RSA Helper Installation Guide

Docker & Podman Setup for Linux and Windows

Version: 1.2.5 **Last Updated:** February 11, 2026

Table of Contents

Overview

RSA Helper is a WebAPI application that supports KSeF (Polish e-invoicing system) processes. This guide covers containerized deployment using either Docker or Podman on Linux and Windows.

Key Information:

- Application runs on internal port **8080**
- Default external port mapping: **5000**
- Requires configuration file: `allowed-nips.json`
- Health check endpoint: `/health`
- Swagger UI: `/swagger`

GitHub Container Registry Authentication

RSA Helper image is hosted on GitHub Container Registry (ghcr.io), you'll need to authenticate before pulling the image.

Prerequisites

1. **GitHub Account** with access to the repository (GitHub account is free and can be easly created at https://github.com/signup)
2. **Personal Access Token (PAT)** with `read:packages` scope

Creating a GitHub Personal Access Token

1. Go to GitHub: https://github.com/settings/tokens
2. Click **"Generate new token" → "Generate new token (classic)"**
3. Configure the token:
   - **Note:** `RSAHelper Docker Access`
   - **Expiration:** Choose appropriate duration (e.g., 90 days, 1 year, or no expiration)
   - **Scopes:** Select `read:packages` (required to pull images)
4. Click **"Generate token"**
5. **IMPORTANT:** Copy the token immediately - you won't be able to see it again!

Get access to package

After creating account please send request to grzegorz.giewon@uptime-erp.com with username and email of your account.

Authentication Methods

Method 1: Docker Login (Recommended)

Linux:

```
# Login to GitHub Container Registry
echo "YOUR_GITHUB_TOKEN" | docker login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin

# Verify login
docker login ghcr.io
```

Windows PowerShell:

```
# Set variables
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_USERNAME = "YOUR_GITHUB_USERNAME"

# Login
$env:GITHUB_TOKEN | docker login ghcr.io -u $env:GITHUB_USERNAME --password-stdin

# Clear token from environment for security
Remove-Item Env:\GITHUB_TOKEN
```

Success message:

```
Login Succeeded
```

Method 2: Podman Login

Linux:

```
# Login to GitHub Container Registry
echo "YOUR_GITHUB_TOKEN" | podman login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin

# Verify login
podman login ghcr.io
```

Windows PowerShell:

```
# Login
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_TOKEN | podman login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin
Remove-Item Env:\GITHUB_TOKEN
```

Method 3: Using Credentials File (Advanced)

Linux:

```
# Docker credentials are stored in:
~/.docker/config.json

# Podman credentials are stored in:
~/.config/containers/auth.json
```

Windows:

```
# Docker credentials location:
C:\Users\YourUsername\.docker\config.json

# Podman credentials location:
C:\Users\YourUsername\.config\containers\auth.json
```

Updating docker-compose.yml for GitHub Container Registry

The RSA Helper image is hosted on GitHub Container Registry at:

```
ghcr.io/ggiewon/rsahelper-webapi:latest
```

Update the image reference in your `docker-compose.yml`:

```yaml
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    # or for specific version (when available):
    # image: ghcr.io/ggiewon/rsahelper-webapi:1.2.0
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    # ... rest of configuration
```

Complete example:

```yaml
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    volumes:
      - ./Data:/app/Data:ro
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
```

```
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
    networks:
      - rsahelper-network

networks:
  rsahelper-network:
    driver: bridge
```

Pulling the Image After Authentication

Docker:

```
# Pull the RSA Helper image
docker pull ghcr.io/ggiewon/rsahelper-webapi:latest

# Or use docker-compose (will use authenticated session)
docker-compose pull
```

Podman:

```
# Pull the RSA Helper image
podman pull ghcr.io/ggiewon/rsahelper-webapi:latest

# Or use podman-compose
podman-compose pull
```

Troubleshooting Authentication

Error: “unauthorized: authentication required”

Solution:

```
# Verify you're logged in
docker login ghcr.io
# or
podman login ghcr.io

# If not logged in, authenticate again with your token
echo "YOUR_TOKEN" | docker login ghcr.io -u YOUR_USERNAME --password-stdin
```

Error: “denied: permission_denied”

Possible causes:

1. Token doesn't have `read:packages` permission
2. You don't have access to the repository
3. Repository is private and you're not a collaborator

Solution:

- Generate a new token with correct permissions
- Contact repository owner for access
- Verify repository visibility settings

Error: "Error response from daemon: Get https://ghcr.io/v2/: denied"

Solution (Linux):

```
# Clear old credentials
rm ~/.docker/config.json

# Login again
docker login ghcr.io
```

Solution (Windows):

```
# Clear credentials
Remove-Item "$env:USERPROFILE\.docker\config.json"

# Login again
docker login ghcr.io
```

Verify Authentication Status

```
# Docker
cat ~/.docker/config.json | grep ghcr.io

# Podman (Linux)
cat ~/.config/containers/auth.json | grep ghcr.io

# Windows PowerShell (Docker)
Get-Content "$env:USERPROFILE\.docker\config.json" | Select-String "ghcr.io"
```

Security Best Practices

1. **Never commit tokens to Git**

```
# Add to .gitignore
echo "*.token" >> .gitignore
echo ".env" >> .gitignore
```

