RSA Helper version 1.2.7

Support Application for KSeF Processes

Table of Contents

---

1. General Information

RSA Helper is a web application (WebAPI) supporting processes related to e-invoice handling in the KSeF system. The application provides the following functionalities:

- Encryption of authorization queries using RSA algorithm

- Invoice encryption using AES-256-CBC algorithm
- Invoice package (ZIP) encryption using AES-256-CBC algorithm
- Generation of cryptographic data for KSeF 2.0 interactive session
- Conversion of e-invoice files from XML to HTML format
- Conversion of e-invoice files from XML to PDF format
- Validation of e-invoice files in XML format
- Generation of QR codes for invoice verification
- Offline mode support (operation without internet access)

---

2. Requirements

2.1. Hardware Requirements

For proper application operation, the following hardware is required:

- 4-core processor
- 2GB RAM
- 500MB disk space

2.2. Software Requirements

The application requires:

- Web Server:
  - Windows: IIS (recommended) or Apache
  - Linux: Apache or Nginx (recommended)
- ASP.NET Core Runtime version 10.0
- Docker/Podman (optional, for containerization)
- Linux, traditional installation (without a container): **a Chromium browser installed in the system** — required by the default PDF generation engine (see section 3.2). This does not apply to the Docker image, which ships with all dependencies

2.3. Additional Requirements

> **NOTE:** The following requirements for access to external services do not apply in offline mode (see section 3.6).

For proper operation, the application requires access to external Ministry of Finance services:

- for generating authorization data and sending invoices to KSeF:
  - https://api-demo.ksef.mf.gov.pl
  - https://api-test.ksef.mf.gov.pl
  - https://api.ksef.mf.gov.pl
- for XML file validation and visualization:
  - http://crd.gov.pl

Checking Address Availability

To ensure that KSeF and CRD servers are accessible from the machine where the application is installed, you can perform the following tests:

Windows (PowerShell):

```
Test-NetConnection ksef-test.mf.gov.pl -Port 443
```

Windows (CMD):

```
curl -f -s -o nul -w "%{http_code}" https://ksef-test.mf.gov.pl
```

Linux:

```
curl -f -s -o /dev/null -w "%{http_code}\n" https://ksef-test.mf.gov.pl
```

If the server is accessible:

- Windows PowerShell: will display `TcpTestSucceeded : True`
- Windows CMD: will display HTTP status code (e.g., `200`, `301`, or `302`)
- Linux: will display HTTP status code (e.g., `200`, `301`, or `302`)

If the server is unavailable, a connection error message will be displayed.

---

3. Installation

3.1. RSA Helper Downloads

For self-hosting infrastructure, the application is available in two versions:

- **Release (KSeF 2.0):** 1.2.6 – available for windows here
- **Release (KSeF 2.0):** 1.2.6 – available for linux here
- **Develop (KSeF 2.0):** 1.2.7 – available for windows here
- **Develop (KSeF 2.0):** 1.2.7 – available for linux here

> **NOTE:** Downloading the application requires signing in to the client zone. Signing in only requires your email address and a one-time code sent to it — no account registration is needed.

Documentation (English and Polish) and other downloadable files are available on the files page.

3.2. Linux Installation

Traditional Installation (with Web Server)

To run the application on Linux, install and configure a web server. Recommended web servers:

- **Nginx** – installation guide available here
- **Apache** – installation guide available here

Additionally, install ASP.NET Core Runtime version 10. Instructions for installing and configuring ASP.NET Core Runtime for Linux systems can be found here.

After installing and configuring the web server, extract the received `RSAHelper.WebAPI.zip` package to the web server root directory.

> **NOTE:** For new installations, after extracting the package, you must create a `Data` directory in the same location and upload the NIP configuration file (see section 3.4)

Browser for PDF Generation (required)

The default HTML to PDF conversion engine (`Syncfusion`) renders documents with a Chromium browser. **The Linux package does not contain a browser** — it is installed in the system and pointed to by the `PDFSettings:BlinkPath` setting. Without this step, PDF generation fails with a `Blink files are missing` error.

The recommended package is `google-chrome-stable` — the same instructions work on Ubuntu and Debian, and `apt` pulls in all required system libraries automatically:

```
cd /tmp
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo apt-get install -y ./google-chrome-stable_current_amd64.deb
google-chrome --version
```

The directory to point the configuration at is `/opt/google/chrome`. The path is set in `appsettings.json`:

```
"PDFSettings": {
  "BlinkPath": "/opt/google/chrome"
}
```

or as an environment variable in the systemd service file (overrides `appsettings.json`):

```
Environment=PDFSettings__BlinkPath=/opt/google/chrome
```

Using the `chromium` package from the distribution repository. The engine looks for an executable named exactly `chrome` in the given directory, while the `chromium` package installs `/usr/lib/chromium/chromium`. A symbolic link is then required, and the configured directory is `/usr/lib/chromium` :

```
sudo apt-get install -y chromium
sudo ln -s /usr/lib/chromium/chromium /usr/lib/chromium/chrome
```

> **NOTE:** On Ubuntu, the `chromium-browser` package installed via `apt` redirects to the snap version, which is cut off from files outside the home directory and is not suitable for this purpose. On Ubuntu, use `google-chrome-stable`.

Example systemd Service File

The following file (`/etc/systemd/system/rsaHelper.service`) contains all settings required for PDF generation:

```
[Unit]
Description=RSA Helper WebAPI
After=network.target

[Service]
WorkingDirectory=/var/www/html
ExecStart=/usr/bin/dotnet /var/www/html/RSAHelper.WebAPI.dll
Restart=always
RestartSec=10
SyslogIdentifier=rsaHelper
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=ASPNETCORE_URLS=http://127.0.0.1:5000
# Private temporary directory for the service — the browser working directory is
# then created in the service namespace and does not collide with runs started
# from a different account
PrivateTmp=yes
# The browser requires a writable home directory
Environment=HOME=/tmp
# Browser directory — see the point above
Environment=PDFSettings__BlinkPath=/opt/google/chrome

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable --now rsaHelper
```

A correct configuration is confirmed by the `PdfGeneration` field in the **Health** endpoint response (see section 4.9) — it should contain `"Available": true`. The result is cached for 5 minutes, so after fixing the configuration restart the service instead of waiting for the refresh.

Docker Installation

Requirements:

- Docker Engine 20.10+
- Docker Compose (optional)

Installation Steps:

1. Download application code or Docker image
2. Create `Data` directory and NIP configuration file (see section 3.4)
3. Run container:

```
# Run container (production version)
docker run -d -p 5000:8080 -v ./Data:/app/Data:ro --name rsahelper-api rsahelper-webapi:latest
```

```
# Run container (develop version)
docker run -d -p 5000:8080 -v ./Data:/app/Data:ro --name rsahelper-api rsahelper-webapi:develop
```

Or with Docker Compose:

Sample docker-compose.yml file can be downloaded from the files page

```
# Start
docker-compose up -d

# Check status
docker-compose ps

# View logs
docker-compose logs -f rsahelper-api

# Stop
docker-compose down
```

Health Check:

```
curl http://localhost:5000/health
```

To verify that the application was installed correctly, call the **Health** endpoint (see section 4.9).

Detailed instructions how to deploy application with containers are available on the files page.

3.3. Windows Installation

On Windows, configure a web server. The recommended server for Windows is **IIS**. Apache server can also be used – installation and configuration guide available here.

Additionally, install **ASP.NET Core Hosting Bundle** version 10. Installation instructions for .NET Hosting Bundle version 10 can be found here.

> **IMPORTANT:** Install the Hosting Bundle (not just the Runtime), which includes both ASP.NET Core Runtime and the IIS module.

After installing the Hosting Bundle, in the **Application Pool** configuration in IIS, set **.NET CLR Version** to **"No Managed Code"**.

