WhiteListChecker version 2.0.0

VAT Taxpayer Whitelist Verification Application

Table of Contents

1. General Information

WhiteListChecker is a web application (WebAPI) for verifying entities against the Polish Ministry of Finance VAT taxpayer whitelist (Wykaz podatników VAT). The application provides the following functionalities:

- Automatic daily download of the whitelist data file from the Ministry of Finance server (plikplaski.mf.gov.pl)
- Verification of NIP (tax identification number) and bank account number against the VAT taxpayer whitelist
- Verification using SHA-512 algorithm (according to Ministry of Finance specification)
- Support for bank account masks (virtual accounts)
- Manual data file download for a specified date
- Automatic cleanup of old data files (configurable retention period)
- Download status notifications (Email, Microsoft Teams)
- Swagger interface for API testing and documentation

Key Information:

- Application listens on internal port **8080**
- Default external port mapping: **5000**
- Swagger UI: `/swagger`
- Version / health check endpoint: `/api/version`
- Data source: `https://plikplaski.mf.gov.pl/pliki/{yyyyMMdd}.7z`

2. Requirements

2.1. Hardware Requirements

For proper application operation, the following hardware is required:

- 2-core processor
- 4 GB RAM (the whitelist data file is large - extracted size exceeds 1GB)
- 2 GB free disk space (for data files and archives for several days)

2.2. Software Requirements

The application requires:

- Docker Engine 20.10+ or Podman 4.0+ (recommended)
- Docker Compose / Podman Compose (optional, recommended)

Alternatively, for running without containerization:

- ASP.NET Core Runtime version 10.0

2.3. Additional Requirements

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

- **https://plikplaski.mf.gov.pl** - source of the daily VAT taxpayer whitelist data file (flat file in 7z format)

Checking Address Availability

To ensure that the Ministry of Finance server is accessible from the machine where the application is installed, you can perform the following tests:

Windows (PowerShell):

```
Test-NetConnection plikplaski.mf.gov.pl -Port 443
```

Windows (CMD):

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

Linux:

```
curl -f -s -o /dev/null -w "%{http_code}\n" https://plikplaski.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. Linux Installation

Docker Installation (recommended)

Requirements:

- Docker Engine 20.10+
- Docker Compose (optional)

Installation steps:

1. Install Docker:

```
sudo apt-get install -y docker.io docker-compose
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker $USER
# IMPORTANT: Log out and log back in for group changes to take effect
```

2. Create application directory:

```
mkdir -p ~/WhiteListChecker
cd ~/WhiteListChecker
```

3. Create `docker-compose.yml` file:

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

networks:
  whitelistchecker-network:
    driver: bridge
```

4. Create data directories:

```bash
mkdir -p DataFiles Downloads logs
chmod 755 DataFiles Downloads logs
```

5. Verify directory structure:

```bash
ls -la
```

Expected structure:

```
WhiteListChecker/
├── docker-compose.yml
├── DataFiles/
├── Downloads/
└── logs/
```

6. Log in to GitHub Container Registry and start:

```
# Authentication (required to pull the image)
echo "YOUR_GITHUB_TOKEN" | docker login ghcr.io -u YOUR_GITHUB_USERNAME --p

# Pull image
docker-compose pull

# Start
docker-compose up -d
```

7. Verify operation:

```
# Container status
docker-compose ps

# Application logs
docker-compose logs -f whitelistchecker-api

# Test version endpoint
curl http://localhost:5000/api/version
```

In web browser:

- Swagger UI: http://localhost:5000/swagger

Detailed installation instructions using containers (Docker and Podman, Linux and Windows) are available in the WhiteListChecker_Installation_Guide.pdf file.