2. **Use environment variables**

```
# Linux - add to ~/.bashrc or ~/.profile
export GITHUB_TOKEN="your_token_here"

# Windows - use User Environment Variables
# Settings → System → About → Advanced system settings → Environment Variables
```

3. **Rotate tokens regularly**

   - Set expiration dates on tokens
   - Generate new tokens every 90 days
   - Revoke old tokens after rotation

4. **Use minimal permissions**

   - Only grant `read:packages` for pulling images
   - Don't use tokens with `write:packages` or `delete:packages` unless necessary

5. **Store tokens securely**

```
# Linux - use secret management tools
# Store in password manager or system keyring

# Example using pass (password manager)
pass insert github/docker-token
pass show github/docker-token | docker login ghcr.io -u username --password-stdin
```

Logout

When finished or to clear credentials:

```
# Docker
docker logout ghcr.io

# Podman
podman logout ghcr.io
```

Alternative: Using Image as .tar File

If you cannot authenticate or prefer offline distribution:

1. **On machine with access:**

```
# Pull and save image
docker pull ghcr.io/ggiewon/rsahelper-webapi:latest
docker save ghcr.io/ggiewon/rsahelper-webapi:latest -o rsahelper-webapi-develop.tar

# Compress (optional)
gzip rsahelper-webapi-develop.tar
```

2. **Transfer file to target machine** (USB, SCP, etc.)

3. **On target machine:**

```
# Load image
docker load -i rsahelper-webapi-develop.tar
# or if compressed:
docker load -i rsahelper-webapi-develop.tar.gz

# Tag image if needed (only if using local name in docker-compose.yml)
docker tag ghcr.io/ggiewon/rsahelper-webapi:latest rsahelper-webapi:latest
```

Linux Installation

Prerequisites

Update your system first:

```
sudo apt-get update
sudo apt-get upgrade -y
```

Linux Docker

Step 1: Install Docker and Docker Compose

```
# Install Docker
sudo apt-get install -y docker.io docker-compose

# Start Docker service
sudo systemctl start docker
sudo systemctl enable docker

# Add current user to docker group (to avoid using sudo)
sudo usermod -aG docker $USER
```

```
# IMPORTANT: Log out and log back in for group changes to take effect
```

Step 2: Verify Installation

After logging back in:

```
docker --version
docker-compose --version
```

Expected output:

```
Docker version 24.x.x, build xxxxxxx
docker-compose version 1.29.x, build xxxxxxx
```

Step 3: Create Working Directory

```
# Create application directory
mkdir -p ~/RSAHelper
cd ~/RSAHelper
```

Step 4: Create docker-compose.yml File

```
nano docker-compose.yml
```

Paste the following content:

```
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    volumes:
      - ./Data:/app/Data:ro
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
    networks:
```

```
      - rsahelper-network

networks:
  rsahelper-network:
    driver: bridge
```

Save and exit (Ctrl+X, then Y, then Enter).

Step 5: Create Data Directory and Configuration File

```
# Create Data directory
mkdir -p Data

# Copy configuration file (if you have it)
cp ~/Downloads/allowed-nips.json Data/

# Or create it manually
nano Data/allowed-nips.json
```

Set proper permissions:

```
chmod 644 Data/allowed-nips.json
chmod 755 Data/
```

Step 5.1 (optional) Set proxy settings

```
# Create it manually

nano Data/appsettings.additional.json
```

with content:

```
{
  "KSeFClientOptions": {
    "Proxy": {
      "Enabled": true,
      "Address": "<proxy url>",
      "Username": "<optional username to auth to proxy>",
      "Password": "<optional password to auth to proxy>"
    }
  }
}
```

Set proper permissions:

```
chmod 644 Data/appsettings.additional.json
```

Step 6: Verify Directory Structure

```
tree -L 2
# Or without tree:
ls -la
ls -la Data/
```

Expected structure:

```
RSAHelper/
├── docker-compose.yml
└── Data/
    └── allowed-nips.json
    └── appsettings.additional.json (optional)
```

Step 7: Load Docker Image

Important: The RSA Helper image is hosted on GitHub Container Registry and requires authentication. See the GitHub Container Registry Authentication section above.

Option A: Pull from GitHub Container Registry (RECOMMENDED - requires authentication):

```
# First, authenticate (if not already done)
echo "YOUR_GITHUB_TOKEN" | docker login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin

# Pull the image using docker-compose
docker-compose pull

# Or pull directly
docker pull ghcr.io/ggiewon/rsahelper-webapi:latest
```

Option B: Load from .tar file (offline method):

```
# If you received the image as a tar file
docker load -i rsahelper-webapi-develop.tar

# Verify image loaded
docker images | grep rsahelper
```

Step 8: Start the Application

```
docker-compose up -d
```

Parameters explanation:

- `up` - starts services defined in docker-compose.yml
- `-d` - runs in detached mode (background)

Step 9: Verify Application is Running

```
# Check container status
docker-compose ps

# Expected output:
# NAME              IMAGE                        STATUS          PORTS
# rsahelper-api     rsahelper-webapi:latest     Up X seconds    0.0.0.0:5000->8080/tcp
```