After installing and configuring the web server, extract the received `RSAHelper.WebAPI.zip` package to the web server root directory.

To verify that the application was installed correctly, call the **Health** endpoint (see section 4.9).

> **NOTE:** For new installations, after extracting the package, you must create a `Data` directory in the same location and upload the NIP configuration file (see section 3.4)

3.4. License File Download

The license file can be downloaded from the client zone (sign-in required, see section 3.1). The file can be loaded into the application in two ways:

1. By directly uploading it to the Data directory on the web server (for Docker-based installation to the mapped Data directory)
2. By calling the /api/Authorization/upload endpoint (see section 4.1.4)

3.5. Proxy Configuration

The appsettings.json file contains a section responsible for proxy configuration, which should be completed so that the RSA Helper application can communicate with external services through a proxy. Change the "Enabled" property to true and enter proxy information in the Address, Username and Password fields (username and password are optional and should only be filled in if the proxy requires authentication)

```
"KSeFClientOptions": {
    ...
    "Proxy": {
      "Enabled": false,
      "Address": "http://proxy.mycompany.com:8080",
      "Username": "",
      "Password": ""
    }
  }
```

Other `KSeFClientOptions` Parameters

The `KSeFClientOptions` section configures the KSeF.Client library used internally by the application. Besides `Proxy`, it contains the following parameters:

```
"KSeFClientOptions": {
  "customHeaders": {},
  "ResourcesPath": "Resources",
  "DefaultCulture": "pl-PL",
  "SupportedCultures": ["pl-PL", "en-US"],
```

```
  "SupportedUICultures": ["pl-PL", "en-US"]
}
```

Parameter	Description	Default
`customHeaders`	Additional HTTP headers attached to every request sent to the KSeF API	`{}` (none)
`ResourcesPath`	Path to the resource (localization files) directory used by the KSeF.Client library	`Resources`
`DefaultCulture`	Default culture used by the client library (e.g. error messages)	`pl-PL`
`SupportedCultures`	List of cultures supported by the client library	`["pl-PL", "en-US"]`
`SupportedUICultures`	List of UI cultures supported by the client library	`["pl-PL", "en-US"]`

> **NOTE:** The KSeF API base URL (`BaseUrl`) is not set statically in this section — the application resolves it dynamically based on the `serverType` parameter passed in the request (`prod`, `demo`, `test`, `test2`).

3.6. Offline Mode Configuration

Offline mode enables the application to work without internet access. In this mode:

- KSeF certificates are loaded from local files instead of from the API
- XSD/XSL schemas are loaded from local folders instead of from crd.gov.pl

Operating Modes

Mode	Enabled	FallbackToOnline	Description
Online	`false`	-	Certificates and schemas downloaded from internet
Hybrid	`true`	`true`	Local files first, network fallback if missing
Strict Offline	`true`	`false`	Local files only, no network access

Configuration in appsettings.json

```
"OfflineMode": {
  "Enabled": true,
  "CertificatesPath": "Data/Certificates",
  "FallbackToOnline": true
}
```

Parameters:

- `Enabled` - enables/disables offline mode (default: `false`)

- `CertificatesPath` - path to certificates folder (default: `Data/Certificates`)
- `FallbackToOnline` - whether to try network download when local resources are missing (default: `true`)

Certificate Files Structure

```
Data/
└── Certificates/
    ├── DEMO/
    │   └── certificates.json
    ├── TEST/
    │   └── certificates.json
    └── PROD/
        └── certificates.json
```

certificates.json File Format

```
[
  {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIID...\n-----END CERTIFICATE-----",
    "validFrom": "2024-01-01T00:00:00Z",
    "validTo": "2025-12-31T23:59:59Z",
    "usage": ["SymmetricKeyEncryption", "KsefTokenEncryption"]
  }
]
```

Fields:

- `certificate` - certificate in PEM format
- `validFrom` - validity start date (ISO 8601)
- `validTo` - validity end date (ISO 8601)
- `usage` - list of usage types: `SymmetricKeyEncryption`, `KsefTokenEncryption`

Local XML Schemas

XSD schemas and XSL templates are automatically provided with the application in folders:

- `XmlSchemas/` - XSD schemas for validation
- `XslFiles/` - XSLT templates for transformation

In offline mode, the application uses these local files instead of downloading from http://crd.gov.pl.

> **NOTE:** In Strict Offline mode (`FallbackToOnline: false`) ensure that all required certificates and schemas are available locally. Otherwise, cryptographic operations and XML validation will fail.

Certificate Download Tool (CertificateDownloader)

For automatic downloading of KSeF certificates and saving them in the format required by offline mode, the **RSAHelper.CertificateDownloader** tool is available.

Download: The tool is available for download from the files page (Tools section, no sign-in required).

Usage:

```
# Download certificates for TEST environment
RSAHelper.CertificateDownloader -e TEST

# Download certificates for PROD environment to default folder
RSAHelper.CertificateDownloader -e PROD

# Download certificates for all environments (DEMO, TEST, PROD)
RSAHelper.CertificateDownloader --all

# Download certificates to custom folder
RSAHelper.CertificateDownloader --all -o ./MyCertificates
```

Parameters:

Parameter	Short	Description
`--environment`	`-e`	KSeF environment: `DEMO`, `TEST` or `PROD` (required if not `--all`)
`--output`	`-o`	Output path (default: `Data/Certificates`)
`--all`	`-a`	Download certificates for all environments
`--help`	`-h`	Display help

Example output:

```
=== RSAHelper - Certificate Downloader ===
Pobieranie certyfikatów ze środowiska: TEST
URL: https://api-test.ksef.mf.gov.pl
Łączenie z API KSeF...
Pobrano 2 certyfikatów
Zapisano certyfikaty do: Data/Certificates/TEST/certificates.json

Podsumowanie:
  - SymmetricKeyEncryption:
    Ważny od: 2024-01-01
    Ważny do: 2025-12-31
  - KsefTokenEncryption:
    Ważny od: 2024-01-01
    Ważny do: 2025-12-31
```

> **NOTE:** The tool requires internet access to download certificates from KSeF API. After downloading certificates, you can run RSAHelper application in offline mode.

3.7. Document Conversion Language Configuration

The application allows configuring the default language for HTML and PDF document conversion. This setting affects the ConvertToHtml and ConvertToPdf endpoints.

Configuration in appsettings.json

```
"PDFSettings": {
  "DefaultLanguage": "pl"
}
```

Parameters:

- `DefaultLanguage` - default language for XSLT templates (default: `pl`)
  - Available values: `pl` (Polish), `en` (English)

Operation Rules

When calling conversion endpoints:

- If the `lang` parameter is provided in the query, it takes precedence
- If `lang` is not provided, the configured `DefaultLanguage` is used

Example:

```
# Uses configured default language (e.g., "pl")
curl -X POST http://localhost:5000/api/ConvertToHtml/fromBase64 -d "..."

# Forces English language
curl -X POST "http://localhost:5000/api/ConvertToHtml/fromBase64?lang=en" -d "..."
```

Docker Configuration

To change the default language in Docker environment, create or update `Data/appsettings.additional.json`:

```
{
  "PDFSettings": {
    "DefaultLanguage": "en"
  }
}
```

Then restart the container:

```
docker-compose restart
```

3.8. PDF Print Configuration

The application allows configuring PDF document generation parameters: conversion engine selection, page margins and page orientation. All settings are located in the `PDFSettings` section of `appsettings.json`.