Podman Installation

```
# Install Podman
sudo apt-get install -y podman
pip3 install podman-compose

# Start (same docker-compose.yml configuration)
echo "YOUR_GITHUB_TOKEN" | podman login ghcr.io -u YOUR_GITHUB_USERNAME --p
podman-compose pull
podman-compose up -d
```

3.2. Windows Installation

Docker Desktop Installation

1. Install Docker Desktop from: https://www.docker.com/products/docker-desktop
2. Create directory `C:\WhiteListChecker`
3. Create `docker-compose.yml` file (same content as in section 3.1)
4. Create directories: `DataFiles`, `Downloads`, `logs`:

```
New-Item -Path "C:\WhiteListChecker" -ItemType Directory
Set-Location -Path "C:\WhiteListChecker"
New-Item -Path ".\DataFiles" -ItemType Directory
New-Item -Path ".\Downloads" -ItemType Directory
New-Item -Path ".\logs" -ItemType Directory
```

5. Log in to GitHub Container Registry:

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

6. Pull and start:

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

7. Verify operation:

```
# Container status
docker-compose ps

# Test version endpoint
Invoke-WebRequest -Uri http://localhost:5000/api/version
```

In web browser:

- Swagger UI: http://localhost:5000/swagger

To test whether the application was installed correctly, you can call the **Version** endpoint (see section 5.4).

> **NOTE:** Alternatively, you can use **Podman Desktop** (free, open source) instead of Docker Desktop. Details in WhiteListChecker_Installation_Guide.pdf.

---

4. Configuration

4.1. Configuration Files

The application uses two configuration files:

1. `appsettings.json` - Main configuration file (built into the Docker image)
2. `appsettings.additional.json` - Optional override file (can be mounted as a Docker volume)

Configuration loading order (from lowest to highest priority):

1. `appsettings.json`
2. `appsettings.additional.json` (optional)
3. Environment variables

Configuration can also be overridden using **environment variables**.

4.2. Configuration Parameters

Below is the full structure of the `appsettings.json` file:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Trace",
      "Microsoft": "Information"
    }
  },
  "AllowedHosts": "*",
  "HttpClient": {
    "TimeoutMinutes": 30
  },
  "Download": {
    "AlertAfterHour": 10
  },
  "Maintenance": {
    "RetentionDays": 4
  },
  "Notifications": {
    "Email": {
      "Enabled": false,
      "SmtpHost": "",
      "SmtpPort": 587,
      "UseSsl": true,
      "Username": "",
      "Password": "",
      "FromAddress": "",
      "FromName": "WhiteListChecker",
      "Recipients": []
    },
    "Teams": {
      "Enabled": false,
      "WebhookUrl": ""
    }
  }
}
```

Section descriptions:

`HttpClient` Section

HTTP client configuration used for downloading data files from the Ministry of Finance server.

Parameter	Description	Default
`TimeoutMinutes`	HTTP request timeout in minutes. Data files are large (>100MB), so the default value is set to 30 minutes	`30`

> **NOTE:** For slow internet connections or large data files, you may need to increase this value.

`Download` Section

Configuration for automatic file download by `DownloadService`.

Parameter	Description	Default
`AlertAfterHour`	Hour (0-23) after which an error notification is sent if the data file for the current day has not yet been downloaded. The Ministry of Finance typically publishes files in the morning, so the default value is 10	`10`

`Maintenance` Section

Configuration for automatic cleanup of old data files by `MaintenanceService`.

Parameter	Description	Default
`RetentionDays`	Number of days to retain data files. Directories older than this value are automatically deleted. A value of 4 means data from the last 4 days is kept	`4`

> **NOTE:** Each extracted data file takes over 1GB of disk space. With the default setting of 4 days, a minimum of ~5GB of disk space is required.

`Logging` Section

Logging level configuration. The application uses **NLog** for logging. Log files are written to the `logs/` directory.

Parameter	Description	Default
`LogLevel:Default`	Default logging level	`Trace`
`LogLevel:Microsoft`	Logging level for Microsoft components	`Information`

Available logging levels (from least to most verbose): `None`, `Critical`, `Error`, `Warning`, `Information`, `Debug`, `Trace`.

4.3. Email Notification Configuration

The `Notifications.Email` section is responsible for configuring email notifications. Notifications are sent on successful file download and on download errors.