Step 10: Test the Application

```
# Health check
curl http://localhost:5000/health

# Application version
curl http://localhost:5000/api/Version

# NIP list
curl http://localhost:5000/api/Authorization
```

In web browser, navigate to:

- Swagger UI: http://localhost:5000/swagger
- Health Check: http://localhost:5000/health

Linux Podman

Step 1: Install Podman

```
# Ubuntu 20.10 and newer
sudo apt-get install -y podman

# For older Ubuntu versions, add repository first:
sudo sh -c "echo 'deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu
wget -nv https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/xUbuntu_$(lsb_release -
sudo apt-get update
sudo apt-get install -y podman

# Install podman-compose
```

```
sudo apt-get install -y python3-pip
pip3 install podman-compose
```

Step 2: Verify Installation

```
podman --version
podman-compose --version
```

Step 3: Create Working Directory

```
mkdir -p ~/RSAHelper
cd ~/RSAHelper
```

Step 4: Create podman-compose.yml File

```
nano podman-compose.yml
```

Paste the following content:

```
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    volumes:
      - ./Data:/app/Data:ro
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s

networks:
  rsahelper-network:
    driver: bridge
```

Step 5: Create Data Directory

```
mkdir -p Data
cp ~/Downloads/allowed-nips.json Data/
chmod 644 Data/allowed-nips.json
```

Step 6: Load Image

Important: The RSA Helper image is hosted on GitHub Container Registry and requires authentication. See the GitHub Container Registry Authentication section.

Option A: Pull from GitHub Container Registry (RECOMMENDED - requires authentication):

```
# Authenticate
echo "YOUR_GITHUB_TOKEN" | podman login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin

# Pull the image using podman-compose
podman-compose pull

# Or pull directly
podman pull ghcr.io/ggiewon/rsahelper-webapi:latest
```

Option B: Load from .tar file (offline method):

```
podman load -i rsahelper-webapi-develop.tar
podman images | grep rsahelper
```

Step 7: Start the Application

```
podman-compose up -d
```

Step 8: Verify and Test

```
# Check status
podman-compose ps

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

# Test endpoints
curl http://localhost:5000/health
curl http://localhost:5000/api/Version
```

Step 9: Configure Systemd Service (Optional - Auto-start on Boot)

```
# Generate systemd service file
cd ~/RSAHelper
podman generate systemd --name rsahelper-api --files --new

# Move to systemd directory
mkdir -p ~/.config/systemd/user/
mv container-rsahelper-api.service ~/.config/systemd/user/

# Enable and start service
systemctl --user enable container-rsahelper-api.service
systemctl --user start container-rsahelper-api.service

# Enable lingering (allows services to run without login)
loginctl enable-linger $USER
```

Windows Installation

Windows Docker

> ⚠️ **IMPORTANT LICENSING NOTICE**
>
> **Docker Desktop is FREE for:**
>
> - Personal use
> - Small businesses (fewer than 250 employees AND less than $10 million in annual revenue)
> - Education and non-commercial open source projects
>
> **Docker Desktop requires a PAID SUBSCRIPTION for:**
>
> - Large businesses (250+ employees OR $10 million+ annual revenue)
> - Commercial use in medium/large organizations
>
> **Pricing:** Starting at $5/user/month (Professional) or $7/user/month (Team)
>
> For detailed licensing information, visit: https://www.docker.com/pricing/
>
> **Alternative:** If your organization requires a free solution, consider using **Podman Desktop** (see Windows Podman section below) which is completely free and open source.

Step 1: Install Docker Desktop

1. Download Docker Desktop from: https://www.docker.com/products/docker-desktop
2. Run the installer: `Docker Desktop Installer.exe`
3. Follow the installation wizard
4. Restart your computer when prompted
5. Launch Docker Desktop from Start menu
6. Wait for Docker to start (the whale icon in system tray will stop animating)

Step 2: Verify Installation

Open PowerShell and run:

```
docker --version
docker-compose --version
```

Expected output:

```
Docker version 24.x.x, build xxxxxxx
Docker Compose version v2.x.x
```

Step 3: Create Working Directory

```
# Create directory
New-Item -Path "C:\RSAHelper" -ItemType Directory
Set-Location -Path "C:\RSAHelper"
```

Step 4: Create docker-compose.yml File

```
# Create file using notepad
notepad docker-compose.yml
```

Paste the following content:

```
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    volumes:
      - ./Data:/app/Data:ro
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
    networks:
      - rsahelper-network

networks:
```

```
  rsahelper-network:
    driver: bridge
```

Save and close Notepad.

Step 5: Create Data Directory

```
# Create Data directory
New-Item -Path ".\Data" -ItemType Directory

# Copy configuration file
Copy-Item -Path "$env:USERPROFILE\Downloads\allowed-nips.json" -Destination ".\Data\"

# Verify structure
Get-ChildItem -Recurse
```

Expected structure:

```
C:\RSAHelper\
├── docker-compose.yml
└── Data\
    └── allowed-nips.json
```

Step 6: Load Docker Image

Important: The RSA Helper image is hosted on GitHub Container Registry and requires authentication. See the GitHub Container Registry Authentication section.

