DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

ARCHITECTURE REFERENCE

ODT-MIM

USAREUR-AF

MIM Future Components

Architecture Reference

HEADQUARTERS
UNITED STATES ARMY EUROPE AND AFRICA
(USAREUR-AF)
Wiesbaden, Germany

DRAFT — NOT FOR OFFICIAL USE. FOR TRAINING PLANNING PURPOSES ONLY.

20 MARCH 2026

DRAFT — UNOFFICIAL — NOT FOR OPERATIONAL USE

MIM FUTURE COMPONENTS

Planned components not yet backed by implementation code. Each entry documents the intended purpose so the stubs can be rebuilt when work begins. Grouped by layer.

APPS

examples

Runnable bridge scenarios demonstrating adapter and SDK usage, plus validation and contract usage samples. Each example should document its own setup and consume only published package APIs.

mim-studio

User-facing workflow/authoring application for MIM model editing and visualization. Entry points, local run instructions, and sample data/fixtures to be defined.

mim-ui

User-facing UI application (distinct from mim-studio). Scope to be differentiated from mim-studio when work begins.

playground

Interactive experimentation environment for exploring MIM models and running ad-hoc queries against the SDK.

FUTURE PACKAGES

Planned packages intended to become stable, versioned public APIs with tests and documentation.

future-packages/contracts

Public contracts package — shared type definitions and validation schemas for cross-package interoperability.

future-packages/mim-sdk

MIM SDK — stable public API surface for programmatic access to MIM models, queries, and transformations.

future-packages/transform

Transform package — MIM model/IR transformation operations (model-to-model, model-to-IR, IR-to-artifact pipelines).

INFRASTRUCTURE

Infrastructure-as-code directories.

infra/docker

Docker containerization configuration for MIM services.

infra/helm

Helm charts for Kubernetes deployment of MIM services.

infra/terraform

Terraform infrastructure-as-code for cloud resource provisioning.

ADAPTERS

Integration packages that bridge between MIM SDK models and external systems. Each adapter translates between MIM instances and external formats, surfaces transport/API errors consistently, and emits logs, metrics, and trace events.

Inputs: MIM SDK instance data, connection config, credentials. Outputs: Remote system writes or extracted instance data.

Adapter	Target
`mim-adapter-dlt`	DLT (Data Load Tool) pipelines
`mim-adapter-files`	Local/remote file system I/O
`mim-adapter-foundry`	Palantir Foundry platform
`mim-adapter-kafka`	Apache Kafka event streaming
`mim-adapter-rest`	Generic REST/HTTP APIs
`mim-adapter-s3`	Amazon S3 / S3-compatible object storage
`mim-adapter-sql`	SQL databases

BACKENDS

Code-generation backends that consume IR models and produce target-language artifacts. Each backend applies target-specific naming conventions and emits trace manifests for traceability.

Backend	Output
`mim-backend-openapi`	OpenAPI specification artifacts from IR
`mim-backend-python`	Python code artifacts from IR (feeds `dist/python/`)
`mim-backend-ts`	TypeScript code artifacts from IR (feeds `dist/ts/`)

SHARED

shared/mim-provenance

Provenance and lineage tracking for MIM artifacts. Tracks generation provenance across the full pipeline (source → IR → generated artifact). Enables `mim trace <symbol>` lookups across backends. Supports schema drift detection and auditability.