```
"Notifications": {
  "Email": {
    "Enabled": true,
    "SmtpHost": "smtp.example.com",
    "SmtpPort": 587,
    "UseSsl": true,
    "Username": "user@example.com",
    "Password": "password",
    "FromAddress": "whitelistchecker@example.com",
    "FromName": "WhiteListChecker",
    "Recipients": ["admin@example.com", "operator@example.com"]
  }
}
```

Parameter	Description	Type	Default
`Enabled`	Enable/disable email notifications	boolean	`false`
`SmtpHost`	SMTP server hostname	string	`""`
`SmtpPort`	SMTP server port	integer	`587`
`UseSsl`	Use SSL/TLS for SMTP connection	boolean	`true`
`Username`	SMTP authentication username	string	`""`

`Password`	SMTP authentication password	string	`""`
`FromAddress`	Sender email address	string	`""`
`FromName`	Sender display name in message header	string	`"WhiteListChecker"`
`Recipients`	List of recipient email addresses for notifications	string[]	`[]`

Common SMTP configurations:

Server	Host	Port	SSL
Microsoft 365	`smtp.office365.com`	`587`	`true`
Gmail	`smtp.gmail.com`	`587`	`true`
Exchange on-premise	`mail.company.com`	`25` or `587`	depends on configuration

> **NOTE:** For Gmail, an App Password is required instead of the standard account password.

4.4. Teams Notification Configuration

The `Notifications.Teams` section is responsible for configuring Microsoft Teams notifications via Incoming Webhook. Notifications are displayed as Adaptive Cards with information about the file name, date, and status.

```
"Notifications": {
  "Teams": {
    "Enabled": true,
    "WebhookUrl": "https://outlook.office.com/webhook/..."
  }
}
```

Parameter	Description	Type	Default
`Enabled`	Enable/disable Teams notifications	boolean	`false`

`WebhookUrl`	Microsoft Teams Incoming Webhook URL	string	`""`

Creating a Webhook in Microsoft Teams

1. Open a channel in Microsoft Teams
2. Click the "..." icon → "**Connectors**"
3. Find "**Incoming Webhook**" and click "**Configure**"
4. Give it a name (e.g., `WhiteListChecker`) and optionally upload an icon
5. Click "**Create**"
6. Copy the generated webhook URL
7. Paste the URL into the `WebhookUrl` configuration

4.5. Overriding Configuration

There are three ways to override the default configuration:

Method 1: `appsettings.additional.json` file

Create an `appsettings.additional.json` file and mount it as a Docker volume. The file overrides only the values defined in it - other settings remain at their defaults.

Example file:

```
{
  "HttpClient": {
    "TimeoutMinutes": 60
  },
  "Download": {
    "AlertAfterHour": 12
  },
  "Maintenance": {
    "RetentionDays": 7
  },
  "Notifications": {
    "Email": {
      "Enabled": true,
      "SmtpHost": "smtp.office365.com",
      "SmtpPort": 587,
      "UseSsl": true,
      "Username": "user@company.com",
      "Password": "password123",
```

```
      "FromAddress": "whitelistchecker@company.com",
      "FromName": "WhiteListChecker",
      "Recipients": ["admin@company.com"]
    },
    "Teams": {
      "Enabled": true,
      "WebhookUrl": "https://outlook.office.com/webhook/..."
    }
  }
}
```

Mounting as Docker volume:

```
volumes:
  - ./appsettings.additional.json:/app/appsettings.additional.json:ro
  - ./DataFiles:/app/DataFiles
  - ./Downloads:/app/Downloads
  - ./logs:/app/logs
```

After changing the file, restart the container:

```
docker-compose restart
```

Method 2: Environment variables

Use double underscore (`__`) as separator for nested keys in environment variables:

```
environment:
  - ASPNETCORE_ENVIRONMENT=Production
  - ASPNETCORE_URLS=http://+:8080
  - HttpClient__TimeoutMinutes=60
  - Download__AlertAfterHour=12
  - Maintenance__RetentionDays=7
  - Notifications__Email__Enabled=true
  - Notifications__Email__SmtpHost=smtp.office365.com
  - Notifications__Email__SmtpPort=587
  - Notifications__Email__Username=user@company.com
  - Notifications__Email__Password=password123
```