Configuration in appsettings.json

```
"PDFSettings": {
  "PdfEngine": "Syncfusion",
  "PdfOrientation": "Portrait",
  "DefaultFontSize": 26,
  "DefaultQrCodeSize": 350,
  "BlinkPath": "",
  "PdfMargins": {
    "Top": 20,
    "Bottom": 20,
    "Left": 20,
    "Right": 20
  }
}
```

Conversion Engine (`PdfEngine`)

The application supports two HTML to PDF conversion engines:

Value	Description	Requirements
`Syncfusion`	Engine based on Chromium browser. Best rendering quality	Windows and Docker: no additional steps. Linux without a container: a browser installed in the system and the `BlinkPath` setting (see section 3.2 and below)
`WkHtmlToPdf`	Engine based on WebKit	Linux: native libraries required (details below)

WkHtmlToPdf on Linux server (Debian/Ubuntu, SLES, Fedora):

- Required native libraries: `libjpeg62`, `libxrender1`, `libfontconfig1`, `libx11-6`, `libxcb1`, `libxext6`
- An OpenSSL runtime library matching your distribution and wkhtmltopdf binary is also required (e.g. `libssl3` on Ubuntu 24.04 / Debian 12, or `libssl1.1` on older systems)
- Example installation:

```
sudo apt-get update
sudo apt-get install -y --no-install-recommends \
  libgdiplus zlib1g libfontconfig1 libfreetype6 libx11-6 libxext6 libxrender1 libssl3
```

Default value: `Syncfusion`

Browser Directory for the Syncfusion Engine (`BlinkPath`)

The directory containing the Chromium browser that the `Syncfusion` engine renders documents with. The setting applies only to Linux installations **without a container** — the Linux package does not contain a browser. Browser installation is described in section 3.2.

Parameter	Description	Default
`BlinkPath`	Browser directory for the `Syncfusion` engine	empty

Installation method	`BlinkPath` value
`google-chrome-stable` (recommended, Ubuntu and Debian)	`/opt/google/chrome`
`chromium` package from the distribution repository	`/usr/lib/chromium` (requires a `chrome` symbolic link — see section 3.2)
Docker image	empty — the browser is part of the image
Windows (IIS)	empty — the browser is part of the package

The value can also be supplied through the `PDFSettings__BlinkPath` environment variable, which overrides `appsettings.json` — convenient when the same configuration file serves several environments. Changing it requires an application restart.

When the setting is empty, the engine looks for the browser in the `runtimes` directory next to the application. On Linux without a container this results in the error `Blink files are missing at .../runtimes/linux/native`, visible in the `PdfGeneration` field of the **Health** endpoint (see section 3.12).

Page Orientation (`PdfOrientation`)

Value	Description
`Portrait`	Portrait orientation
`Landscape`	Landscape orientation

Default value: `Portrait`

Page Margins (`PdfMargins`)

Margins are specified in millimeters. The setting applies to all conversion engines.

Parameter	Description	Default
`Top`	Top margin	`0`
`Bottom`	Bottom margin	`0`
`Left`	Left margin	`0`
`Right`	Right margin	`0`

Document Font Size (`DefaultFontSize`)

Base root font size (in pixels) for templates that support the `baseFontSize` XSLT parameter (currently the alternative **FA(3)** template, PL and EN). All other sizes in the template are expressed in `rem`, so changing this value scales the whole document proportionally.

Parameter	Description	Default

`DefaultFontSize`	Base document font size in px	`26`

Resolution priority:

1. The `fontSize` parameter passed in the request (query, or the `FontSize` field in the `withQrCodes` body)
2. `PDFSettings.DefaultFontSize` from `appsettings.json`
3. The template default (`26` px) when none of the above is set

A `null` or non-positive value falls back to the template default. Templates that do not declare the `baseFontSize` parameter ignore this setting. Changing `appsettings.json` requires an application restart.

QR Code Display Size (`DefaultQrCodeSize`)

Size (width in pixels) of the QR codes displayed at the end of the invoice visualization, when supplied through the `withQrCodes` endpoints. The size applies equally to both QR codes and is independent of the visualization template used.

Parameter	Description	Default
`DefaultQrCodeSize`	Displayed QR code width in px	`350` (as shipped in `appsettings.json`)

Resolution priority:

1. The `qrCodeSize` parameter passed in the request (the `QrCodeSize` field in the `withQrCodes` body)
2. `PDFSettings.DefaultQrCodeSize` from `appsettings.json` (`350` by default)
3. The built-in fallback (`207` px), only reached if `DefaultQrCodeSize` is removed from `appsettings.json` entirely

A `null` or non-positive value falls back to the default. Changing `appsettings.json` requires an application restart.

3.9. QR Code Renderer and Label Font Configuration

The application allows configuring the renderer and the font used to render labels displayed below QR codes (e.g., KSeF number).

Configuration in appsettings.json

```
"QrCode": {
  "Renderer": "SkiaSharp",
  "FontFamily": "<embedded>",
  "DefaultFontSize": 14,
  "EmbeddedFontPath": "Fonts/DejaVuSans.ttf",
  "BmpBitsPerPixel": 1
}
```

Parameters:

- `Renderer` - renderer used to generate the QR code image and labels. Available values: `SkiaSharp` (default) and `ImageSharp` (new renderer)
- `FontFamily` - font name used for labels (default: `<embedded>`). Use `<embedded>` to use the built-in font (DejaVu Sans from the `Fonts` directory), or specify a system font name (e.g., `Arial`) - the font must then be installed on the

operating system

- `DefaultFontSize` - label font size in pixels (default: `14`)
- `EmbeddedFontPath` - relative path to the embedded font file (default: `Fonts/DejaVuSans.ttf`). Only used when `FontFamily` is set to `<embedded>`
- `BmpBitsPerPixel` - color depth of the output BMP image: `1` = monochrome (small file size), `24` = full color with antialiased text (better label quality). Default: `1`

3.10. KSeF Certificate Authentication Configuration (KsefAuth)

The application supports authenticating with KSeF 2.0 using a client certificate, and storing credentials in a local, encrypted store on the server. Settings are located in the `KsefAuth` section of `appsettings.json` and apply to the `api/authorization-certificates` endpoints (obtaining/refreshing an access token) and the `api/certificates` endpoints (creating, updating, reading metadata for, and deleting locally stored credentials).

Configuration in appsettings.json

```
"KsefAuth": {
  "CredentialStoragePath": "Data/KsefCredentials",
  "MasterKey": "",
  "KeyVersion": "v1",
  "AllowInlineCredentials": true,
  "AllowStoredCredentials": true,
  "AuthenticationTimeoutSeconds": 120
}
```

Parameters:

Parameter	Description	Default
`CredentialStoragePath`	Path to the local credential store directory	`Data/KsefCredentials`
`MasterKey`	Master data-protection key used by RSAHelper to encrypt credentials stored locally. Recommended format: Base64	none (empty)
`KeyVersion`	Encryption key version used when writing new credential records	`v1`
`AllowInlineCredentials`	Whether credentials (e.g. a certificate) may be passed directly in the request body, without being saved to the local store	`true`
`AllowStoredCredentials`	Whether previously stored credentials from the local store may be used	`true`
`AuthenticationTimeoutSeconds`	Maximum time (in seconds) to wait for a KSeF authentication flow to complete	`120`

> **NOTE:** `MasterKey` protects the credentials stored under `CredentialStoragePath`. Set it to a strong, unique secret before deploying to production and keep it out of source control (e.g. in an environment variable or a secrets

> manager). Disabling both `AllowInlineCredentials` and `AllowStoredCredentials` disables certificate-based KSeF authentication entirely.

3.11. Logging Configuration

The application uses the **NLog** library, configured through the `nlog.config` file located in the application directory (next to `RSAHelper.WebAPI.dll`). The file is loaded with `autoReload="true"`, so changes saved to it take effect immediately, without restarting the application or service.