Option A: Pull from GitHub Container Registry (RECOMMENDED - requires authentication):

```
# Authenticate
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_TOKEN | docker login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin
Remove-Item Env:\GITHUB_TOKEN

# Pull the image using docker-compose
docker-compose pull

# Or pull directly
docker pull ghcr.io/ggiewon/rsahelper-webapi:latest
```

Option B: Load from .tar file (offline method):

```
docker load -i rsahelper-webapi-develop.tar
docker images | Select-String rsahelper
```

Step 7: Start the Application

```
docker-compose up -d
```

Step 8: Verify Application is Running

```
# Check container status
docker-compose ps

# Check logs
docker-compose logs rsahelper-api
```

Step 9: Test the Application

```
# Health check
Invoke-WebRequest -Uri http://localhost:5000/health

# Application version
Invoke-WebRequest -Uri http://localhost:5000/api/Version

# Or use curl (if installed)
curl http://localhost:5000/health
```

In web browser:

- Swagger UI: http://localhost:5000/swagger
- Health Check: http://localhost:5000/health

Windows Podman

> ✅ **FREE AND OPEN SOURCE**
>
> Podman Desktop is completely **free** for all users and use cases:
>
> - No licensing restrictions
> - No commercial subscriptions required
> - Open source (Apache 2.0 license)
> - Suitable for personal, educational, and commercial use
>
> **Perfect alternative to Docker Desktop for organizations requiring free solutions.**

Step 1: Install Podman Desktop

1. Download Podman Desktop from: https://podman-desktop.io/downloads
2. Run the installer: `podman-desktop-x.x.x-setup.exe`

3. Follow the installation wizard
4. Launch Podman Desktop from Start menu
5. Complete the initial setup wizard
    - Initialize Podman Machine
    - Set resource limits (CPU, Memory)

Step 2: Install Podman CLI

Podman Desktop includes the CLI, but verify it's in your PATH:

```
# Check Podman version
podman --version

# If command not found, add to PATH:
# Typically located at: C:\Program Files\RedHat\Podman\
$env:Path += ";C:\Program Files\RedHat\Podman\"
```

Step 3: Start Podman Machine

```
# Initialize Podman machine (first time only)
podman machine init

# Start Podman machine
podman machine start

# Verify it's running
podman machine list
```

Expected output:

```
NAME                     VM TYPE     CREATED      LAST UP            CPUS        MEMORY      DISK SIZE
podman-machine-default*  wsl         2 hours ago  Currently running  2           2GiB        100GiB
```

Step 4: Create Working Directory

```
New-Item -Path "C:\RSAHelper" -ItemType Directory
Set-Location -Path "C:\RSAHelper"
```

Step 5: Create Compose File

```
notepad docker-compose.yml
```

Note: Podman uses the same docker-compose.yml format. Paste:

```
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    volumes:
      - ./Data:/app/Data:ro
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
```

Step 6: Create Data Directory

```
New-Item -Path ".\Data" -ItemType Directory
Copy-Item -Path "$env:USERPROFILE\Downloads\allowed-nips.json" -Destination ".\Data\"
```

Step 7: Load Image

Important: The RSA Helper image is hosted on GitHub Container Registry and requires authentication. See the GitHub Container Registry Authentication section.

Option A: Pull from GitHub Container Registry (RECOMMENDED - requires authentication):

```
# Authenticate
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_TOKEN | podman login ghcr.io -u YOUR_GITHUB_USERNAME --password-stdin
Remove-Item Env:\GITHUB_TOKEN

# Pull the image using podman-compose
podman-compose pull

# Or pull directly
podman pull ghcr.io/ggiewon/rsahelper-webapi:latest
```

Option B: Load from .tar file (offline method):

```
podman load -i rsahelper-webapi-develop.tar
podman images | Select-String rsahelper
```

Step 8: Start the Application

```
# Using podman-compose
podman-compose up -d

# Or using podman directly
podman play kube docker-compose.yml
```

Step 9: Verify and Test

```
# Check status
podman ps

# View logs
podman logs rsahelper-api

# Follow logs
podman logs -f rsahelper-api

# Test endpoints
Invoke-WebRequest -Uri http://localhost:5000/health
```

Common Operations

Viewing Logs

Docker:

```
# Linux
docker-compose logs -f rsahelper-api
docker-compose logs --tail=100 rsahelper-api
docker-compose logs --since 1h rsahelper-api

# Windows PowerShell
docker-compose logs -f rsahelper-api
docker-compose logs --tail 100 rsahelper-api
```

Podman:

```
# Linux
podman-compose logs -f rsahelper-api
podman logs -f rsahelper-api

# Windows PowerShell
podman logs -f rsahelper-api
podman logs --tail 100 rsahelper-api
```

Stopping the Application

Docker:

```
# Stop without removing container
docker-compose stop

# Stop and remove container (data in Data/ remains)
docker-compose down

# Stop and remove everything including volumes
docker-compose down -v
```

Podman:

```
# Stop without removing
podman-compose stop

# Stop and remove
podman-compose down

# Using podman directly
podman stop rsahelper-api
podman rm rsahelper-api
```

Restarting the Application

Docker:

```
# Restart running container
docker-compose restart

# Start after stop
docker-compose start

# Recreate container
docker-compose up -d --force-recreate
```