```
      - Notifications__Email__FromAddress=whitelistchecker@company.com
      - Notifications__Email__Recipients__0=admin@company.com
      - Notifications__Email__Recipients__1=operator@company.com
      - Notifications__Teams__Enabled=true
      - Notifications__Teams__WebhookUrl=https://outlook.office.com/webhook/...
```

Method 3: Combining both methods

Environment variables have the highest priority, followed by `appsettings.additional.json`, and then `appsettings.json`. Both methods can be combined.

---

5. 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

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).

5.1. Check

Base Path: `/api/check`

Controller responsible for verifying NIP number and bank account against the VAT taxpayer whitelist.

5.1.1. Verify NIP and Bank Account

Endpoint: `POST /api/check`

Description: Verifies one or more NIP+bank account pairs against the VAT taxpayer whitelist for a specified date. The method loads the data file for the given date, computes a SHA-512 hash for each NIP+account pair (according to the Ministry of Finance specification), and then compares it against the list of active and exempt taxpayer hashes.

Content-Type: `multipart/form-data`

Parameters (form data):

- `Date` (string, required) - Date for which the whitelist is checked, format: `yyyyMMdd` (e.g., `20260208`)
- `NipAccount` (array of string, required) - List of NIP+account pairs to verify. Each element has the format: `NIP|BankAccountNumber`

NipAccount Format:

Format of each array element: `NNNNNNNNNN|PPKKBBBBBBBBCCCCCCCCCCCC`

Where:

- `NNNNNNNNNN` - 10-digit NIP number
- `|` - separator
- `PP` - country code (e.g., `PL`)
- `KK` - IBAN check digits
- `BBBBBBBB` - bank routing number (8 digits)
- `CCCCCCCCCCCC` - client account number

Example: `1234567890|PL12345678901234567890123456`

Response:

- Status: `200 OK`

- Format: JSON


```
{
  "checks": [
    {
      "nipAccount": "1234567890|PL12345678901234567890123456",
      "existOnList": true,
      "fileCrc": "a1b2c3d4e5f6...",
      "date": "20260208",
      "sha512": "f7e6d5c4b3a2..."
    }
  ]
}
```


Response structure:

Field	Type	Description
`checks`	array	Array of verification results

`SingleCheckEntry` structure (element of `checks` array):

Field	Type	Description
`nipAccount`	string	Checked NIP+account pair (echo of input parameter)
`existOnList`	boolean	`true` - the NIP+account pair exists on the VAT taxpayer whitelist (active or exempt); `false` - the pair was not found on the list
`fileCrc`	string	SHA checksum of the data file used for verification. Used to confirm file integrity
`date`	string	Date of the data file used for verification (format `yyyyMMdd`)
`sha512`	string	Computed SHA-512 hash for the given NIP+account pair. The hash is computed according to the MF specification (multiple SHA-512 transformations)

Operation Mechanism:

1. The application loads the whitelist data file for the given date
2. For each NIP+account pair: a. Concatenates date and NIP+account pair into a single string: `{date}{nipAccount}` b. Computes multiple SHA-512 hash (number of transformations specified in the data file) c. Compares computed hash against the list of active taxpayer hashes (`skrotyPodatnikowCzynnych`) d. Compares computed hash against the list of exempt taxpayer hashes (`skrotyPodatnikowZwolnionych`) e. If not found - tries with bank account masks (virtual account support)
3. Returns verification result for each pair

Bank Account Mask Handling:

The Ministry of Finance provides a list of bank account masks in the data file. Masks are used to handle virtual accounts where part of the account number is replaced with `X` character. If the standard check does not produce a positive result, the application automatically applies appropriate masks and repeats the verification.

Error codes:

- `500 Internal Server Error` - Server error (e.g., missing data file for specified date, file read error)

Example call (curl):

```
curl -X POST http://localhost:5000/api/check \
  -F "Date=20260208" \
  -F "NipAccount=1234567890|PL12345678901234567890123456"
```

Example with multiple pairs:

```
curl -X POST http://localhost:5000/api/check \
  -F "Date=20260208" \
  -F "NipAccount=1234567890|PL12345678901234567890123456" \
  -F "NipAccount=9876543210|PL65432109876543210987654321"
```

Windows PowerShell:

