KodeMed

Integration Guide

For Software Developers & System Integrators

Version 2026.5.15.59502 | KodeMed AG

Integration Overview

Two integration paths to KodeMed

Option A: COM DLL

Direct HIS Integration

- Load DLL via KODEMED_DLL_PATH env var
- Call SendConfig() with XML or JSON
- Call DoCodingWithFormat() for full UI flow
- Call DoCoding() for headless processing
- Retrieve results via GetResults()
- Works from .NET, VB6, VBA, C#, Delphi

Option B: REST / WebSocket API

3rd Party Apps via CodingClient

- POST /api/v1/coding/session with data
- Server pushes to CodingClient via WebSocket
- CodingClient opens coding UI on user’s desktop
- Poll session status via GET endpoint
- Any language/platform (Java, Python, ...)
- Real-time updates via WebSocket

Option A: COM DLL Integration

Step-by-step for .NET / VB / C# applications

C# Example

```
var coding = new Coding();

// 1. Configure
coding.SendConfig(configXml);

// 2. Process with UI (auth + browser)
coding.DoCodingWithFormat(data,
FormatType.SpiGes);

// 3. Get results
string results = coding.GetResults();

// 4. Cleanup
coding.DisposeClient();
```

VB.NET / VBA Example

```
Dim km As Object
Set km = CreateObject("KodeMed.Coding")
km.SendConfig configXml
km.DoCodingWithFormat data, 0   ' SpiGes
result = km.GetResults()
km.DisposeClient
```

DLL Path Resolution

- KODEMED_DLL_PATH env var (set by CodingClient)
- COM: CreateObject("KodeMed.Coding")
- Reflection: Assembly.LoadFile(dllPath)

Option B: REST API Integration

For any language/platform

Endpoint Flow

Step	Method	Endpoint	Description
1	POST	/api/v1/coding/session	Create session with case data + format
2	-	(WebSocket push)	Server pushes CODING_SESSION_LAUNCH to CodingClient
3	GET	/api/v1/coding/session/{id}	Poll session status
4	GET	/api/v1/coding/session/{id}/result	Retrieve coding results

Example: Create Session (curl)

```
curl -X POST https://server/api/v1/coding/session \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/json" \
  -d '{"data": "<SpiGes XML>", "format": "spiges", "instanceId": "my-app-1"}'

# Response: {"sessionId": "abc-123", "redirectUrl": "https://ui/spiges/session/abc-123/cases"}
```

Option C: WebSocket Integration

Real-time bidirectional communication

WebSocket Message Types

Type	Direction	Description
CONNECTED	S -> C	Connection established
HEARTBEAT	S -> C	Keep-alive ping
HEARTBEAT_ACK	C -> S	Heartbeat response
CODING_SESSION_LAUNCH	S -> C	Launch coding session
SESSION_ACTION	C -> S	Apply/Discard from UI
SESSION_CLOSE	C -> S	DLL closes session
UI_SESSION_JOIN	C -> S	UI joins coding session

Connection

```
// Connect
ws://server:8080/ws/dll

// With auth token
ws://server:8080/ws/dll
  ?token=<jwt>
  &instanceId=<id>

// Auto-reconnect: 60s
```

Payload Format (JSON)

```
{
  "type": "CODING_SESSION_LAUNCH",
  "instanceId": "dll-xyz",
  "payload": { "data": "...",
    "format": "spiges" }
}
```

Authentication

OAuth2 / OpenID Connect

OAuth2 / OIDC Flow

- Compatible with: Keycloak, Auth0, Azure AD, Okta
- Authorization Code flow with PKCE
- DLL opens embedded browser for login
- Token auto-refresh (access + refresh tokens)
- User locale read from JWT ‘locale’ claim
- Client IDs: kodemed-dll (DLL), kodemed-ui (browser)
- Realm: configurable (default: kodemed)

Configuration

```
<OAuth2Url>
  https://sso.example.com/auth
</OAuth2Url>
<OAuth2Realm>kodemed</OAuth2Realm>
<OAuth2ClientId>kodemed-dll</OAuth2ClientId>
```

Token Lifecycle

- Access token: short-lived (~5 min)
- Refresh token: long-lived (~30 min)
- Auto-refresh before expiry
- WebSocket reconnect refreshes token

Data Formats

SpiGes XML, BFS, Custom

SpiGes XML Structure (Primary)

```
<?xml version="1.0" encoding="UTF-8"?>
<Unternehmen xmlns="...spiges-data/1.5"
  ent_id="100000001" version="1.5">
  <Standort burnr="10000001">
    <Fall fall_id="1">
      <Administratives
        abc_fall="A" alter="38"
        geschlecht="1"
        eintrittsdatum="2022061010"
        austrittsdatum="2022061209" />
      <Diagnose diagnose_id="1"
        diagnose_kode="K3530"
        diagnose_poa="3" />
      <Behandlung behandlung_id="1"
        behandlung_chop="4701"
        behandlung_beginn="202206110000" />
    </Fall>
  </Standort>
</Unternehmen>
```

FormatType Enum

Value	Format	Extensions
0	SpiGes	.xml
1	BFS	.dat, .xml
2	Custom	plugin-defined

Plugin System

- Implement IFormatPlugin interface
- Auto-discovery from plugin directory
- Transform custom data to SpiGes model
- Built-in: JSON Clinical Plugin