Podman:

```
podman-compose restart
podman-compose start
podman restart rsahelper-api
```

Updating the Application

Docker:

```
# 1. Stop and remove old container
docker-compose down

# 2. Pull new image
docker-compose pull

# 3. Start new version
docker-compose up -d

# 4. Check logs
docker-compose logs -f rsahelper-api
```

Podman:

```
# 1. Stop container
podman-compose down

# 2. Pull new image
podman-compose pull

# 3. Start new version
podman-compose up -d

# 4. Verify
podman-compose ps
```

Checking Resource Usage

Docker:

```
# Real-time stats
docker stats rsahelper-api

# Container details
docker inspect rsahelper-api

# Processes inside container
docker top rsahelper-api
```

Podman:

```
# Real-time stats
podman stats rsahelper-api

# Container details
podman inspect rsahelper-api

# Processes
podman top rsahelper-api
```

Configuration Management

Update via API

```
# Linux
curl -X POST http://localhost:5000/api/Authorization/upload \
  -H "Content-Type: application/json" \
  -d '{
    "nips": ["5732296089", "9721121810"],
    "version": "1.0",
    "createdAt": "2024-12-29T12:00:00Z",
    "description": "Production NIP list",
    "hash": "your-hmac-sha256-hash-in-base64"
  }'
```

Windows PowerShell:

```
$body = @{
    nips = @("5732296089", "9721121810")
    version = "1.0"
    createdAt = "2024-12-29T12:00:00Z"
    description = "Production NIP list"
    hash = "your-hmac-sha256-hash-in-base64"
} | ConvertTo-Json

Invoke-RestMethod -Uri "http://localhost:5000/api/Authorization/upload" `
    -Method Post `
    -Body $body `
    -ContentType "application/json"
```

Retrieve Current NIP List

```
# JSON format
curl http://localhost:5000/api/Authorization

# Text format (one NIP per line)
curl http://localhost:5000/api/Authorization/NipListFile

# Save to file
curl http://localhost:5000/api/Authorization/NipListFile > nip-list.txt
```

Windows PowerShell:

```
# JSON format
Invoke-RestMethod -Uri "http://localhost:5000/api/Authorization"

# Text format
```

```
Invoke-WebRequest -Uri "http://localhost:5000/api/Authorization/NipListFile" `
    -OutFile "nip-list.txt"
```

Offline Mode Configuration

RSA Helper supports offline mode, allowing the application to work without internet access. In offline mode, KSeF certificates and XML schemas are loaded from local files instead of being downloaded from external services.

Overview

Mode	Description
Online (default)	Certificates and schemas downloaded from internet
Hybrid	Local files first, network fallback if missing
Strict Offline	Local files only, no network access

Configuration

Add or modify the `OfflineMode` section in `appsettings.additional.json`:

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

Parameters:

- `Enabled` - Enable/disable offline mode (default: `false`)
- `CertificatesPath` - Path to certificates folder (default: `Data/Certificates`)
- `FallbackToOnline` - Try network if local files missing (default: `true`)

Docker Configuration for Offline Mode

Update your `docker-compose.yml` to mount the certificates directory:

```
services:
  rsahelper-api:
    image: ghcr.io/ggiewon/rsahelper-webapi:latest
    container_name: rsahelper-api
    ports:
      - "5000:8080"
    volumes:
```

```
      - ./Data:/app/Data:ro
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
```

Certificate Directory Structure

```
Data/
├── allowed-nips.json
├── appsettings.additional.json (optional - for offline config)
└── Certificates/
    ├── DEMO/
    │   └── certificates.json
    ├── TEST/
    │   └── certificates.json
    └── PROD/
        └── certificates.json
```

Certificate File Format

Each `certificates.json` file should contain:

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

Certificate Download Tool

Use the **RSAHelper.CertificateDownloader** tool to download certificates from KSeF API and save them in the required format.

Download: available on the files page (Tools section, no sign-in required).

Usage:

```
# Download certificates for all environments
RSAHelper.CertificateDownloader --all -o ./Data/Certificates

# Download certificates for specific environment
RSAHelper.CertificateDownloader -e TEST -o ./Data/Certificates

# Download certificates for PROD environment
RSAHelper.CertificateDownloader -e PROD -o ./Data/Certificates
```

Parameters:

Parameter	Short	Description
`--environment`	`-e`	KSeF environment: `DEMO`, `TEST` or `PROD`
`--output`	`-o`	Output path (default: `Data/Certificates`)
`--all`	`-a`	Download for all environments
`--help`	`-h`	Display help

Setting Up Offline Mode

Step 1: Download Certificates (requires internet)

```
# On a machine with internet access
RSAHelper.CertificateDownloader --all -o ./Certificates

# This creates:
# ./Certificates/DEMO/certificates.json
# ./Certificates/TEST/certificates.json
# ./Certificates/PROD/certificates.json
```

Step 2: Copy Certificates to Server

```
# Linux
scp -r ./Certificates user@server:~/RSAHelper/Data/

# Windows
Copy-Item -Path ".\Certificates" -Destination "\\server\RSAHelper\Data\" -Recurse
```

Step 3: Create Offline Configuration

Create `Data/appsettings.additional.json`:

