KodeMed

System Architecture

For CTOs, IT Administration & DevOps

Version 2026.5.15.59502 | KodeMed AG

System Overview

KodeMed platform components

KodeMed.Server

Java backend (REST + WebSocket)

KodeMed.DataServer

Classification data API

KodeMed.GrouperServer

DRG grouping API (SwissDRG, TARPSY, ST Reha)

KodeMed.CodingUI

React/Vite web frontend

KodeMed.Interface

COM DLL (.NET 9.0)

KodeMed.CodingClient

System tray app – portable EXE
WebSocket, webhook, language selector

```
Flow:  HIS/3rd Party  →  Server (REST/WS)  →  CodingClient (WS)  →  DLL  →  CodingUI (WebView2)

Flow:  HIS (direct)  →  DLL (COM)  →  Server  →  CodingUI (WebView2)
```

HIS Integration

Hospital Information System integration via COM DLL

DLL API (IKodeMed)

Method	Description
SendConfig(xml)	Load configuration
DoCodingWithFormat(data, format)	Full UI flow (auth + browser)
DoCoding(data, format)	Headless processing
GetResults()	Retrieve coding results
GetResultData()	Get clean SpiGes XML
DisposeClient()	Release resources

Integration Example (C#)

```
// 1. Create instance
var coding = new Coding();

// 2. Configure
coding.SendConfig(configXml);

// 3. Process with UI
coding.DoCodingWithFormat(
  spigesData, FormatType.SpiGes);

// 4. Check action
if (coding.LastCodingAction
    == CodingAction.Applied)
{
  var results = coding.GetResults();
  var xml = coding.GetResultData();
}

// 5. Cleanup
coding.DisposeClient();
```

Client Installation

Three deployment methods

MSI Installer (recommended)

- KodeMed.msi package
- Per-user (no admin) or per-machine (Citrix)
- Silent install via GPO/SCCM/Intune
- COM registration + config included
- WebView2 runtime required

Manual / Xcopy

- Copy DLL + deps
- Manual COM registration
- Set env vars manually
- For advanced users

CodingClient

- Installed via MSI
- System tray + WebSocket
- Auto-reconnect + webhook
- Citrix/VDI compatible
- WebView2-based UI

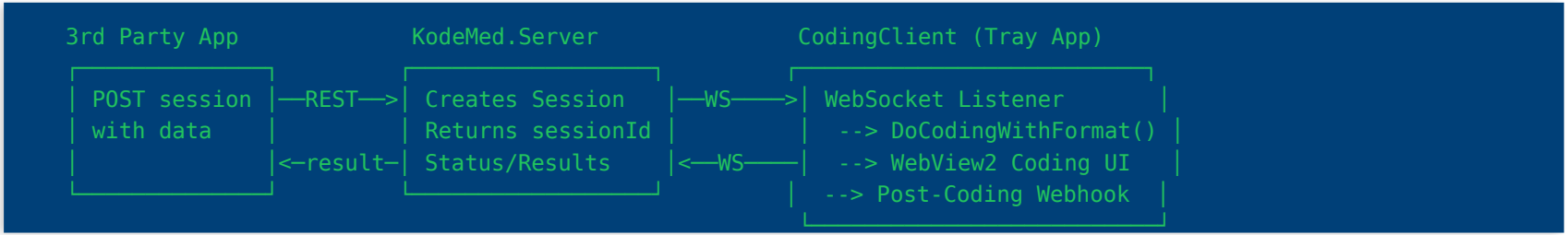
Configuration (set automatically by MSI installer or on first launch)

Variable	Description	Example
KODEMED_HOME	Installation directory	C:\Users\joe\KodeMed\
KODEMED_DLL_PATH	Full path to KodeMed.dll	C:\Users\joe\KodeMed\KodeMed.dll
KODEMED_EXE_PATH	Path to CodingClient.exe	C:\Users\joe\KodeMed\KodeMed.CodingClient.exe

CodingClient Architecture

System tray application for 3rd-party integration

Integration Flow



Features

- Single-instance (global mutex)
- WebSocket listener for CODING_SESSION_LAUNCH
- Post-coding webhook (fire-and-forget, retry)
- Language selector (DE/FR/IT/EN, tray menu)
- OAuth2 sign in/out via tray menu
- Auto-start on Windows login (HKCU\Run)
- Open sessions from local files