Default Logging Targets

File	Content	Minimum level	Retention
`logs/log-main.log`	All application logs	`Warning`	7 days
`logs/log-errors.log`	Errors only	`Error`	7 days
`logs/log-requests.log`	Incoming HTTP request details (see below)	`Error` (default)	2 days

In addition to the files above, logs at `Info` level and higher are also written to the console.

HTTP Request Logging (`RequestLoggingMiddleware`)

The `RSAHelper.WebAPI.Middleware.RequestLoggingMiddleware` class logs every incoming HTTP request at the very start of the request pipeline (before routing and controller execution) — the method, full URL, protocol, content type and length, all headers, and the request body. The middleware is registered globally and runs in every environment, including production.

These entries are logged at `Information` level, and their destination is controlled by a dedicated rule in `nlog.config`:

```
<logger name="RSAHelper.WebAPI.Middleware.RequestLoggingMiddleware" minlevel="Error" writeTo="requestLogfile
```

By default, this rule's `minlevel` is set to `Error`. Since the middleware only ever logs at `Information` level, no entry meets that threshold by default, and nothing is written to `logs/log-requests.log`. The rule also carries `final="true"`, which stops further rule processing for this logger based purely on the name match — regardless of level — so requests are not written to the console or to `log-main.log` either, even though they would otherwise match the general `<logger name="*" minlevel="Info" writeTo="console" />` rule.

To see the exact API calls (full requests including headers and body), change this rule's `minlevel` in `nlog.config` from `Error` to `Info`:

```
<logger name="RSAHelper.WebAPI.Middleware.RequestLoggingMiddleware" minlevel="Info" writeTo="requestLogfile"
```

After saving the file, the change is picked up automatically (thanks to `autoReload="true"`), and subsequent requests start being written to `logs/log-requests.log`, e.g.:

```
2026-08-18 10:15:32.1234 | Info | HTTP Request started: GET http://localhost:5000/api/Authorization/57322960
```

> **NOTE:** With `minlevel="Info"`, the log captures the full request, including all headers and the body (e.g. the submitted invoice XML). Enable this level only temporarily, for diagnostics, and set `minlevel` back to `Error` afterwards. The `logs` directory should have restricted access, since logged requests may contain sensitive data.

Docker Configuration

The `nlog.config` file is baked into the image at build time and is not mounted as a volume in `docker-compose.yml` by default. For a `minlevel` change to survive container recreation, mount your own `nlog.config` the same way the `Data` directory is mounted:

```yaml
volumes:
  - ./Data:/app/Data:ro
  - ./nlog.config:/app/nlog.config:ro
```

For a quick, temporary change on an already running container (until it is next recreated), copy the file into the container — the change should be picked up automatically thanks to `autoReload="true"`; restart the container if it is not:

```
docker cp nlog.config rsahelper-api:/app/nlog.config
docker restart rsahelper-api  # if the change is not picked up automatically
```

3.12. Troubleshooting

PDF generation errors on Linux (installation without a container)

The state of PDF generation is reported by the `PdfGeneration` field in the **Health** endpoint response (section 4.9). When `"Available"` is `false`, the `Error` field carries the message that identifies the cause:

Message in the `Error` field	Cause	Solution
`Blink files are missing at .../runtimes/linux/native`	`PDFSettings:BlinkPath` is not set, or the configured directory contains no `chrome` file	section 3.2 — browser installation and path setting
`Failed to launch Base!` together with `chrome_crashpad_handler: --database is required` in the logs	the service account has no writable home directory	`Environment=HOME=/tmp` in the service file
`Failed to launch Base!` with no mention of `crashpad`	missing browser system libraries	`ldd /opt/google/chrome/chrome | grep "not found"` and install the missing packages
`Failed to create connection`	the browser starts but drops the control connection	check whether it starts on its own: `sudo -u www-data`

		`/opt/google/chrome/chrome --headless=new --no-sandbox --dump-dom about:blank`
`Access to the path '/tmp/chromium-rsahelper/...' is denied`	the browser working directory in `/tmp` belongs to another user, because the application was previously run from a different account	`PrivateTmp=yes` in the `[Service]` section of the service file, or a one-off `sudo rm -rf /tmp/chromium-rsahelper`

The check result is cached for 5 minutes — after fixing the configuration, restart the service instead of waiting for the refresh.

`503 Service Unavailable` with Apache + .NET setup

In this setup, a `503 Service Unavailable` error usually means Apache itself is working, but the .NET application is not responding. Apache is trying to proxy traffic to the application backend (for example on port `5000`), but the backend is either not listening or has crashed.

1. Check service status

First, verify whether the system service is actually running:

```
sudo systemctl status <service-name>.service
```

- if you see `Active: failed`, the application failed during startup
- if you see `Active: active (running)`, the application is running, but it may be listening on a different port than Apache expects

2. Check application logs

If the service is failing or unstable, inspect the latest systemd logs:

```
sudo journalctl -u <service-name>.service -n 50 --no-pager
```

Look for `Critical` or `Error` entries. Common causes include:

- invalid `appsettings.json` configuration
- missing system dependency
- incorrect application path or missing permissions

3. Check whether the application is listening

Verify whether the application is actually listening on the port expected by Apache:

```
sudo ss -tulpn | grep 5000
```

If the command returns nothing, the application is not listening on port `5000`. It may be using a different port such as `5001` or `8080`, or it may not have started at all.

4. Most common causes and fixes

Incorrect path in the `.service` file

Make sure `ExecStart` contains the full path to `dotnet` (usually `/usr/bin/dotnet`) and the correct path to the application `.dll`.

Missing permissions

If the service runs as `www-data`, ensure that user has access to the application directory:

```
sudo chown -R www-data:www-data /var/www/rsahelper
```

Environment or configuration issue

If logs indicate a configuration or dependency problem, temporarily switch the service to development environment in the `.service` file:

```
Environment=ASPNETCORE_ENVIRONMENT=Development
```

After each change to the `.service` file, run:

```
sudo systemctl daemon-reload
sudo systemctl restart <service-name>.service
```

5. Quick manual run

To see the startup error directly in the console, stop the service and run the application manually:

```
cd /var/www/rsahelper
dotnet RSAHelper.WebAPI.dll
```

If the application prints a message similar to:

```
Now listening on: http://localhost:5000
```

but the browser still shows `503`, the problem is most likely in the Apache proxy or reverse proxy configuration.

---

4. Usage

The application provides access to all functions through the **Swagger** interface, where all available functions can be called from any web browser.

Swagger Interface Access

Public Server:

- Release server: `https://rsahelper.unitsoftware.com.pl/swagger`
- Develop server: `https://testrsahelper.unitsoftware.com.pl/swagger`

Local Server:

- Address format: `http://<serverIPAddress>:<port>/swagger`
- Where:
  - `serverIPAddress` - IP or hostname where the application is installed
  - `port` - port on which the application is available (default 80 or 8080 for Docker)

Docker:

- `http://localhost:5000/swagger`

> **NOTE:** Description of individual functions along with parameters for the currently installed version is available through the Swagger interface and may differ from the information contained in this manual (the manual always contains a description of the latest application version).

4.1. Authorization

Base Path: `/api/Authorization`

Controller responsible for managing the list of authorized NIP numbers and configuration file operations.

4.1.1. Get List of All NIPs

Endpoint: `GET /api/Authorization`

Description: Returns a list of all NIP numbers supported by the application in JSON format.

Parameters: None

Response:

```
{
  "nips": [
    "5732296089",
    "9721121810",
    "1234567890"
  ]
}
```

Example call:

```
GET /api/Authorization
```

4.1.2. Get Text File with NIP List

Endpoint: `GET /api/Authorization/NipListFile`

Description: Returns a list of NIPs in text format (one NIP per line), useful for export.