```
$body = @{
    Date = "20260208"
    NipAccount = @(
        "1234567890|PL12345678901234567890123456",
        "9876543210|PL65432109876543210987654321"
    )
}
Invoke-RestMethod -Uri "http://localhost:5000/api/check" -Method Post -Form
```

Example response with multiple results:

```
{
  "checks": [
    {
      "nipAccount": "1234567890|PL12345678901234567890123456",
      "existOnList": true,
      "fileCrc": "a1b2c3d4e5f6789...",
      "date": "20260208",
      "sha512": "f7e6d5c4b3a29..."
    },
    {
      "nipAccount": "9876543210|PL65432109876543210987654321",
      "existOnList": false,
      "fileCrc": "a1b2c3d4e5f6789...",
      "date": "20260208",
      "sha512": "1a2b3c4d5e6f7..."
    }
  ]
}
```

> **NOTE:** For verification to succeed, the data file for the specified date must have been previously downloaded. Files are downloaded automatically by `DownloadService` (see section 6.1) or manually using the Download endpoint (see section 5.2).

5.2. Download

Base Path: `/api/download`

Controller responsible for manually downloading the whitelist data file.

5.2.1. Download Data File

Endpoint: `GET /api/download?date={yyyyMMdd}`

Description: Manually initiates downloading and unpacking the whitelist data file from the Ministry of Finance server (`https://plikplaski.mf.gov.pl/pliki/{date}.7z`). The file is downloaded, saved to the `Downloads` directory, unpacked to `DataFiles/{date}/` , and then the archive is deleted.

Parameters:

- `date` (query, optional) - Date of the file to download in `yyyyMMdd` format (e.g., `20260208`). If not provided, current date is used
- `force` (query, optional, boolean) - Force re-download even if the file for the given date already exists. Default: `false`

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "requestedDate": "20260208",
  "downloadStatus": "Ok",
  "errorMessage": null
}
```

Response structure:

Field	Type	Description
`requestedDate`	string	Date of the file that was downloaded (format `yyyyMMdd`)
`downloadStatus`	enum	Download operation status (see table below)
`errorMessage`	string/null	Error message (null if operation completed successfully)

`downloadStatus` values:

Value	Description
`Unknown`	Unknown status (initial state)
`Ok`	File downloaded and unpacked successfully
`Skipped`	File for the given date already exists on disk. Use the `force=true` parameter to force re-download
`Error`	An error occurred during download or unpacking. Details in `errorMessage` field

Example calls:

```
# Download file for current date
curl http://localhost:5000/api/download
```

```
# Download file for specific date
curl "http://localhost:5000/api/download?date=20260208"
```

```
# Force re-download (even if file already exists)
curl "http://localhost:5000/api/download?date=20260208&force=true"
```

Windows PowerShell:

```
# Download file for current date
Invoke-RestMethod -Uri "http://localhost:5000/api/download"

# Force re-download
Invoke-RestMethod -Uri "http://localhost:5000/api/download?date=20260208&fo
```

Example response - file already exists:

```
{
  "requestedDate": "20260208",
  "downloadStatus": "Skipped",
  "errorMessage": null
}
```

Example response - error:

```
{
  "requestedDate": "20260208",
  "downloadStatus": "Error",
  "errorMessage": "Response status code does not indicate success: 404 (Not
}
```

> **NOTE:** Downloading large files may take several minutes depending on internet connection speed. The HTTP request timeout is configurable via the `HttpClient:TimeoutMinutes` parameter (default 30 minutes).

5.3. Clear

Base Path: `/api/clear`

Controller responsible for manually cleaning up old data files.

5.3.1. Clear Old Data

Endpoint: `GET /api/clear?date={yyyyMMdd}`

Description: Deletes data directories older than the cutoff date calculated from the reference date and `Maintenance:RetentionDays` setting. The endpoint is used for manual cleanup of old data files. Automatic cleanup is performed by `MaintenanceService` (see section 6.2).