Citrix / VDI Highlights

- Self-contained: no .NET runtime dependency
- All writes user-level (HKCU, env vars)
- Portable: copy folder and run
- COM via HKCU\Software\Classes (no admin)
- Re-setup on non-persistent VDI login

Server Architecture

Java backend with REST API and WebSocket

KodeMed.Server (Java)

- Spring Boot application
- REST API: config, session, audit, instances, layouts, health
- WebSocket endpoint (/ws/dll, /ws/app)
- JPA/Hibernate persistence (H2/PostgreSQL)
- Audit trail (session events, user actions)
- Layout management (save/load/share)
- OAuth2 token validation
- Grouper integration (SwissDRG)
- Docker deployment

KodeMed.DataServer

- Classification data API
- ICD-10-GM codes + descriptions
- CHOP procedure codes
- ATC medication codes
- SwissDRG grouper catalog
- Full-text search with autocomplete
- Versioned catalogs (yearly updates)
- REST API with caching

Coding UI

React/Vite web frontend

KodeMed.CodingUI (React + Vite)

- React 18 with TypeScript + Vite
- Zustand state management
- 14+ data blocks (draggable, react-grid-layout)
- Editable code tables (ICD, CHOP, ATC)
- Code search with autocomplete
- SpiGes attribute editing per row
- Administrative data forms
- Real-time grouper results (SwissDRG)
- Multi-language UI (DE/FR/IT/EN)
- Undo history bar (visual step-through)
- Embedded in DLL via WebView2
- Standalone browser mode (OAuth2/OIDC)
- WebSocket live updates
- Save/load/share custom layouts
- Responsive design + dark/light theme
- Keyboard navigation
- Export/import functionality

Authentication & Security

OAuth2 / OpenID Connect

OAuth2 / OIDC

- Authorization Code + PKCE flow
- Keycloak, Auth0, Azure AD compatible
- DLL: embedded browser login
- UI: standard OIDC redirect
- Token auto-refresh
- Multi-tenant via realms
- Group-based access control
- OpenID Connect user claims

Security Measures

- HTTPS everywhere (TLS 1.2+)
- WSS for WebSocket connections
- JWT token validation on server
- CORS policy enforcement
- Session expiry (configurable)
- Audit logging (all session events)

GDPR / nDSG Compliance

- Data minimization defaults (webhook opt-in)
- includeResultData/includeGrouperResults: off
- SSRF protection on webhook URLs
- No secrets in client config

Data Model

SpiGes XML structure (BFS / Swiss Federal Statistics)

```
Unternehmen (Enterprise)
├── ent_id, version
├── Standort[] (Sites)
│   ├── burnr (BUR number)
│   ├── Fall[] (Cases)
│   │   ├── Administratives
│   │   │   (demographics, dates,
│   │   │    insurance, canton...)
│   │   ├── Diagnose[]
│   │   │   (ICD-10 codes, POA)
│   │   ├── Behandlung[]
│   │   │   (CHOP procedures)
│   │   ├── Medikament[]
│   │   │   (ATC medications)
│   │   ├── Rechnung[] (invoices)
│   │   └── Patientenbewegung[]
│   └── KostentraegerStandort[]
└── KostentraegerUnternehmen[]
```

Key Entities

Entity	Key Fields
Administratives	abc_fall, alter, geschlecht, ein/austritt
Diagnose	diagnose_kode (ICD-10), diagnose_poa
Behandlung	behandlung_chop (CHOP), behandlung_beginn
Medikament	medi_atc, medi_dosis, medi_einheit
GrouperResult	DRG, MDC, PCCL, cost weight

Post-Coding Webhook

Fire-and-forget HTTP callback after coding sessions

How It Works

- Coder completes session (Apply/Discard/...)
- CodingClient POSTs JSON to configured URL
- Async: does not block the coder
- Retry with exponential backoff (1s→2s→4s...60s)
- No retry on 4xx (except 429)

Configuration (env vars)

Variable	Default
KODEMED_HOOK_ENABLED	false
KODEMED_HOOK_URL	(empty)
KODEMED_HOOK_INCLUDE_RESULT_DATA	false
KODEMED_HOOK_INCLUDE_GROUPER_RESULTS	false
KODEMED_HOOK_EVENTS	applied

