Terms of Reference

Reusable Component: Notification 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 routinely need to notify applicants and stakeholders about business events (submission received, status changes, approvals, rejections, payments, certificate issuance). Actual delivery is handled by an **centalized government notification platform** external to this component. This component addresses the gap where each project would otherwise implement its own notification logic and its own integration with that external platform independently, leading to duplicated effort and inconsistent handling of templates, recipients, and delivery failures.

2. Objective

Develop a configurable, reusable **Notification Framework** for Mendix applications that provides a standardized approach to generating, managing, and delivering notifications across government digital services. The framework shall integrate with the existing government notification platform to deliver notifications through supported communication channels, while allowing individual applications to configure notification events, recipients, and message templates without implementing channel-specific delivery logic.

3. Scope of Work

3.1 Functional Requirements

Standardized Notification Model

Provide a standardized notification model that enables applications to consistently define and trigger business notifications regardless of the underlying delivery channel.

Applications should be able to configure:

- Notification events;
- Recipient determination rules;
- Notification templates;
- Delivery priorities;
- Channel preferences (subject to capabilities provided by the external notification service).

Event-Based Notification Generation

Provide reusable mechanisms for generating notifications in response to business events.

Applications should be able to trigger notifications from configurable custom business events such as:

- Application submitted;
- Application approved;
- Application rejected;
- Payment received;
- Certificate issued;
- Status changed;

Template Management

Provide reusable support for configurable notification templates.

Templates should support:

- Parameterized content;
- Multiple languages;

Integration with External Notification Service

The framework forwards notification requests to the external government notification service, independent of the underlying delivery channel or implementation, with defined error handling for when the service is unavailable or rejects a request.

The external notification service is expected to expose a single API endpoint (most likely a single write API accepting a notification request, regardless of the eventual delivery channel). As such, this integration is a natural candidate to be implemented on top of the **API Integration Framework** rather than as bespoke integration code within this component.

Vendors should describe:

- Whether and how the framework uses the API Integration Framework to call the external notification service, rather than implementing its own separate integration logic;

- The protocol, request/response model, and authentication used to reach the external notification service;
- How delivery failures, timeouts, and rejections are handled - whether requests are retried/queued, and how the calling application is informed;
- How channel selection is resolved when a recipient's preferred channel is unavailable or unsupported by the external service.

Notification History

Provide reusable capabilities for recording notification requests and their processing status within the application where required.

Applications should be able to:

- View notification history related to business transactions;
- Track notification request status.

Vendors should describe how notification history relates to the **Audit & Activity History Framework** - specifically, whether a sent/failed notification is expected to also be logged as a business activity via that framework's non-CRUD business event mechanism, or whether Notification History is maintained as a self-contained log.

3.2 Non-Functional Requirements

- **Reliability**: Notification requests should not be silently lost if the external service is temporarily unavailable; the framework should support at-least-once delivery semantics toward the external service (e.g., via retry or queuing).
- **Performance**: Notification generation should not block the business transaction that triggers it (e.g., submitting an application should not wait on notification delivery to complete).
- **Security**: Notification content and recipient data must be handled in line with applicable data protection requirements, particularly for channels involving personal contact details (phone, email).
- **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 event with a parameterized, multi-language template.

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 notification event, recipient rule, and template are onboarded, and what is/isn't possible without custom development.
3. **External service integration** - integration approach with the government notification platform, including failure handling (see Section 3.1).
4. **Reuse & extensibility** - how the framework will be consumed by other components in this initiative (e.g., Application & Case Management, Audit & Activity History).
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.