.0 or HTTP/1.1.                                                                |
| Headers       | Convenient MAP with lower-case header names.                                         |
| HeaderValues  | Lower-case header names → ARRAYs preserving duplicates.                              |
| Body          | Retained response body as a Kokum STRING.                                            |
| Json          | Decoded JSON value, or NULL when unavailable/invalid.                                |
| JsonError     | Empty on successful/no failed decoding; text when claimed JSON could not be decoded. |
| ContentType   | Response Content-Type or an empty string.                                            |
| ContentLength | Actual retained body bytes; not merely the declared Content-Length header.           |
| RedirectCount | Number of followed redirect transitions.                                             |
| ElapsedMs     | Admission/transport chain duration, including connections and TLS.                   |

## Header lookup example

FRAGMENT — READ DUPLICATES WITHOUT COMMA-FOLDING

```
IF HASKEY(response["HeaderValues"], "set-cookie")
    cookies = response["HeaderValues"]["set-cookie"]
ENDIF
```

Response keys use the exact case shown. Header-name keys are lower-case. Json is decoded automatically for application/json and application/\*+json content types. Body remains available when claimed JSON is invalid. The API does not claim binary-download or decompression support.

Source basis: [K09].

## APPENDIX A • HTTP REFERENCE

## Diagnostics: validation, network and TLS

| Code    | Meaning / interpretation                                                      |
|---------|-------------------------------------------------------------------------------|
| KUR1001 | Invalid argument or incompatible runtime type.                                |
| KUR1008 | Invalid option, body mode, raw JSON, header conflict or ambiguous auth input. |
| KUR1900 | Allocation or internal preparation failure.                                   |
| KUR2001 | DNS resolution failed.                                                        |
| KUR2002 | TCP connection failed.                                                        |
| KUR2003 | Plain HTTP request write failed.                                              |
| KUR2004 | Plain HTTP response read failed.                                              |
| KUR2005 | Connection phase timed out.                                                   |
| KUR2006 | Response read timed out.                                                      |
| KUR2007 | Total HTTP request budget expired.                                            |
| KUR3001 | HTTPS trust store or CA file could not be loaded.                             |
| KUR3002 | TLS context, SNI, identity or ALPN setup failed.                              |
| KUR3003 | TLS handshake or selected protocol failed.                                    |
| KUR3004 | TLS handshake timed out.                                                      |
| KUR3005 | Certificate chain or server identity verification failed.                     |
| KUR3006 | HTTPS response read failed.                                                   |
| KUR3007 | HTTPS request write failed.                                                   |

Handle these inside TRY/CATCH. They represent a failed native operation, not an ordinary HTTP status. Keep error detail in trusted diagnostics; do not expose remote URLs, secrets or internal system details in a public API response.

Source basis: [K09], [K10], [K11].

## APPENDIX A • HTTP REFERENCE

## Diagnostics: framing, limits, redirects and admission

| Code    | Meaning / interpretation                                               |
|---------|------------------------------------------------------------------------|
| KUR4001 | Malformed status line.                                                 |
| KUR4002 | Malformed response header.                                             |
| KUR4003 | Response-header limit exceeded.                                        |
| KUR4004 | Invalid or unsupported response framing.                               |
| KUR4005 | Response ended before the declared/framed body completed.              |
| KUR5001 | Response body exceeds MaxResponseBytes.                                |
| KUR5002 | Prepared POST body exceeds 64 MiB.                                     |
| KUR6001 | Redirect transition limit exceeded.                                    |
| KUR6002 | Invalid, unsafe, unsupported or ambiguous redirect Location.           |
| KUR6003 | Redirect loop detected.                                                |
| KUR6004 | HTTPS-to-HTTP redirect blocked.                                        |
| KUR6005 | Cross-origin POST replay blocked without explicit body approval.       |
| KUR6006 | Authentication helper requires HTTPS or explicit insecure-test opt-in. |
| KUR7001 | Process-wide HTTP admission timed out before transmission.             |

### Know what has already happened

KUR7001 indicates no HTTP request was sent. Validation failures occur before transport. By contrast, a response read failure or timeout after a POST can leave an unknown server-side outcome. The absence of a response is not evidence of rollback.

### Malformed replies now rejected

Phase 4.17.7 rejects embedded NUL in response field names, empty/trailing Transfer-Encoding items, Transfer-Encoding on HTTP/1.0, invalid status-code ranges and prohibited chunk trailers. Do not treat older acceptance of those replies as supported behavior.

Source basis: [K09], [K10], [K14].