Parameters: None

Response:

- Content-Type: `application/txt`
- File: `NipList.txt`
- Format: One NIP per line

Example response:

```
5732296089
9721121810
1234567890
```

4.1.3. Check if NIP is Supported

Endpoint: `GET /api/Authorization/{nip}`

Description: Checks whether the given NIP number is on the list of those supported by the application.

Parameters:

- `nip` (path, required) - NIP number to check (10 digits)

Response:

- Format: `boolean`

- `true` - NIP is supported
- `false` - NIP is not supported

Example call:

```
GET /api/Authorization/5732296089
```

Response:

```
true
```

4.1.4. Upload NIP File

Endpoint: `POST /api/Authorization/upload`

Description: Updates the configuration file with allowed NIP numbers. The file must contain a valid HMAC-SHA256 hash.

Parameters:

- Body (JSON):

```
{
  "nips": ["5732296089", "9721121810"],
  "version": "1.0",
  "createdAt": "2024-12-08T12:00:00Z",
  "description": "Production NIP list",
  "hash": "valid-hmac-sha256-hash-base64"
}
```

Fields:

- `nips` (array, required) - List of NIP numbers
- `version` (string, required) - File version
- `createdAt` (datetime, optional) - Creation date
- `description` (string, optional) - Description
- `hash` (string, required) - HMAC-SHA256 hash (computed using the key configured in the application)

Response:

```
{
  "message": "File loaded successfully.",
  "nipsCount": 2,
```

```
  "version": "1.0"
}
```

Error codes:

- `400 Bad Request` - Invalid file format or incorrect hash
- `500 Internal Server Error` - Server error during file write

4.2. Calculate

Base Path: `/api/Calculate`

Controller responsible for RSA encryption of authorization tokens.

4.2.1. RSA Token Encryption

Endpoint: `GET /api/Calculate/{serverType}/{inputString}`

Description: Method that encrypts a string using RSA algorithm with KSeF public key. Used for encrypting tokens during KSeF authorization.

Parameters:

- `serverType` (path, required) - KSeF environment type:
  - `prod` - Production
  - `demo` - Demonstration
  - `test` - Test
- `inputString` (path, required) - String to encrypt (token)
- `nip` (query, required) - Entity's NIP (authorization parameter)

Operation mechanism:

- For `prod`, `test` and `demo`: Dynamically fetches public keys from KSeF API using `CryptographyService`

Response:

- Status: `200 OK`
- Format: `string` (Base64)
- Content: Encrypted string encoded in Base64

Error codes:

- `401 Unauthorized` - Missing NIP parameter
- `401 Unauthorized` - NIP is not supported by the application
- `500 Internal Server Error` - Server error

Example call:

```
GET /api/Calculate/test/myToken123?nip=5732296089
```

Response:

```
SGVsbG8gV29ybGQhIFRoaXMgaXMgYW4gZXhhbXBsZSBvZiBSU0EgZW5jcnlwdGVkIGRhdGE=
```

4.3. Invoices

Base Path: `/api/Invoices`

Controller responsible for generating cryptographic data and encrypting invoices for KSeF 2.0 (interactive session).

4.3.1. Generate Encryption Data

Endpoint: `GET /api/Invoices/GenerateEncryptionData/{serverType}`

Description: Generates AES-256 key, initialization vector (IV), and encrypted symmetric key needed for:

1. Initializing interactive session in KSeF 2.0
2. Encrypting invoice before sending

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "encryptedSymmetricKey": "e+SdWZTeqd3aeSNyMay16Tag/JTQRjNxXyJm3TRxTTHGu1dUcL9MoqZ3TZazsCBc...",
  "initializationVector": "9CrrDfN8jEdZHAbLpodNw==",
  "symmetricKey": "ZAb5UNVeDh6DoZQvuPmMlDFfB39IqREEbFywQ/VUjSM="
}
```

Response field descriptions:

- `encryptedSymmetricKey` - Encrypted symmetric key (RSA). Must be passed in KSeF interactive session initialization request.
- `initializationVector` - AES initialization vector (IV). Must be used when encrypting invoice and passed in session initialization.
- `symmetricKey` - AES-256 symmetric key in plain form. Must be used when encrypting invoice.

Example call:

```
GET /api/Invoices/GenerateEncryptionData/test
```

Usage flow:

1. Call this endpoint to receive cryptographic data
2. Use `encryptedSymmetricKey` and `initializationVector` to initialize KSeF session
3. Use `symmetricKey` and `initializationVector` in `fromBase64` endpoint to encrypt invoice

4.3.2. Encrypt Invoice and Generate Metadata

Endpoint: `POST /api/Invoices/fromBase64/{serverType}`

Description: Encrypts invoice in XML format using AES-256-CBC algorithm and returns metadata required for sending to KSeF (hashes, sizes, encrypted content).

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`
- `nip` (query, required) - Entity's NIP (authorization parameter)
- `cipherKey` (query, required) - AES key encoded in Base64 (`symmetricKey` field from `GenerateEncryptionData`)
- `cipherIv` (query, required) - IV vector encoded in Base64 (`initializationVector` field from `GenerateEncryptionData`)

Body:

- Content-Type: `text/plain`
- Content: Invoice XML file encoded in Base64

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "invoiceSize": 1234,
  "encryptedInvoiceHash": "d7iTpRuPx1IS/JPTVXTgIwmQRtYvXCoIqNM2rNOzgMw=",
  "encryptedInvoiceSize": 1456,
  "encryptedInvoiceContent": "HpGGgaC0JS21I643ZALb3gA=..."
}
```

Response field descriptions:

- `invoiceHash` - SHA-256 hash of original (unencrypted) invoice
- `invoiceSize` - Size of original invoice in bytes
- `encryptedInvoiceHash` - SHA-256 hash of encrypted invoice

- `encryptedInvoiceSize` - Size of encrypted invoice in bytes
- `encryptedInvoiceContent` - Encrypted invoice encoded in Base64

Error codes:

- `400 Bad Request` - Missing NIP parameter or invalid input data format
- `403 Forbidden` - NIP is not supported by the application
- `500 Internal Server Error` - Invoice processing error

Special features:

- Automatic BOM (Byte Order Mark) removal from XML file
- Support for various UTF-8 encodings

Example call:

```
POST /api/Invoices/fromBase64/test?nip=5732296089&cipherKey=ZAb5UNVeD...&cipherIv=9CrrDfN8jEdZH...
Content-Type: text/plain

PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPEZha3R1cmE+Li4uPC9GYWt0dXJhPg==
```

Full KSeF 2.0 invoice submission process:

1. Call `GET /api/Invoices/GenerateEncryptionData/{serverType}` → receive cryptographic data
2. Initialize interactive session in KSeF using `encryptedSymmetricKey` and `initializationVector`
3. Call `POST /api/Invoices/fromBase64/{serverType}` with keys from step 1 → receive metadata
4. Send encrypted invoice to KSeF using data from step 3

4.3.3. Encrypt ZIP Package and Generate Metadata

Endpoint: `POST /api/Invoices/package/{serverType}`

Description: Encrypts ZIP package using AES-256-CBC algorithm and returns metadata required for sending invoice package to KSeF (hashes, sizes, encrypted content). Used for sending multiple invoices in a single package.

