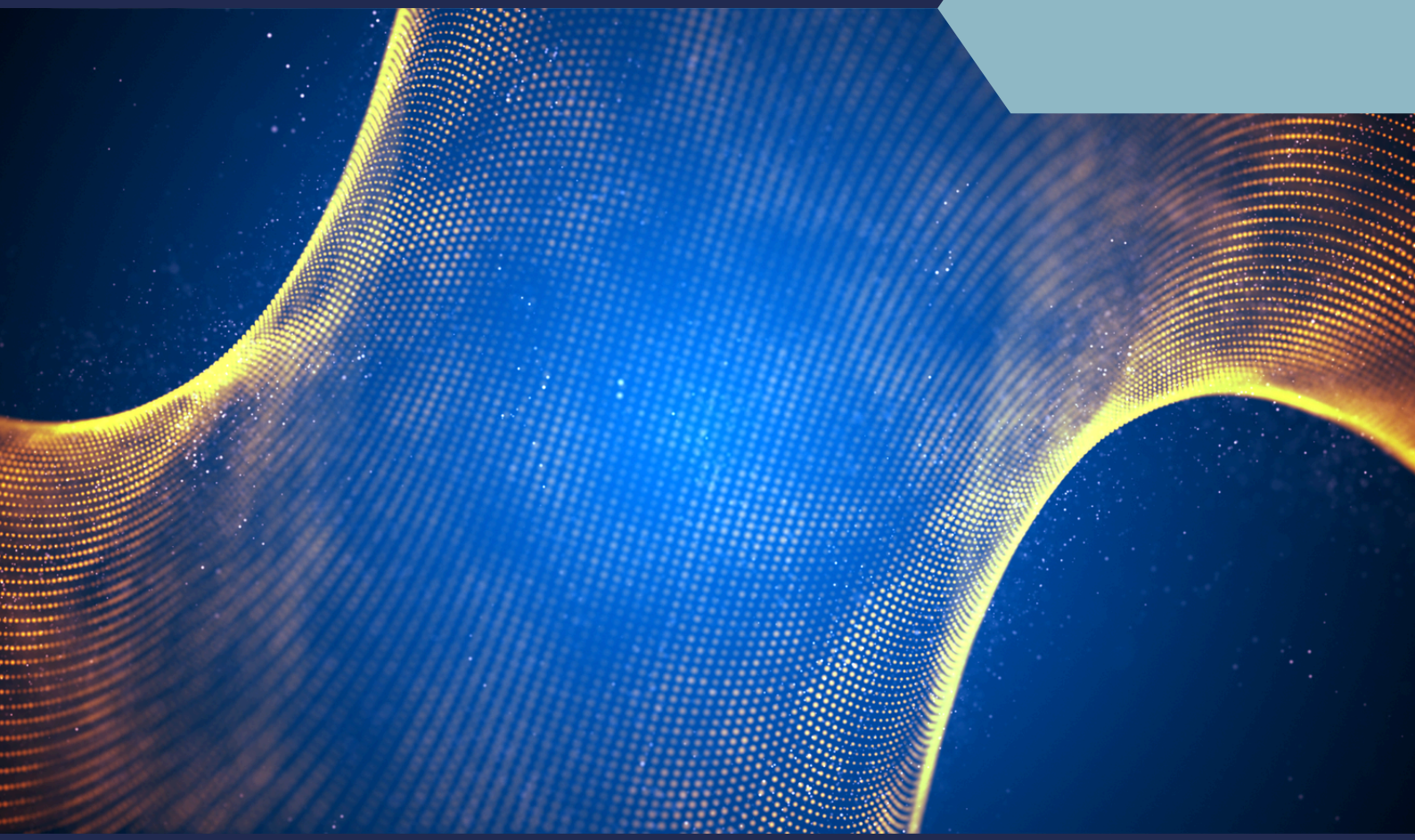
NETERIUM



NETERIUM INTEGRATION FRAMEWORK

INTRODUCTION TO THE NETERIUM SOFTWARE DEVELOPMENT KIT AND SAMPLE APPLICATIONS

Welcome to the Neterium Integration Framework

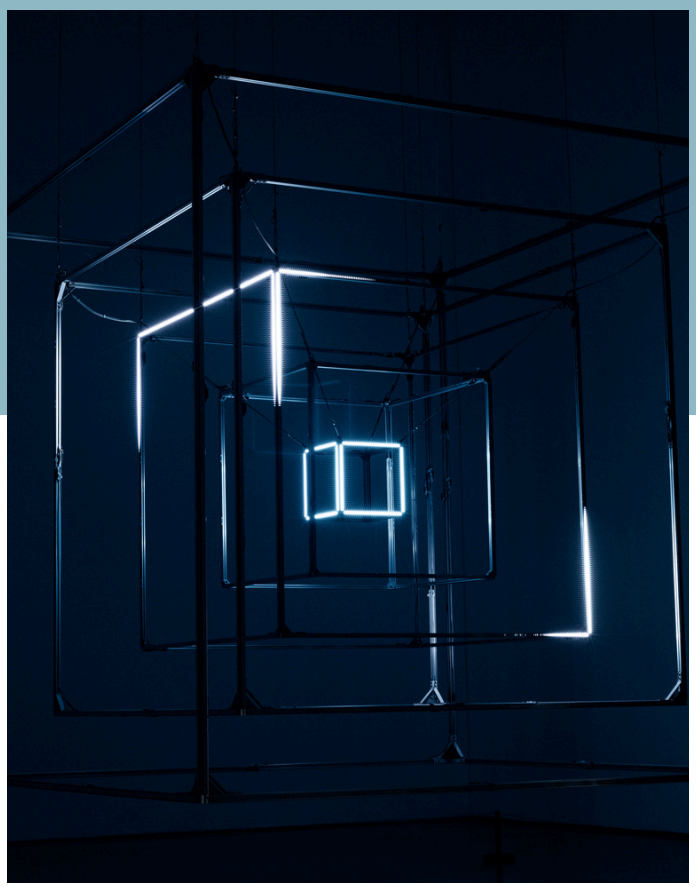
Welcome to the Neterium Integration Framework
This framework offers a structured toolkit that simplifies and accelerates the integration of the Neterium Screening API into Java applications. It is designed to promote clarity, maintainability, and adherence to best practices, enabling developers to focus on implementing business logic rather than dealing with infrastructure or low-level integration details.
The purpose of this document is to provide a high-level overview of the framework and its components. Detailed technical documentation, along with all components and sample applications described below, is available on GitHub.