## APPENDIX B • WEBSOCKET REFERENCE

# Configuration and message-control methods

| Method family           | Signature / role                                                                                                     |
|-------------------------|----------------------------------------------------------------------------------------------------------------------|
| Endpoint                | SetUrl(url); Connect([url])                                                                                          |
| Subprotocols            | SetProtocols(text) / SetSubprotocols(text)                                                                           |
| Headers                 | SetHeader(name,value); RemoveHeader(name); ClearHeaders()                                                            |
| Bearer / feed / API key | SetBearerToken(token[,header]); SetFeedToken(token[,header]); SetApiKey(value[,header])                              |
| Basic / Cookie          | SetBasicAuth(user,password); SetCookie(value)                                                                        |
| Query                   | SetQueryParameter(name,value) / AddQueryParameter(name,value);<br>RemoveQueryParameter(name); ClearQueryParameters() |
| Reconnect               | SetReconnect(enabled[,delayMs[,maxAttempts]])                                                                        |
| Sending                 | SendText(text); SendJSON(value); SendBinary(array); Ping([payload])                                                  |
| Receiving               | Receive/Poll([timeoutMs]); ReceiveText([timeoutMs]); ReceiveJSON([timeoutMs])                                        |
| Queue handling          | Wait([timeoutMs]); Drain([maximumCount]); ClearMessages()                                                            |
| Status / close          | Snapshot(); Close([code[,reason[,waitMs]]]); Disconnect with normal-close semantics                                  |

Change endpoint, headers, authentication and related connection configuration while disconnected. Read-only status properties include Id, Url, State, IsConnected/Connected, IsRunning, PendingMessages, PendingSends, SentMessages, SentBytes, ReceivedMessages, ReceivedBytes, DroppedMessages, ReconnectCount, Protocol, LastError, LastMessage, LastCloseCode, LastCloseReason and Reconnect.

Wait tests whether events become available; use Receive/Drain to consume them. A returned snapshot is portable status data, not a live controller. Keep close reasons and control payloads small and valid for the protocol.

Source basis: [K08], src/kokum\_websocket.c method dispatcher.

## APPENDIX C • VALIDATION

# What was checked for this manual

| Example family                                  | Validation performed for this book                                                                              |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| GET / POST / JSON / forms / XML / empty bodies  | Executed against the new local Kokum lab service.                                                               |
| HTTP errors, invalid claimed JSON and redirects | Executed locally; status, body/decode and follow/no-follow behavior observed.                                   |
| HTTPS / Basic / Bearer / cookie omit            | Executed against the disposable local TLS helper with verification enabled.                                     |
| ASYNC / named RUN / task wait timeout           | Executed through VM-backed bytecode; same-task completion checked.                                              |
| Remote FlatDB CRUD / async read                 | Executed against a freshly created local FlatDB server and matching shared key.                                 |
| FlatDB-backed service                           | Compiled, served by KokumVM and queried over HTTP.                                                              |
| WebSocket GUI echo                              | Project checked/bundled; real native form-bridge connect/send/drain/close interactions against websockets 16.0. |
| HTTP GUI polling                                | Project checked/bundled; real native form-bridge start/timer/result interaction.                                |
| TigerDB / PostgreSQL examples                   | Semantic checks plus native provider regression fixtures; not live application-database execution.              |
| LAN configuration / platform guidance           | Current source/configuration review; no physical multi-user LAN or native Windows/macOS run for this book.      |

The console GET, JSON POST and FlatDB CRUD examples were additionally exercised via AST, IR and separately compiled KBC execution. This is a manual example-validation run, not a new full release certification.

**GUI TEST BOUNDARY** Native form-bridge tests exercise real runtime callbacks and patches but are not a visual, human-driven Designer test on every desktop platform. Use the specified component IDs and bindings.

Source basis: [V1]; source baseline [K01].

## APPENDIX D • TLS LAB FIXTURE

# A disposable HTTPS endpoint for the examples

## tls\_lab.py — PYTHON 3 LOCAL TEST FIXTURE

```

import json
import ssl
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer

class Echo(BaseHTTPRequestHandler):
    def do_GET(self):
        self.reply(None)

    def do_POST(self):
        size = int(self.headers.get("Content-Length", "0"))
        if size > 65536:
            self.send_error(413)
            return
        self.reply(json.loads(self.rfile.read(size) or b"null"))

    def reply(self, body):
        data = json.dumps({"received": body,
            "hasAuth": "Authorization" in self.headers,
            "hasCookie": "Cookie" in self.headers}).encode()
        self.send_response(200)
        self.send_header("Content-Type", "application/json")
        self.send_header("Content-Length", str(len(data)))
        self.end_headers()
        self.wfile.write(data)

context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.minimum_version = ssl.TLSVersion.TLSv1_2
context.load_cert_chain("lab-cert.pem", "lab-key.pem")
server = ThreadingHTTPServer(("127.0.0.1", 18443), Echo)
server.socket = context.wrap_socket(server.socket, server_side=True)
server.serve_forever()

```

Run in the directory containing lab-cert.pem and lab-key.pem created in Chapter 7. It binds only loopback, accepts the book's small valid JSON requests and reports header presence rather than values. Stop it with Ctrl+C. It is deliberately not a production server, general upload handler or authentication service.