Parameters:

- `date` (query, optional) - Reference date in `yyyyMMdd` format (e.g., `20260208`). If not provided, current date is used

Operation Mechanism:

For the given date, the endpoint:

- calculates cutoff date: `date - RetentionDays`
- deletes all directories with a date older than the cutoff date
- e.g., for `date=20260208` and `RetentionDays=4`, cutoff date is `20260204`, so directories up to `20260203` are deleted

Response:

- Status: `200 OK`
- Format: JSON

```
{
  "startDate": "2026-02-08T00:00:00",
  "deletedDataFolders": 3,
  "errorMessage": null
}
```

Response structure:

Field	Type	Description
`startDate`	DateTime	Reference date (parsed from `date` parameter)
`deletedDataFolders`	integer	Number of actually deleted data directories
`errorMessage`	string/null	Error message (null if operation completed successfully)

Example calls:

```
# Clear old data relative to current date
curl http://localhost:5000/api/clear
```

```
# Clear old data relative to specific date
curl "http://localhost:5000/api/clear?date=20260208"
```

Windows PowerShell:

```
Invoke-RestMethod -Uri "http://localhost:5000/api/clear"
```

Example response:

```
{
  "startDate": "2026-02-08T00:00:00",
  "deletedDataFolders": 5,
  "errorMessage": null
}
```

5.4. Version

Base Path: `/api/version`

Controller responsible for application version information.

5.4.1. Get Application Version

Endpoint: `GET /api/version`

Description: Returns information about application version and build date. Can be used as **Health Check** to verify if the application is running and available. The endpoint is also used by the Docker HEALTHCHECK mechanism to monitor container status.

Parameters: None

Response:

- Status: `200 OK`

- Format: JSON

```
{
  "versionId": "2.0.0+<git-commit-sha>",
  "buildDate": "2026-02-08T12:30:23.705232"
}
```

Response structure:

Field	Type	Description
`versionId`	string	Application version from `AssemblyInformationalVersion`; may include build metadata with commit hash (e.g., `2.0.0+<git-commit-sha>`)
`buildDate`	DateTime	Application build date and time (ISO 8601 format)

Example call:

```
GET /api/version
```

```
curl http://localhost:5000/api/version
```

Windows PowerShell:

```
Invoke-RestMethod -Uri "http://localhost:5000/api/version"
```

Usage as Health Check: Method can be called from any HTTP client. Receiving status `200 OK` means the application is working correctly.

Docker HEALTHCHECK configuration:

```
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost:8080/api/version"]
  interval: 30s
  timeout: 3s
  retries: 3
  start_period: 10s
```

Checking container health status:

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

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

---

6. Background Services

The application automatically starts two background services (Hosted Services) on startup. These services run for the entire lifetime of the application and require no user interaction.

6.1. DownloadService

Description: Service that automatically downloads the daily VAT taxpayer whitelist data file from the Ministry of Finance server.

Schedule: Runs every **10 minutes** (first attempt immediately after application startup).

Data source: `https://plikplaski.mf.gov.pl/pliki/{yyyyMMdd}.7z`

Detailed operation:

1. Checks if the data file for the current date already exists in the `DataFiles` directory
2. If the file exists - operation is skipped (no logging)

3. If the file does not exist: a. Generates download URL: `https://plikplaski.mf.gov.pl/pliki/{yyyyMMdd}.7z` b. Downloads the 7z archive file (streaming to file in `Downloads` directory) c. Unpacks the archive to `DataFiles/{yyyyMMdd}/` directory d. Deletes the downloaded archive file from `Downloads` directory e. Sends notification about successful download (Email and/or Teams, if configured)

Error handling:

- On download failure (`HttpRequestException`) or timeout (`TaskCanceledException`), up to **3 retry attempts** are made with exponential backoff:
  - Attempt 1 → wait 2 seconds
  - Attempt 2 → wait 4 seconds
  - Attempt 3 → no retry
- If all 3 attempts fail:
  - If current hour is >= `AlertAfterHour` (default 10:00): an error notification is sent
  - Error notification is sent **at most once per day** (to avoid flooding the inbox)
  - Service continues attempting downloads every 10 minutes

Thread safety: The service uses a semaphore (`SemaphoreSlim`) to prevent parallel download operations.

