Terms of Reference

Reusable Component: API Integration Framework

Mendix Reusable Components Initiative - ISAA

Version	Status
0.0.1	Draft

1. Background

This Terms of Reference document is issued as part of ISAA's (Information Systems Agency of Armenia) initiative to build a standard, reusable foundation of Mendix components for public digital services. Full context, goals, and target users for the overall initiative are described in the accompanying **Concept Note**.

Public digital services built on Mendix routinely need to exchange data with external public systems - population, business, address, and cadastre registries, tax and social systems, and other authoritative sources. This component addresses the gap to avoid, each project implementing this integration independently, leading to duplicated effort and inconsistent handling of authentication, error handling, and data mapping.

2. Objective

Develop a configurable, reusable **API Integration Framework** for Mendix applications that standardizes integration with external government systems and services through APIs, providing reusable capabilities for secure API consumption and data exchange across Mendix applications.

3. Scope of Work

3.1 Functional Requirements

Standardized API Integration Model

Provide a common approach for integrating Mendix applications with external systems through APIs.

Reusable API Integration Capabilities

The framework should provide reusable capabilities for:

- API configuration and consumption;
- Authentication and secure communication;

- Request/response handling;
- Data mapping and transformation;
- Error handling and retry mechanisms;
- API monitoring and operational management.

REST API Support

For external systems accessed directly (not via X-Road), the framework should support integration via REST APIs, covering standard HTTP methods for both retrieving data (GET) and submitting or updating data (POST, PUT/PATCH, DELETE), with JSON as the primary payload format. Where such a system publishes an OpenAPI/Swagger specification, the framework should support importing it to accelerate configuration of a new integration.

X-Road Integration

Government registries and services are exposed behind the national **X-Road** data exchange layer rather than directly. The framework must support consuming services through X-Road, in addition to direct REST integrations, including:

- Routing requests through the organization's local X-Road Security Server rather than calling backend services directly;
- Addressing services using X-Road identifiers (X-Road instance, member class, member code, subsystem code, service code, version) rather than plain URLs;
- Constructing and parsing X-Road message headers (e.g., client identifier, service identifier, message ID, and representing-party information where a request acts on behalf of a citizen or organization);
- Support for both X-Road REST-style services.

Vendors should describe how the framework distinguishes between an X-Road-based integration and a direct REST integration at configuration time, and how X-Road message IDs can be captured for correlation with the Audit & Activity History Framework.

Configuration-Based Adoption

New API integrations should be enabled through configuration where possible, minimizing custom development effort.

Vendors should describe:

- How a new API integration is added (configuration steps vs. required development);
- Which authentication/secure-communication mechanisms are supported out of the box (e.g., OAuth2, mTLS, API keys) and how new mechanisms can be added;
- How request/response mapping and transformation are configured without custom code;
- How retries, timeouts, and circuit-breaking are configured per integration;

- How API monitoring and operational visibility (call volume, failures, latency) are exposed to administrators.

3.2 Non-Functional Requirements

- **Security**: All external API traffic encrypted in transit (TLS 1.2+); credentials and secrets stored using Mendix-supported secret management, not hardcoded in configuration.
- **Performance**: Framework overhead should not materially degrade response times for read APIs; asynchronous handling should be supported for long-running or write operations.
- **Resilience**: Configurable retry and timeout policies per integration; graceful degradation when an external system is unavailable.
- **Observability**: Structured logging of API calls (request/response metadata, not necessarily full payloads) sufficient to support the Audit & Activity History Framework and operational troubleshooting.
- **Compatibility**: Must run on *Mendix version > 11.10.0* and be upgradable across supported Mendix versions.

4. Deliverables

Vendors are expected to provide the following, in addition to the specific requirements above:

Standard Implementation Deliverables

#	Deliverable	Description
1	Mendix Source	Mendix application source (Team Server/Git) and deployment package.
2	Technical Documentation	Module purpose, architecture, dependencies, exposed entities, microflows, Java actions, and configuration.
3	Architecture Documentation	Architecture overview and key design decisions.
4	Security Configuration	Module roles, user role mappings, authentication and authorization configuration.
5	API Documentation	Interface specifications, request/response schemas, sample payloads, and error handling.
6	Configuration Guide	Environment-specific constants and configuration settings.
7	Third-Party Component Inventory	Marketplace modules and external dependencies used.
8	Code Quality Report	Static code analysis and Mendix quality metrics.

9	Test Plan	Unit, integration, system, UAT, regression, and non-functional testing approach.
10	Test Results	Test cases and execution evidence.
11	Security Test Report	Security testing and vulnerability assessment results (where applicable).
12	Performance Test Report	Performance and load testing results (where applicable).
13	User Documentation	User and administrator guides.
14	Training Materials	Training guides and supporting materials (where applicable).

Additional Deliverables for Reusable Components

#	Deliverable	Description
1	Module README	Purpose, prerequisites, installation and upgrade instructions.
2	Interface Contract	Public microflows, entities, Java actions, parameters, and expected outputs.
3	Configuration Guide	Post-import configuration instructions.
4	Sample Application	Reference application demonstrating component implementation, including at least one worked example integration (e.g., State Population Registry).

5. Vendor Response Requirements

Vendor proposals should address:

1. **Technical approach** - proposed architecture, how it maps to the functional requirements in Section 3.
2. **Configuration model** - concretely how a new API integration (e.g., a new registry) is onboarded, and what is/isn't possible without custom development.
3. **Security approach** - supported authentication schemes and secret management.
4. **Reuse & extensibility** - how the framework avoids tight coupling to any single integration, and how it will be consumed by other components in this initiative.
5. **Team & experience** - Experience with Mendix (certificates, if available) and completed relevant projects.
6. **Timeline** - proposed delivery schedule against the deliverables in Section 4.
7. **Indicative Cost** - cost breakdown by phase/deliverable.
8. **Assumptions & risks** - any assumptions made and risks identified.