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

Step 4: Restart Container

```
docker-compose down
docker-compose up -d
```

Verify Offline Mode

Check application logs:

```
docker-compose logs rsahelper-api | grep -i "offline\|certificate"
```

Test encryption endpoint:

```
curl "http://localhost:5000/api/Calculate/test/testToken?nip=5732296089"
```

Hybrid Mode (Recommended for Production)

Hybrid mode uses local certificates when available, with network fallback:

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

Benefits:

- Faster response times (local certificates)
- Automatic fallback if local certificates expire
- Resilience during network outages

Certificate Maintenance

Certificates have expiration dates. Monitor and update them regularly:

```
# Check certificate validity
cat Data/Certificates/PROD/certificates.json | jq '.[].validTo'

# Re-download certificates before expiration
RSAHelper.CertificateDownloader --all -o ./Data/Certificates
docker-compose restart
```

Troubleshooting Offline Mode

Certificates Not Loading

```
# Check if certificates exist
ls -la Data/Certificates/*/certificates.json

# Verify JSON format
cat Data/Certificates/TEST/certificates.json | jq .

# Check container can access files
docker exec rsahelper-api ls -la /app/Data/Certificates/
```

Fallback to Network Not Working

Ensure `FallbackToOnline` is set to `true` and the container has network access:

```
# Test network from container
docker exec rsahelper-api curl -I https://api-test.ksef.mf.gov.pl
```

Certificate Validation Errors

Check certificate dates and usage types:

```
# View certificate details
cat Data/Certificates/TEST/certificates.json | jq '.[] | {validFrom, validTo, usage}'
```

Document Conversion Configuration

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

Configuration

Add or modify the `PDFSettings` section in `appsettings.json` or `appsettings.additional.json`:

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

Parameters:

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

Usage

When making API requests to conversion endpoints:

- If the `lang` query parameter is provided, 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 "..."

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

Docker Configuration

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

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

Then restart the container:

```
docker-compose restart
```

PDF Print Configuration

RSA Helper allows configuring PDF document generation parameters: conversion engine selection, page margins and page orientation. All settings are located in the `PDFSettings` section.

Configuration

Add or modify the `PDFSettings` section in `appsettings.json` or `appsettings.additional.json`:

```
{
  "PDFSettings": {
    "PdfEngine": "Syncfusion",
    "PdfOrientation": "Portrait",
    "PdfMargins": {
      "Top": 20,
      "Bottom": 20,
      "Left": 20,
      "Right": 20
    }
  }
}
```

Conversion Engine (`PdfEngine`)

The application supports three HTML to PDF conversion engines:

Value	Description	Requirements
`Syncfusion`	Engine based on Chromium browser (Blink). Best rendering quality	Requires Chromium installed
`WkHtmlToPdf`	Engine based on WebKit	No additional dependencies
`PdfSharp`	Lightweight rendering engine	No additional dependencies, limited CSS support

Default value: `Syncfusion`

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`

Docker Configuration

To change PDF settings in Docker, create or update `Data/appsettings.additional.json`:

```
{
  "PDFSettings": {
    "PdfEngine": "Syncfusion",
    "PdfOrientation": "Portrait",
    "PdfMargins": {
      "Top": 10,
      "Bottom": 10,
      "Left": 10,
      "Right": 10
    }
  }
}
```

Then restart the container:

```
docker-compose restart
```

QR Code Label Font Configuration

RSA Helper allows configuring the font used to render labels displayed below QR codes (e.g., KSeF number).

By default, the application uses a built-in DejaVu Sans font (`<embedded>`), which ensures consistent rendering across all environments without requiring any system fonts to be installed.

Configuration

Add or modify the `QrCode` section in `appsettings.json` or `appsettings.additional.json`:

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

Parameters:

- `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`

Docker Configuration

To change QR code font settings in Docker, create or update `Data/appsettings.additional.json`:

```
{
  "QrCode": {
    "FontFamily": "<embedded>",
    "DefaultFontSize": 12
  }
}
```

> **Note:** When using `<embedded>`, no additional font installation in the Docker container is required. To use a system font instead, specify its name (e.g., `"FontFamily": "DejaVu Sans"`) and ensure it is installed in the container.

Then restart the container:

```
docker-compose restart
```

Troubleshooting

Problem: Port 5000 Already in Use

Linux - Check what's using the port:

```
sudo netstat -tlnp | grep 5000
# or
sudo lsof -i :5000
```

Solution 1: Stop the application using port 5000

Solution 2: Change port in docker-compose.yml:

```
ports:
  - "5001:8080"  # Changed from 5000 to 5001
```

Then restart:

```
docker-compose down
docker-compose up -d
```

Windows - Check port usage:

```
Get-NetTCPConnection -LocalPort 5000
# or
netstat -ano | findstr :5000
```

Problem: Container Won't Start

Check logs:

```
# Docker
docker-compose logs rsahelper-api

# Podman
podman-compose logs rsahelper-api
```

Check container details:

```
# Docker
docker inspect rsahelper-api

# Podman
podman inspect rsahelper-api
```

Verify Docker/Podman is running:

```
# Linux Docker
sudo systemctl status docker

# Restart if needed
sudo systemctl restart docker

# Windows - restart Docker Desktop or Podman Desktop from system tray
```