Related configuration:

- `HttpClient:TimeoutMinutes` - download timeout (default 30 min)
- `Download:AlertAfterHour` - hour after which error notification is sent (default 10)

6.2. MaintenanceService

Description: Service that automatically cleans up old data files to prevent excessive disk usage.

Schedule: Runs every **12 hours** (first attempt immediately after application startup).

Detailed operation:

1. Calculates cutoff date: `current date - RetentionDays` (default 4 days)
2. Scans all subdirectories in the `DataFiles` folder

3. For each subdirectory: a. Parses directory name as date (`yyyyMMdd` format) b. If directory date is older than cutoff date - directory is deleted c. Logs information about deleted directories
4. Errors are logged but do not interrupt service operation

Related configuration:

- `Maintenance:RetentionDays` - retention period in days (default 4)

Example: With default configuration `RetentionDays=4` and current date 2026-02-08:

- Preserved directories: `20260208`, `20260207`, `20260206`, `20260205`
- Deleted directories: `20260204` and older

---

7. Notifications

The application supports two independent notification channels. Both channels can operate simultaneously.

Email

Email notifications are sent as HTML messages with event information. They are sent in the following situations:

Event	Description	Example content
File downloaded	Information about successful download of a new data file	"Pobrano nowy plik danych: 20260208.7z"
Download error	Information about missing data file after configured hour	"Brak pliku danych 20260208.7z po godzinie 10:00. Błąd: ..."

Email notifications contain:

- Message subject with event information
- Body with file name, date and time of the event
- Error content (for error notifications)
- Footer with application name

Configuration: see section 4.3.

Microsoft Teams

Teams notifications are sent as Adaptive Cards via Incoming Webhook. Cards contain:

Element	Description
Title	Event name
File	Data file name
Date/time	UTC timestamp of the event
Message	Event details or error message

Configuration: see section 4.4.

> **NOTE:** Download error notifications are sent at most **once per day** per channel, regardless of the number of failed attempts. Successful download notifications are sent for each download.

---

8. Data Volumes

The application uses three directories for data storage:

Volume	Container Path	Description	Size
DataFiles	`/app/DataFiles`	Extracted whitelist data files	~1GB per day
Downloads	`/app/Downloads`	Temporary 7z archive (deleted after unpacking)	~100MB temporarily
logs	`/app/logs`	Application log files (NLog)	Depends on logging level

Data directory structure:

```
DataFiles/
├── 20260205/
```

```
│   └── (whitelist data files)
├── 20260206/
│   └── (whitelist data files)
├── 20260207/
│   └── (whitelist data files)
└── 20260208/
    └── (whitelist data files)
```

Docker Compose - volume mounting:

```yaml
volumes:
  - ./DataFiles:/app/DataFiles
  - ./Downloads:/app/Downloads
  - ./logs:/app/logs
```

> **NOTE:** The `DataFiles` directory grows daily by ~1GB of new data files. Old files are automatically cleaned up by `MaintenanceService` according to the `RetentionDays` setting (default: 4 days). With the default setting, a minimum of ~5GB of free disk space is required.

---

9. Application Versions

Version History

Version 2.0.0 (current)

- Migration to IHost and modern .NET configuration
- Configuration extracted to `appsettings.json`
- Support for `appsettings.additional.json` (configuration override without Docker image changes)
- Email notifications - configurable SMTP server, multiple recipients, HTML messages
- Microsoft Teams notifications - Adaptive Cards via Incoming Webhook
- `DownloadService` - automatic file download with retry support (exponential backoff)
- `MaintenanceService` - automatic old data cleanup every 12 hours

- Download error notifications (once per day, after configured hour)
- `FrozenSet` optimization for O(1) hash lookup performance
- Upgrade to .NET 10.0
- Containerization with Docker/Podman (image available on GitHub Container Registry)
- Localization support (messages in PL and EN)
- Docker HEALTHCHECK on `/api/version` endpoint
- Non-root user in container for security

Version 1.x

- First production version
- Basic NIP and bank account verification against whitelist
- Manual data file download
- SHA-512 verification with bank account mask support

---

10. License

Last documentation update: 2026-02-08

Documentation version: 1.0

Compatible with application version: 2.0.0+