Both hasAuth and hasCookie indicate only whether a header was present. They do not validate the credentials. This helper is outside Kokum and exists solely to test verified HTTPS and per-call header construction reproducibly. [R7]

Source basis: [V1], [R7].

## APPENDIX E • SOURCES

# Source map and version authority

The current implementation and tests take precedence over historical introductory documentation. Paths below are relative to the Phase 4.17.7 source archive. References in chapter footnotes use these identifiers.

| ID  | Authoritative source area                                                                                                  |
|-----|----------------------------------------------------------------------------------------------------------------------------|
| K01 | DEVELOPMENT_LINE; PHASE4_17_7_RELEASE_NOTES.md; final source archive and verification report.                              |
| K02 | src/tlang_builtin_catalog.c — built-in names and arities.                                                                  |
| K03 | config/kokum-ide-lan.conf.example; docs/LAN_PROJECT_OPERATIONS.md; docs/LAN_CSV_AUTHENTICATION.md; src/kokum_ide_server.c. |
| K04 | flatdb/README.md; src/kokum_flatdb.c; flatdb/kokum_flatdb_server.c; docs/PHASE4_13_2_REMOTE_FLATDB_CONNECTIVITY_FIX.md.    |
| K05 | docs/TIGERDB_PROVIDER.md; src/kokum_tigerdb_client.c.                                                                      |
| K06 | docs/POSTGRESQL_PROVIDER.md; src/kokum_postgresql_client.c; tests/postgresql_api_test.c.                                   |
| K07 | docs/DB_ENV_CONF.md; docs/CREATE_TABLE_FROM_JSON.md; component creation and bundle/resource handling.                      |
| K08 | docs/WEBSOCKET_CLIENT.md; include/kokum_websocket.h; src/kokum_websocket.c; tests/run_kokum_027_websocket_*.               |
| K09 | docs/URL_HTTP_CLIENT.md; include/kokum_url.h; src/kokum_url.c.                                                             |
| K10 | docs/URL_REDIRECT_SECURITY.md; redirect/auth/cookie tests in tests/phase4_17_5.                                            |
| K11 | docs/URL_ASYNC_MICROSERVICES_IDE.md; docs/ASYNC_TASKS.md; src/kokum_task.c; tests/phase4_17_6.                             |
| K12 | docs/PHASE4_15_0_BACKEND_SERVICES_FOUNDATION.md; src/kokum_backend.c; examples/kokum_029_url_services.                     |
| K13 | docs/WEB_HOST_SECURITY.md; docs/IDE_SECURITY.md; docs/WEB_FORM_BRIDGE_PROTOCOL.md; src/tlang_web_form.c.                   |
| K14 | docs/URL_VM_PORTABILITY_FUZZ_PERFORMANCE.md; tests/phase4_17_7; final HTTP verification report.                            |
| V1  | New book-specific examples, compiler checks and local runtime/fixture tests performed on Linux for this edition.           |

## APPENDIX E • SOURCES

## Protocol references and release limits

These primary references explain the underlying protocols and external lib dependency. They do not expand Kokum's implemented feature set. Accessed for this edition on 9 September 2026.

**R1 HTTP semantics and redirects**

<https://www.rfc-editor.org/rfc/rfc9110.html>

**R2 HTTP/1.1 message framing**

<https://www.rfc-editor.org/rfc/rfc9112.html>

**R3 URI reference resolution**

<https://www.rfc-editor.org/rfc/rfc3986.html>

**R4 The WebSocket Protocol**

<https://www.rfc-editor.org/rfc/rfc6455.html>

**R5 PostgreSQL libpq TLS modes**

<https://www.postgresql.org/docs/current/libpq-ssl.html>

**R6 websockets 16.0 threading server API**

<https://websockets.readthedocs.io/en/16.0/reference/sync/server.html>

**R7 Python ssl: TLS wrapper**

<https://docs.python.org/3/library/ssl.html>

**R8 OpenSSL 3.5 threading guidance**

<https://docs.openssl.org/3.5/man7/openssl-threads/>

### What this edition deliberately does not promise

Native Windows/macOS certification; an Internet-ready network IDE; an automatic cookie jar; HTTP connection pooling, decompression, streaming or multipart; additional URL client verbs; a cancellation or retry engine; a public WebSocket server; implicit credential forwarding; or live external database access without operator configuration.

### Baseline fingerprint

**SHA-256 OF THE SOURCE ZIP USED FOR THIS BOOK**

```
d057a629b0c9274a3558e8894eae448d
f284ec56de9bb54318ab6d864de168bb
```

This fingerprint identifies the source used to prepare the manual, not the DOCX itself. All examples and interface names refer to that baseline. Keep a copy of the release-specific documentation when upgrading.