Problem: Missing allowed-nips.json File

Check if file exists:

```
# Linux
ls -la Data/

# Windows
Get-ChildItem .\Data\
```

Check permissions (Linux):

```
ls -l Data/allowed-nips.json
chmod 644 Data/allowed-nips.json
```

Verify you're in the correct directory:

```
# Linux
pwd
# Should show: /home/username/RSAHelper

# Windows
Get-Location
# Should show: C:\RSAHelper
```

Problem: Permission Denied (Linux)

For Docker:

```
# Check if user is in docker group
groups

# If "docker" is not in the list:
sudo usermod -aG docker $USER

# Log out and log back in, or use:
newgrp docker

# Temporarily use sudo (not recommended for regular use)
sudo docker-compose up -d
```

For Podman:

```
# Podman runs rootless by default, no special permissions needed
# If issues persist, try:
```

```
podman system migrate
```

Problem: Application Not Responding

Check if container is running:

```
# Docker
docker-compose ps
docker ps -a | grep rsahelper

# Podman
podman-compose ps
podman ps -a | grep rsahelper
```

Check if port is open:

```
# Linux
netstat -tlnp | grep 5000

# Windows
Test-NetConnection localhost -Port 5000
```

Test from inside container:

```
# Docker
docker exec rsahelper-api curl http://localhost:8080/health

# Podman
podman exec rsahelper-api curl http://localhost:8080/health
```

Check health status:

```
# Docker
docker inspect --format='{{.State.Health.Status}}' rsahelper-api

# Podman
podman inspect --format='{{.State.Health.Status}}' rsahelper-api
```

Problem: Container Keeps Restarting

View recent logs:

```
docker-compose logs --tail=100 rsahelper-api
```

Disable auto-restart temporarily and run interactively:

```
docker-compose down
docker-compose up
# (without -d flag to see logs in real-time)
```

Common causes:

- Missing or invalid configuration file
- Port conflicts
- Insufficient resources (CPU/RAM)
- Corrupted image

Problem: Cannot Access Swagger UI

Verify application is running:

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

Check firewall (Linux):

```
sudo ufw status
sudo ufw allow 5000/tcp
```

Check Windows Defender Firewall:

```
# Create firewall rule
New-NetFirewallRule -DisplayName "RSA Helper" -Direction Inbound -LocalPort 5000 -Protocol TCP -Action Allow
```

Try accessing from different browser or clear cache

Problem: Podman Machine Not Starting (Windows)

```
# Check machine status
podman machine list

# Stop and remove existing machine
podman machine stop
```

```
podman machine rm podman-machine-default

# Reinitialize
podman machine init
podman machine start

# Check logs
podman machine inspect podman-machine-default
```

Production Considerations

Security Best Practices

Linux

```
# 1. Configure firewall
sudo ufw allow from 192.168.1.0/24 to any port 5000
sudo ufw enable

# 2. Set proper file permissions
chmod 644 Data/allowed-nips.json
chmod 755 Data/

# 3. Regular updates
sudo apt-get update
sudo apt-get upgrade docker.io
```

Windows

```
# 1. Configure Windows Firewall
New-NetFirewallRule -DisplayName "RSA Helper - Allow Local Network" `
    -Direction Inbound `
    -LocalPort 5000 `
    -Protocol TCP `
    -Action Allow `
    -RemoteAddress 192.168.1.0/24

# 2. Use Docker Desktop's security scanning
docker scan rsahelper-webapi:latest
```

Using Reverse Proxy (Linux - Nginx)

```
# Install Nginx
sudo apt-get install nginx
```

```
# Create configuration
sudo nano /etc/nginx/sites-available/rsahelper
```

Add the following configuration:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Enable and restart:

```
sudo ln -s /etc/nginx/sites-available/rsahelper /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

SSL/TLS Configuration (Optional)

```
# Install Certbot
sudo apt-get install certbot python3-certbot-nginx

# Obtain certificate
sudo certbot --nginx -d your-domain.com

# Auto-renewal is configured automatically
```

Backup and Restore

Backup

Linux:

```
# Backup Data directory
cd ~/RSAHelper
tar -czf rsahelper-backup-$(date +%Y%m%d).tar.gz Data/

# Full backup
```

```
cd ~
tar -czf rsahelper-full-$(date +%Y%m%d).tar.gz RSAHelper/
```

Windows:

```
# Backup Data directory
Compress-Archive -Path "C:\RSAHelper\Data" `
    -DestinationPath "C:\Backups\rsahelper-data-$(Get-Date -Format 'yyyyMMdd').zip"

# Full backup
Compress-Archive -Path "C:\RSAHelper" `
    -DestinationPath "C:\Backups\rsahelper-full-$(Get-Date -Format 'yyyyMMdd').zip"
```

Restore

Linux:

```
# Stop container
cd ~/RSAHelper
docker-compose down

# Restore data
tar -xzf rsahelper-backup-20241229.tar.gz

# Start container
docker-compose up -d
```

Windows:

```
# Stop container
Set-Location C:\RSAHelper
docker-compose down

# Restore data
Expand-Archive -Path "C:\Backups\rsahelper-data-20241229.zip" `
    -DestinationPath "C:\RSAHelper" -Force