Overview

The framework consists of two main parts:

1. The SDK (Software Development Kit) components — a set of reusable building blocks built on the Spring Framework, the standard for modern Java applications.
2. Simple, real-world sample applications that demonstrate how quickly and easily the SDK can be implemented in practice..

Together, these components help you move seamlessly from initial testing to a production-ready integration.



Prerequisites

As a developer, and in order to maximize efficiency, we recommend gaining a basic understanding of Neterium's offering as well as the fundamentals of its API. This information is available in the Introduction to Neterium and Getting Started for Developers guide.

To work effectively with the SDK, you should have a good understanding of Java 21, Spring 6 and Spring Boot 3, Maven, the OAuth2 protocol, and OpenAPI (particularly concepts related to HTTP and JSON) as well as basic multi-threading principles.

In addition, a basic understanding of HTML and JavaScript, along with basic knowledge of MongoDB, will help you better understand how the sample applications operate.

.

The SDK

The SDK provides all the key components you need to use the Neterium API safely and efficiently. It covers common use cases out of the box, while keeping room for customization.
The goals of the SDK are to:

- Accelerate development through ready-to-use components
- Hide unnecessary complexity such as authentication, throttling, and related concerns
- Ensure your application interacts with the API the right way, taking full advantage of the API rich features

Technical prerequisites
The SDK relies on the Spring Framework (specifically Spring Boot) for configuration, dependency injection, and lifecycle management.

> While the SDK's components are designed to be used directly within a Spring Boot application, developers remain free to reuse or adapt the source code and integrate it into any Java application — with or without Spring Boot.

The following software needs to be installed on your machine to compile the SDK and run sample applications:

- **Java 21**
- **Maven 3.9**
- **Git**

SDK Components

The components are grouped into *business components* and *operational components* and are further described below:

Business Components		
	Jetscan	**Jetflow**
Mapping component	Maps a CSV-file containing name records (i.e. physical persons or entities) into a JSON screening request payload. The number of name records per request is configurable	Maps a file containing credit transfers (in ISO 20022 pacs.008 or FIN MT 103 format) into a JSON. (Note: other formats can be mapped through a configuration file) The number of transactions per request is configurable
Screening Component	Takes as input a Jetscan API Screening Request, orchestrates its sending to the screener and provides the API Screening Response as output	Takes as input a Jetflow API Screening Request, orchestrates its sending to the screener and provides the API Screening Response as output
Hit handler component	This component determines whether a Jetscan hit has already been encountered. It does so by comparing a checksum (computed from the Watchlist Profile and included in the API response) with a checksum previously received and stored by the user. If the checksums do not match (i.e., no previous hit exists), the API response is returned as output (and the checksum is stored for subsequent comparisons). If the checksums match, the system returns the decision previously made for that hit.	
Private lists	This component maps private list data provided in CSV format into an XML OFAC SDN format file, which is the input format required by Jetflow/Jetscan. The file is then subsequently uploaded.	
Operational Components		
Throttling component	This component optimizes the rate at which (Jetscan or Jetflow) API requests are sent to the screening platform, ensuring controlled, stable, and compliant usage of the API.	
Instrumentation component	Collects operational data—such as performance metrics—which can readily be displayed for monitoring purposes. The Throttling component leverages these data to optimize the rate at which API requests are sent.	
Authentication	Manages the generation and renewal of (bearer) authentication tokens, supporting two renewal modes: • **Proactive** renewal: tokens are refreshed just before they expire. • **Lazy** renewal: tokens are checked and renewed only when required. Using this component, authentication is fully automated, ensuring the application communicates with the API reliably without any manual authentication intervention.	

Sample applications

To demonstrate the ease of leveraging the SDK components and integrating our APIs, we provide a suite of ready-to-run sample applications showcasing real-world use cases and implementation best practices.
These applications, available separately for Jetscan and Jetflow, include:

- **Screening application:** Generates a random dataset (customer records or transactions) and sends them to the screening engine
- **Throttling application:** the throttling application dashboard demonstrates the real-time operation of the throttling component. It allows to monitor how the volume of submitted screening requests is managed, controlled, and limited according to predefined metrics, ensuring system stability and preventing overload
- **L1 Alert Manager:** demonstrates how the various SDK components can be leveraged to implement L1 alert management functionality. It provides a simple yet effective user interface that illustrates how this functionality can be implemented in practice.

Next steps

You are welcome to download the Integration framework components (i.e. SDK and sample applications) from GitHub: https://github.com/neterium