CodingClient Setup

Portable system tray application

Installation (3 Steps)

- 1. Copy KodeMed folder to any location
- 2. Run KodeMed.CodingClient.exe
- 3. Auto-setup: env vars + COM DLL registration

Environment Variables (auto-set)

Variable	Value
KODEMED_HOME	Working directory
KODEMED_DLL_PATH	Full path to KodeMed.dll
KODEMED_EXE_PATH	Full path to CodingClient.exe

Citrix / VDI Compatibility

- Self-contained EXE: no .NET runtime needed
- User-level only: all writes to HKCU
- No admin rights required
- COM registration via HKCU\Software\Classes
- Config: next to EXE or %KODEMED_HOME%
- Non-persistent VDI: re-setup on each login

Tray Menu (multilingual: DE/FR/IT/EN)

- Sign In / Sign Out (OAuth2)
- Open Session from File (SpiGes / BFS / Custom)
- Language (DE/FR/IT/EN)
- Server Settings, Auto-start, Setup / Repair

API Quick Reference

DLL Methods, REST Endpoints, WebSocket Messages

DLL Methods (IKodeMed)

Method	Description
SendConfig(string)	Load XML/JSON configuration
DoCoding(string, FormatType)	Process data (headless, no UI)
DoCodingWithFormat(string, FormatType)	Process with UI (auth + browser)
GetResults()	Get results (XML or JSON)
DisposeClient()	Release all resources

REST Endpoints

Method	Endpoint	Action
POST	/api/v1/coding/session	Create session
GET	/api/v1/coding/session/{id}	Get status
GET	/api/v1/coding/session/{id}/result	Get results
POST	/api/v1/coding/session/{id}/complete	Complete session
POST	/api/v1/coding/session/{id}/cancel	Cancel session
GET	/api/v1/config	Get server config

WebSocket Messages (ws://server/ws/dll)

Message	Direction	Payload	Description
CODING_SESSION_LAUNCH	Server -> Client	data, format, sessionId	Launch coding UI for new session
SESSION_ACTION	Client -> Server	sessionId, action	User applied or discarded changes
SESSION_CLOSE	Client -> Server	sessionId	DLL/WebView2 closed the session

DLL-side Webhook

Fired by CodingClient — the legacy in-process flow

How It Works

- Coder completes session → CodingClient POSTs JSON
- Fire-and-forget: does not block the coder
- Exponential backoff retry (1s→2s→4s... max 60s)
- No retry on 4xx (except 429 Too Many Requests)

Server Config (env vars)

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

Client Config (local auth)

```
// kodemed-client-config.json
{
  "hook": {
    "authType": "bearer",
    "authToken": "sk-hospital-key"
  }
}
```

Data Minimization (GDPR/nDSG)

- includeResultData defaults to false (opt-in)
- includeGrouperResults defaults to false (opt-in)
- Auth credentials never sent from server
- SSRF protection on webhook URLs

HIS REST Webhook Contract

Server-side dispatch on case.coded / case.discarded

Trigger

- Coder presses Apply → event_type = case.coded
- Coder presses Discard → event_type = case.discarded
- Per-tenant config: IntegrationProfile.webhookUrl + webhookEventTypes
- Fired by the Server (NOT the DLL) — partners without a DLL use this

Idempotency

- Idempotency-Key header: <session_id>:<event_type>
- Stable across retries — use it to deduplicate on the HIS side
- Delivery is at-least-once; design the receiver to handle duplicates

Payload (HisWebhookPayload)

```
{
  "event_type":  "case.coded",
  "occurred_at": "2026-05-12T21:45:12Z",
  "session_id":  "550e8400-…-446655440000",
  "instance_id": "his-instance-12345",
  "spiges": {
    "ent_id":  "01.01.01.01",
    "burnr":   "0000000001",
    "fall_id": "F-2026-00001"
  },
  "result_data": "<spiges>…</spiges>"
}
```

Webhook Signing + Retry + DLQ

HMAC-SHA256 signed, retried with backoff, DLQ for admin replay

HMAC-SHA256 Signature

- Header: Authorization: HMAC-SHA256 t=<unix>,v1=<hex-sig>
- Signature = HMAC-SHA256(secret, "<t>.<raw-body>")
- Sent only when IntegrationProfile.webhookSigning=HMAC_SHA256
- Verify with constant-time compare; reject if t skew > 5 min (replay)

Retry Schedule

- Default 4 attempts at 30s, 2m, 10m, 1h
- Retryable: 5xx, 408, 429, network errors
- Permanent: 2xx (delivered), 4xx (except 408/429)
- Configurable per profile: webhookRetryDelaysSeconds, webhookRetryMax

Dead-Letter Queue

- On final failure: row persisted to WebhookFailure table
- List failures: GET /api/v1/admin/webhook-failures
- Inspect one: GET /api/v1/admin/webhook-failures/{id}
- Replay: POST /api/v1/admin/webhook-failures/{id}/replay

```
# Replay a failed delivery
curl -X POST \
  -H "Authorization: Bearer <admin-jwt>" \
  https://kodemed/api/v1/admin/\
    webhook-failures/<id>/replay
```

KodeMed

Contact & Support

info@kodemed.ch
www.kodemed.ch