Webhook Payload

```
POST https://his.example.com/api/results
Content-Type: application/json

{
  "eventType": "coding_session_completed",
  "sessionId": "sess_abc123",
  "codingAction": "Applied",
  "applied": true,
  "username": "dr.muller",
  "sourceFormat": "SpiGes",
  "casesProcessed": 3,
  "diagnosesCount": 12,
  "treatmentsCount": 7,
  "resultData": "...(opt-in)...",
  "grouperResults": [...(opt-in)...]
}
```

Auth: bearer token or custom header (local config only)

Deployment

Docker, on-premise, cloud

Server Components

```
# Docker Compose
services:
  kodemed-server:
    image: kodemed/server:latest
    ports: ["8080:8080"]

  kodemed-dataserver:
    image: kodemed/dataserver:latest
    ports: ["8081:8081"]

  kodemed-grouper-server:
    image: kodemed/grouper-server:latest
    ports: ["8082:8082"]

  kodemed-ui:
    image: kodemed/codingui:latest
    ports: ["3000:3000"]

# OIDC Provider (external IdP, not bundled)
# https://customer-sso.example.com/auth
```

Client Deployment

- CodingClient: portable self-contained EXE
- Copy to any folder and run
- Auto-registers DLL + env vars
- Citrix/VDI: publish as app or in image
- Group Policy: set KODEMED_HOME env var
- Silent first-run setup
- Config override: kodemed-client-config.json
- Auto-update: planned for future versions

Deployment Topology

Kubernetes / OpenShift with Harbor Registry

Harbor Registry

- kodemed/server:latest
- kodemed/dataserver:latest
- kodemed/grouper:latest
- kodemed/codingui:latest

Trivy scan + Cosign/Notary

pull >>>

Kubernetes / OpenShift

Ingress / Route

Service	HPA	Port
kodemed-server	Yes	:8080
kodemed-dataserver	Yes	:8081
kodemed-grouper	-	:8082
kodemed-ui	-	:3000

PostgreSQL (StatefulSet / external)

- Helm charts or plain K8s manifests
- Horizontal Pod Autoscaling (HPA) for Server & DataServer

- Private registry for image storage & distribution
- Vulnerability scanning (Trivy) on every push

Configuration

XML/JSON config, environment variables, webhook

DLL / Client Config (XML or JSON)

```
<KodeMedConfig>
  <ServerUrl>https://api.example.com
  </ServerUrl>
  <CodingUIUrl>https://ui.example.com
  </CodingUIUrl>

  <!-- OAuth2 -->
  <OAuth2Url>https://sso/auth</OAuth2Url>
  <OAuth2Realm>kodemed</OAuth2Realm>
  <OAuth2ClientId>kodemed-dll</OAuth2ClientId>

  <!-- Language (de/fr/it/en, auto-detect) -->
  <Language>de</Language>

  <!-- Processing -->
  <OutputFormat>XML</OutputFormat>
  <ValidateSchema>true</ValidateSchema>
</KodeMedConfig>
```

Key Settings

Setting	Default	Description
ServerUrl	-	Main server URL
OAuth2Url	-	Identity provider
Language	auto	de, fr, it, en
OutputFormat	XML	XML or JSON
SessionExpiry	60	Minutes
OfflineTimeout	5	Minutes

Webhook Env Vars (Server)

Variable	Default
KODEMED_HOOK_ENABLED	false
KODEMED_HOOK_URL	(empty)
KODEMED_HOOK_EVENTS	applied
KODEMED_HOOK_RETRY_COUNT	3

Quick Start

Get running in 3 steps

Step 1: Deploy Server

```
docker compose up -d    # Starts Server, DataServer, GrouperServer, CodingUI
```

Step 2: Install Client (choose one)

```
# Option A: CodingClient (recommended)
# Just run KodeMed.CodingClient.exe
# Auto-setup handles everything
```

```
# Option B: MSI silent install (GPO/SCCM)
msiexec /i KodeMed.msi /quiet SERVERURL="https://..."
```

Step 3: Start Coding

```
var coding = new Coding();
coding.SendConfig(File.ReadAllText("kodemed-config.xml"));
coding.DoCodingWithFormat(spigesXml, FormatType.SpiGes);     // Opens auth + coding UI
string results = coding.GetResults();                         // XML or JSON results
```