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`

Body:

- Content-Type: `application/json`

```
{
  "inputZipData": "UEsDBBQAAAAIAA...",
  "nip": "5732296089",
  "cipherKey": "ZAb5UNVeDh6DoZQvuPmMlDFfB39IqREEbFywQ/VUjSM=",
```

```
  "cipherIv": "9CrrDfN8jEdZHAbLpodNw=="
}
```

Body fields:

- `inputZipData` (string, required) - ZIP package encoded in Base64
- `nip` (string, required) - Entity's NIP (authorization parameter)
- `cipherKey` (string, required) - AES key encoded in Base64 (`symmetricKey` field from `GenerateEncryptionData`)
- `cipherIv` (string, required) - IV vector encoded in Base64 (`initializationVector` field from `GenerateEncryptionData`)

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "packageHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "packageSize": 45678,
  "encryptedPackageHash": "d7iTpRuPx1IS/JPTVXTgIwmQRtYvXCoIqNM2rNOzgMw=",
  "encryptedPackageSize": 45696,
  "encryptedPackageContent": "HpGGgaC0JS21I643ZALb3gA=..."
}
```

Response field descriptions:

- `packageHash` - SHA-256 hash of original (unencrypted) ZIP package
- `packageSize` - Size of original package in bytes
- `encryptedPackageHash` - SHA-256 hash of encrypted package
- `encryptedPackageSize` - Size of encrypted package in bytes
- `encryptedPackageContent` - Encrypted package encoded in Base64

Error codes:

- `400 Bad Request` - Missing NIP parameter or invalid input data format
- `403 Forbidden` - NIP is not supported by the application
- `500 Internal Server Error` - Package processing error

Example call:

```
POST /api/Invoices/package/test
Content-Type: application/json

{
  "inputZipData": "UEsDBBQAAAAIAA...",
  "nip": "5732296089",
  "cipherKey": "ZAb5UNVeDh6DoZQvuPmMlDFfB39IqREEbFywQ/VUjSM=",
```

```
  "cipherIv": "9CrrDfN8jEdZHAbLpodNw=="
}
```

Invoice package submission process for KSeF 2.0:

1. Call `GET /api/Invoices/GenerateEncryptionData/{serverType}` → receive cryptographic data
2. Initialize interactive session in KSeF using `encryptedSymmetricKey` and `initializationVector`
3. Prepare ZIP package containing XML invoices
4. Call `POST /api/Invoices/package/{serverType}` with keys from step 1 → receive metadata
5. Send encrypted package to KSeF using data from step 4

4.4. ValidateXml

Base Path: `/api/ValidateXml`

Controller responsible for validating XML documents against JPK XSD schemas.

4.4.1. XML Validation (from Base64)

Endpoint: `POST /api/ValidateXml/fromBase64`

Description: Validates XML document encoded in Base64 against the appropriate JPK XSD schema. Schema is automatically detected based on document content.

Parameters:

- Body (text/plain): XML document encoded in Base64

Content-Type: `text/plain`

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "isValid": true,
  "messages": [
    {
      "severity": "Warning",
      "message": "Warning about incomplete data"
    }
  ]
}
```

Response structure:

- `isValid` (boolean) - `true` if document is valid, `false` otherwise

- `messages` (array) - List of validation messages
  - `severity` - Severity level: `Error`, `Warning`
  - `message` - Message content with line number (if available)

Example call:

```
POST /api/ValidateXml/fromBase64
Content-Type: text/plain

PD94bWwgdmVyc2lvbj0iMS4wIj8+CjxGYWt0dXJhPi4uLjwvRmFrdHVyYT4=
```

4.4.2. XML Validation (directly)

Endpoint: `POST /api/ValidateXml`

Description: Validates XML document against the appropriate JPK XSD schema. Schema is automatically detected based on document content.

Parameters:

- Body (application/xml): XML document in plain form

Content-Type: `application/xml`

Response: Identical to 4.4.1

Example call:

```
POST /api/ValidateXml
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Faktura xmlns="http://crd.gov.pl/wzor/2023/06/29/12648/">
  ...
</Faktura>
```

Supported structures:

- FA(1)_v1-0E - Structured invoice version 1
- FA(2)_v1-0E - Structured invoice version 2
- FA(3)_v1-0E - Structured invoice version 3
- JPK_V7M(2) - JPK VAT with declaration

Schema detection mechanism: The application automatically detects the appropriate XSD schema based on:

1. Form code
2. System code

3. Schema version

4.5. ConvertToHtml

Base Path: `/api/ConvertToHtml`

Controller responsible for converting e-invoice XML documents to HTML format using XSLT transformation.

4.5.1. XML to HTML Conversion (from Base64)

Endpoint: `POST /api/ConvertToHtml/fromBase64`

Description: Converts e-invoice document in XML format (Base64 encoded) to HTML representation. XSLT transformation is automatically selected based on document type.

Parameters:

- `lang` (query, optional) - XSLT template language
  - Default value: `pl`
  - Available values: `pl` (Polish), `en` (English)
- `fontSize` (query, optional) - Base document font size in px (applies to the alternative FA(3) template)
  - When omitted, `PDFSettings.DefaultFontSize` is used, then the template default (`26`)
  - A non-positive value is ignored
- Body (text/plain): XML document encoded in Base64

Content-Type: `text/plain`

Response:

- Status: `200 OK`
- Content-Type: `text/plain`
- Content: HTML document encoded in Base64

Example call:

```
POST /api/ConvertToHtml/fromBase64?lang=en
Content-Type: text/plain

PD94bWwgdmVyc2lvbj0iMS4wIj8+CjxGYWt0dXJhPi4uLjwvRmFrdHVyYT4=
```

4.5.2. XML to HTML Conversion (directly)

Endpoint: `POST /api/ConvertToHtml`

Description: Converts e-invoice document in XML format to HTML representation.

Parameters:

- `lang` (query, optional) - XSLT template language
  - Default value: `pl`
  - Available values: `pl` (Polish), `en` (English)
- `fontSize` (query, optional) - Base document font size in px (applies to the alternative FA(3) template)
  - When omitted, `PDFSettings.DefaultFontSize` is used, then the template default (`26`)
  - A non-positive value is ignored
- Body (application/xml): XML document in plain form

Content-Type: `application/xml`

Response:

- Status: `200 OK`
- Content-Type: `text/html`
- Content: HTML document

Example call:

```
POST /api/ConvertToHtml?lang=en
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Faktura xmlns="http://crd.gov.pl/wzor/2023/06/29/12648/">
  ...
</Faktura>
```

Supported document types:

- FA(1) - Structured invoice version 1
- FA(2) - Structured invoice version 2
- FA(3) - Structured invoice version 3
- JPK_V7M(2) - JPK VAT with declaration

Operation mechanism:

1. Detect document type
2. Select appropriate XSLT template
   - For `lang=pl` : template without suffix is used (e.g., `FA(3)_v1-0E.xsl`)
   - For `lang=en` : template with `_en` suffix is used (e.g., `FA(3)_v1-0E_en.xsl`)
3. Transform XML → HTML

4.6. ConvertToPdf

Base Path: `/api/ConvertToPdf`

Controller responsible for converting e-invoice XML documents to PDF format.

4.6.1. XML to PDF Conversion (from Base64)

Endpoint: `POST /api/ConvertToPdf/fromBase64`

Description: Converts e-invoice document in XML format (Base64 encoded) to PDF document. Conversion is done through XML → HTML → PDF transformation.

Parameters:

- `fileName` (query, optional) - Name of returned PDF file
  - Default value: `eInvoice.pdf`
- `lang` (query, optional) - XSLT template language
  - Default value: `pl`
  - Available values: `pl` (Polish), `en` (English)
- `fontSize` (query, optional) - Base document font size in px (applies to the alternative FA(3) template)
  - When omitted, `PDFSettings.DefaultFontSize` is used, then the template default (`26`)
  - A non-positive value is ignored
- Body (application/xml): XML document encoded in Base64

Content-Type: `application/xml`

Response:

- Status: `200 OK`
- Content-Type: `application/pdf`
- Headers: `Content-Disposition: attachment; filename="<fileName>"`
- Content: PDF file

Example call:

```
POST /api/ConvertToPdf/fromBase64?fileName=invoice_2024.pdf&lang=en
Content-Type: application/xml

PD94bWwgdmVyc2lvbj0iMS4wIj8+CjxGYWt0dXJhPi4uLjwvRmFrdHVyYT4=
```

4.6.2. XML to PDF Conversion (directly)

Endpoint: `POST /api/ConvertToPdf`

Description: Converts e-invoice document in XML format to PDF document.