# Start container
docker-compose up -d
```

Monitoring

Set up Log Rotation (Linux - Docker)

```
# Create Docker daemon configuration
sudo nano /etc/docker/daemon.json
```

Add:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  }
}
```

Restart Docker:

```
sudo systemctl restart docker
```

Continuous Health Monitoring

Linux:

```
# Create monitoring script
nano ~/monitor-rsahelper.sh
```

Add:

```
#!/bin/bash
while true; do
    if ! curl -f -s http://localhost:5000/health > /dev/null; then
        echo "$(date): RSA Helper is down, restarting..."
        cd ~/RSAHelper
        docker-compose restart
    fi
    sleep 60
done
```

Make executable and run:

```
chmod +x ~/monitor-rsahelper.sh
nohup ~/monitor-rsahelper.sh &
```

Windows (PowerShell script):

```
# Create monitoring script
while ($true) {
    try {
```

```
        $response = Invoke-WebRequest -Uri "http://localhost:5000/health" -TimeoutSec 5
        if ($response.StatusCode -ne 200) {
            Write-Host "$(Get-Date): RSA Helper unhealthy, restarting..."
            Set-Location C:\RSAHelper
            docker-compose restart
        }
    } catch {
        Write-Host "$(Get-Date): RSA Helper is down, restarting..."
        Set-Location C:\RSAHelper
        docker-compose restart
    }
    Start-Sleep -Seconds 60
}
```

Auto-start on System Boot

Linux - Docker

Docker is already set to auto-start. The container will automatically restart due to `restart: unless-stopped` policy.

Verify:

```
sudo systemctl status docker
docker inspect rsahelper-api | grep -A 5 RestartPolicy
```

Linux - Podman with Systemd

```
# Generate systemd service
cd ~/RSAHelper
podman generate systemd --name rsahelper-api --files --new

# Install service
mkdir -p ~/.config/systemd/user/
mv container-rsahelper-api.service ~/.config/systemd/user/

# Enable and start
systemctl --user enable container-rsahelper-api.service
systemctl --user start container-rsahelper-api.service

# Enable lingering (allows service to run without login)
loginctl enable-linger $USER
```

Windows - Docker Desktop

Docker Desktop starts automatically with Windows. Containers with `restart: unless-stopped` will auto-start.

Verify in Docker Desktop:

1. Open Docker Desktop
2. Go to Settings → General
3. Ensure “Start Docker Desktop when you log in” is checked

Windows - Podman

Podman Machine can be configured to auto-start:

```
# Configure Podman machine to start on boot
podman machine set --rootful=false --now podman-machine-default
```

Resource Limits

Add resource limits to docker-compose.yml:

```
services:
  rsahelper-api:
    # ... existing configuration ...
    deploy:
      resources:
        limits:
          cpus: '2'
          memory: 2G
        reservations:
          cpus: '1'
          memory: 1G
```

Cleanup and Maintenance

Docker:

```
# Remove unused images
docker image prune -a

# Remove unused volumes
docker volume prune

# Remove all unused resources
docker system prune -a

# Check disk usage
docker system df
```

Podman:

```
# Remove unused images
podman image prune -a

# Remove unused volumes
podman volume prune

# System cleanup
podman system prune -a
```

```
# Check disk usage
podman system df
```

Quick Reference

Docker Commands Cheatsheet

```
# Start
docker-compose up -d

# Stop
docker-compose stop

# Remove
docker-compose down

# Restart
docker-compose restart

# Logs
docker-compose logs -f

# Status
docker-compose ps

# Execute command in container
docker exec -it rsahelper-api bash

# View resource usage
docker stats rsahelper-api
```

Podman Commands Cheatsheet

```
# Start
podman-compose up -d

# Stop
podman-compose stop

# Remove
podman-compose down

# Restart
podman-compose restart

# Logs
podman logs -f rsahelper-api

# Status
podman ps

# Execute command in container
```

```
podman exec -it rsahelper-api bash

# View resource usage
podman stats rsahelper-api
```

Testing Endpoints

```
# Health check
curl http://localhost:5000/health

# Version
curl http://localhost:5000/api/Version

# NIP list (JSON)
curl http://localhost:5000/api/Authorization

# NIP list (text)
curl http://localhost:5000/api/Authorization/NipListFile

# Check specific NIP
curl http://localhost:5000/api/Authorization/5732296089

# Swagger UI (browser)
http://localhost:5000/swagger
```

Support and Documentation

- **Application Manual:** RSAHelper_Manual.pdf
- **Swagger Documentation:** http://localhost:5000/swagger
- **Health Check:** http://localhost:5000/health
- **Docker Documentation:** https://docs.docker.com
- **Podman Documentation:** https://docs.podman.io

Appendix: Example allowed-nips.json Structure

```
{
  "nips": [
    "5732296089",
    "9721121810",
    "1234567890"
  ],
  "version": "1.0",
  "createdAt": "2024-12-29T12:00:00Z",
  "description": "Production NIP list",
  "hash": "base64-encoded-hmac-sha256-hash"
}
```

Note: The `hash` field must be a valid HMAC-SHA256 hash computed using the key configured in the application.

Document Version: 1.1 **Last Updated:** February 11, 2026 **Compatible with RSA Helper:** 1.2.5+