Parameters:

- `fileName` (query, optional) - Name of returned PDF file
  - Default value: `eInvoice.pdf`
- `lang` (query, optional) - XSLT template language
  - Default value: `pl`
  - Available values: `pl` (Polish), `en` (English)

- `fontSize` (query, optional) - Base document font size in px (applies to the alternative FA(3) template)
  - When omitted, `PDFSettings.DefaultFontSize` is used, then the template default (`26`)
  - A non-positive value is ignored
- Body (application/xml): XML document in plain form

Content-Type: `application/xml`

Response: Identical to 4.6.1

Example call:

```
POST /api/ConvertToPdf?fileName=invoice.pdf&lang=en
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Faktura xmlns="http://crd.gov.pl/wzor/2023/06/29/12648/">
  ...
</Faktura>
```

Supported document types:

- FA(1) - Structured invoice version 1
- FA(2) - Structured invoice version 2
- FA(3) - Structured invoice version 3
- JPK_V7M(2) - JPK VAT with declaration

Operation mechanism:

1. Detect document type
2. Transform XML → HTML (XSLT)
3. Convert HTML → PDF

License requirements: PDF conversion uses Syncfusion.HtmlToPdfConverter library, which requires a license (license key is embedded in the application).

4.7. QrCode

Base Path: `/api/QrCode`

Controller responsible for generating QR codes for invoice verification in the KSeF system.

General Information about QR Codes

The application generates **two types of QR codes** according to KSeF specification:

1. **QR Code #1 (Online Verification)** - Link to invoice verification in KSeF system
   - Endpoints: `get` , `post` , `postContent`
   - Contains: Verification URL with NIP, date and invoice hash

- Usage: Invoice verification by recipients

2. **QR Code #2 (Certificate Verification)** - Link with digital signature
   - Endpoints: `postVer`, `postVerContent`
   - Contains: URL + ECDSA signature from certificate
   - Usage: Advanced verification with certificate

Technical parameters:

- Image format: BMP (default)
- Size: 100-2000 pixels (default 270). For sharp QR codes, a multiple of 45 is recommended; otherwise, the working size is reduced to the nearest lower multiple of 45
- ECC Level: M (Medium Error Correction)
- Encoding: UTF-8
- Renderer: `SkiaSharp` (default) or `ImageSharp`; selected through the `QrCode.Renderer` setting in `appsettings.json`

4.7.1. Generate QR Code #1 (GET)

Endpoint: `GET /api/QrCode/get/{serverType}/{nip}/{issueDate}`

Description: Generates QR code containing link to online invoice verification in KSeF system.

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`
- `nip` (path, required) - Invoice issuer's NIP (10 digits)
- `issueDate` (path, required) - Invoice issue date in format `dd-MM-yyyy`
- `invoiceHash` (query, required) - Invoice SHA-256 hash (from Invoices endpoint, `invoiceHash` field)
- `ksefNo` (query, optional) - KSeF number to display under QR code as label, can also be any other text, e.g. OFFLINE
- `size` (query, optional) - Code size in pixels
  - Default: 270
  - Range: 100-2000
  - Recommendation: use a multiple of 45 for sharp edges
  - If not a multiple of 45: working size is reduced to the nearest lower multiple of 45

Response:

- Status: `200 OK`
- Content-Type: `image/bmp`
- Content: QR code image in BMP format

Error codes:

- `400 Bad Request` - QR code size outside 100-2000 range

Example call:

```
GET /api/QrCode/get/test/5732296089/15-12-2024?invoiceHash=abc123def456...&ksefNo=KSeF-123456&size=400
```

4.7.2. Generate QR Code #1 (POST)

Endpoint: `POST /api/QrCode/post/{serverType}`

Description: Generates QR code #1 (online verification). Alternative to GET method, used when parameters are too long or POST is preferred.

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`
- Body (JSON):

```
{
  "nip": "5732296089",
  "issueDate": "15-12-2024",
  "invoiceHash": "abc123def456...",
  "ksefNo": "KSeF-123456",
  "size": 400
}
```

Body fields:

- `nip` (string, required) - Invoice issuer's NIP (10 digits)
- `issueDate` (string, required) - Issue date in format `dd-MM-yyyy`
- `invoiceHash` (string, required) - Invoice SHA-256 hash
- `ksefNo` (string, optional) - KSeF number to display under QR code as label, can also be any other text, e.g. OFFLINE
- `size` (integer, optional) - Size in pixels (100-2000, default 270); for sharp edges use a multiple of 45, otherwise the working size is reduced to the nearest lower multiple of 45

Response:

- Status: `200 OK`
- Content-Type: `image/bmp`
- Content: QR code image

Example call:

```
POST /api/QrCode/post/test
Content-Type: application/json

{
  "nip": "5732296089",
  "issueDate": "15-12-2024",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
```

```
  "ksefNo": "KSeF-123456-789",
  "size": 400
}
```

4.7.3. Generate QR Code #1 Content

Endpoint: `POST /api/QrCode/postContent/{serverType}`

Description: Returns the string (URL) that should be placed in QR code #1. Useful when you want to generate QR code in your own system.

Parameters:

- `serverType` (path, required) - Environment type
- Body (JSON): Identical to 4.7.2

Response:

- Status: `200 OK`
- Content-Type: `text/plain`
- Content: Verification URL

Example response:

```
https://ksef-test.mf.gov.pl/web/verify/5732296089/2024-12-15/QSS1NTy1DruETws1yrWJJt%2FjlTFFJ%2FDP2%2FFasvuAqH70%
```

Example call:

```
POST /api/QrCode/postContent/test
Content-Type: application/json

{
  "nip": "5732296089",
  "issueDate": "15-12-2024",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70="
}
```

4.7.4. Generate QR Code #2 (with certificate)

Endpoint: `POST /api/QrCode/postVer/{serverType}`

Description: Generates QR code #2 containing verification link with digital signature (ECDSA) based on certificate. Provides higher security level.

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`
- Body (JSON):

Content-Type: `application/json`

JSON fields:

- `nip` (string, required) - Invoice issuer's NIP (10 digits)
- `invoiceHash` (string, required) - Invoice hash (Base64)
- `certificate` (string, required) - X.509 certificate in PEM format, Base64 encoded
- `password` (string, required) - Password for private key in certificate, Base64 encoded
- `privateKey` (string, optional) - ECDSA private key in PEM format, Base64 encoded (if not embedded in certificate)
- `size` (integer, optional) - QR code size in pixels (100-2000, default 270); for sharp edges use a multiple of 45, otherwise the working size is reduced to the nearest lower multiple of 45

Response:

- Status: `200 OK`
- Content-Type: `image/bmp`
- Content: QR code with "CERTYFIKAT" label

Example call:

```
POST /api/QrCode/postVer/test
Content-Type: application/json

{
  "nip": "5732296089",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "certificate": "MIIDXTCCAkW...",
  "password": "aGFzbG8xMjM=",
  "size": 400
}
```

4.7.5. Generate QR Code #2 Content

Endpoint: `POST /api/QrCode/postVerContent/{serverType}`

Description: Returns the string (URL with signature) that should be placed in QR code #2. Useful when you want to generate QR code in your own system.

Parameters:

- `serverType` (path, required) - Environment type: `prod`, `demo`, `test`
- Body (JSON): Identical to 4.7.4

Content-Type: `application/json`

Response:

- Status: `200 OK`
- Content-Type: `text/plain`
- Content: Verification URL with ECDSA signature

Example call:

```
POST /api/QrCode/postVerContent/test
Content-Type: application/json

{
  "nip": "5732296089",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "certificate": "MIIDXTCCAkW...",
  "password": "aGFzbG8xMjM="
}
```

4.8. Version

Base Path: `/api/Version`

Controller responsible for application version information.

4.8.1. Get Application Version

Endpoint: `GET /api/Version`

Description: Returns application version and build date. This is an informational endpoint. Use the **Health** endpoint (section 4.9) to check application status.

Parameters: None

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "versionId": "1.2.0",
  "buildDate": "2024-12-08T12:30:23.705232"
}
```

Response fields:

- `versionId` - Application version (Semantic Versioning)
- `buildDate` - Application build date and time (ISO 8601)

Example call:

```
GET /api/Version
```

Addresses for different environments:

- Test: `https://testrsahelper.unitsoftware.com.pl/api/Version`
- Local: `http://localhost:8080/api/Version`

4.9. Health

Base Path: `/health`

Controller responsible for monitoring application status (used by Docker HEALTHCHECK).

4.9.1. Health Check

Endpoint: `GET /health`

Description: Returns detailed application and dependency status. Used by Docker HEALTHCHECK to monitor container availability. The HTTP code remains `200 OK`; the logical state should be read from the `Status` field.

Parameters: None

Response:

- Status: `200 OK`
- Format: JSON


```
{
  "Status": "Healthy",
  "Timestamp": "2026-04-16T12:00:00.0000000Z",
  "OfflineMode": false,
  "MinistryServers": [
    {
      "Url": "https://api-demo.ksef.mf.gov.pl/api/v2",
      "Available": true,
      "StatusCode": 200,
      "ResponseTimeMs": 124,
      "Error": null
    }
  ],
  "LocalCertificates": [],
  "AllowedNips": {
    "Available": true,
    "CheckedAt": "2026-04-16T12:00:00.0000000Z",
    "Count": 42,
    "Error": null
  },
  "PdfGeneration": {
    "Available": true,
    "CheckedAt": "2026-04-16T12:00:00.0000000Z",
    "Error": null
  },
```

```
  "QrGeneration": {
    "Available": true,
    "CheckedAt": "2026-04-16T12:00:00.0000000Z",
    "Error": null
  }
}
```


Response fields:

- `Status` - Application state: `Healthy` or `Degradated`
- `Timestamp` - UTC time when the response was generated (ISO 8601)
- `OfflineMode` - Indicates whether the application is running in offline mode
- `MinistryServers` - Status list of external servers checked by the application
- `MinistryServers[].Url` - Checked server address
- `MinistryServers[].Available` - Whether the server responded successfully
- `MinistryServers[].StatusCode` - HTTP status code returned by the server, if available
- `MinistryServers[].ResponseTimeMs` - Response time in milliseconds
- `MinistryServers[].Error` - Error message or `null`
- `LocalCertificates` - Status of local certificate files for `DEMO`, `TEST`, and `PROD`; in online mode this array is usually empty
- `LocalCertificates[].Environment` - Environment name
- `LocalCertificates[].Available` - Whether the certificate file is available and readable
- `LocalCertificates[].HasValidCertificates` - Whether at least one currently valid certificate exists
- `LocalCertificates[].TotalCount` - Total number of certificates in the file
- `LocalCertificates[].ValidCount` - Number of currently valid certificates
- `LocalCertificates[].Error` - Diagnostic error code/message or `null`
- `AllowedNips` - Status of the loaded allowed NIP list
- `AllowedNips.Available` - Whether the NIP list is available and not empty
- `AllowedNips.CheckedAt` - Time when the check was performed
- `AllowedNips.Count` - Number of entries in the list
- `AllowedNips.Error` - Error message or `null`
- `PdfGeneration` - PDF generation subsystem status
- `PdfGeneration.Available` - Whether the PDF probe succeeded
- `PdfGeneration.CheckedAt` - Time of the last check
- `PdfGeneration.Error` - Error message or `null`
- `QrGeneration` - QR generation subsystem status
- `QrGeneration.Available` - Whether the QR probe succeeded
- `QrGeneration.CheckedAt` - Time of the last check
- `QrGeneration.Error` - Error message or `null`

Notes:

- In online mode, `Status` depends on ministry server availability, the allowed NIP list, PDF generation, and QR generation
- In offline mode, `Status` depends on local certificate validity, the allowed NIP list, PDF generation, and QR generation
- `MinistryServers` information is still returned in offline mode, but it does not determine the final state there

Example call:

```
GET /health
```

Docker HEALTHCHECK configuration:

```
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD curl -f http://localhost:8080/health || exit 1
```

Usage with Docker:

```
# Check container status
docker ps

# Check health check logs
docker inspect --format='{{.State.Health.Status}}' rsahelper-api
```

---

5. Application Versions

Version History

Version 1.2.7 (current)

- Added a configurable QR code display size (`qrCodeSize` parameter and `PDFSettings.DefaultQrCodeSize`)
- Added document-language-aware KSeF number label (“Numer KSeF” / “KSeF Number” depending on `lang`)
- Introduced a new logging configuration (NLog) with daily file rotation and 7-day log retention
- Added the ability to log WebAPI call details (`RequestLoggingMiddleware`)

Version 1.2.6

- Added KSeF 2.0 client certificate authentication
- Added a configurable document font size for the alternative FA(3) template (`fontSize` parameter and `PDFSettings.DefaultFontSize`)
- Added a new ImageSharp-based QR code renderer as an alternative to SkiaSharp (`QrCode.Renderer`)
- Extended the `/health` endpoint with detailed readiness checks for PDF generation, QR code generation and the allowed NIPs list
- Added support for displaying the KSeF number on invoices in the alternative English template
- Fixed PDF and QR code generation in Linux/Docker containers
- Fixed missing JST/GV fields in the alternative FA(3) template

Version 1.2.5

- Added a new PDF generation engine
- Additional PDF generation settings
- Additional QR code generation settings (font family and font size)

Version 1.2.4

- Added offline mode enabling operation without internet access
- Three operating modes: Online, Hybrid (with network fallback), Strict Offline
- Local KSeF certificate support (loading from JSON files instead of API)
- Local XSD/XSL schema resolution (without crd.gov.pl access)
- Added support for setting default language in document visualization

Version 1.2.3

- Added endpoint for encrypting invoice ZIP packages (`POST /api/Invoices/package/{serverType}`)
- Enabled sending multiple invoices in a single package to KSeF

Version 1.2.2

- Fixed PDF generation on Linux systems
- Added automatic system Chromium detection
- Added cross-platform support (Windows/Linux) for PDF conversion
- Updated documentation with Chromium requirements

Version 1.2.1

- Library updates
- Proxy implementation

Version 1.2.0

- Updated API documentation
- New QR Code endpoints (new structure)
- Docker support
- Upgrade to .NET 10.0
- New Authorization endpoints: `upload` , `NipListFile`
- Added `/health` endpoint for Docker HEALTHCHECK

Version 1.1.0

- Support for KSeF 2.0 (new Invoices endpoint group)
- AES-256-CBC encryption for invoices
- Cryptographic data generation (GenerateEncryptionData)
- FA(3)_v1-0E validation

Version 1.0.6

- Added QR code generation methods (getHash, getImage)

- Support for ECCLevel M in QR codes

Version 1.0.5

- E-invoice visualization in PDF format
- Removed `documentType` parameter from ConvertToHtml method (automatic detection)
- Integration with Syncfusion HtmlToPdfConverter

Version 1.0.4

- First production version
- RSA algorithm support (authorization in KSeF system)
- XML to HTML conversion
- FA(1)_v1-0E and FA(2)_v1-0E file validation

Licenses

The application uses the following commercial libraries:

- **Syncfusion HtmlToPdfConverter** - License embedded in application

---

Last documentation update: 2026-08-15

Documentation version: 1.4

Compatible with application version: 1.2.7+