NL - Web API v7 Implementation Documentation Overview

Web API v7 Implementatie Documentatie

Inhoudsopgave

---

1. Introductie

Overzicht van de Software

De Web API versie 7 vertegenwoordigt een nieuw ontworpen online applicatie die het mogelijk maakt om geautomatiseerde productspecificaties op te halen en in te dienen. Deze versie introduceert belangrijke verbeteringen en nieuwe functionaliteiten:

- Ondersteuning voor zowel XML als JSON formaten
- Verbeterde performantie en schaalbaarheid
- Uitgebreide validatie mogelijkheden
- Geoptimaliseerde data-uitwisseling

Doelgroep

Deze documentatie is specifiek samengesteld voor:

- Producenten en groothandels die de API willen implementeren
- Afnemers die geautomatiseerde toegang tot productinformatie nodig hebben
- Technische teams verantwoordelijk voor de integratie
- Ontwikkelaars die de API gaan implementeren

---

2. Installatiehandleiding

Systeemvereisten

Voor gebruik van de Web API zijn de volgende minimale vereisten van toepassing:

- Een betrouwbare en stabiele internetverbinding
- Kennis van Endpoints en authenticatie afhandeling
- Toegang tot de specifieke URL of netwerk adres
- Geldige authenticatie gegevens

Basisinstellingen

1. Toegang tot de API omgeving verkrijgen via PS
2. Authenticatie gegevens configureren
3. Basis endpoints testen
4. Implementatie van error handling

---

3. Getting Started - Je Eerste API Call

Deze sectie leidt je stap-voor-stap door het proces van je eerste API call, van authenticatie tot het ophalen van een product. Ideaal voor ontwikkelaars die meteen aan de slag willen.

Wat heb je nodig?

Voordat je begint, zorg ervoor dat je het volgende hebt:

- **API Base URL**: `<https://webapi.psinfoodservice.com>`
- **Gebruikersnaam**: Ontvangen van PS in Foodservice
- **Wachtwoord**: Ontvangen van PS in Foodservice
- **Product ID**: Een geldige productsheet ID (psId) waartoe je toegang hebt
- **HTTP client**: cURL, Postman, of programmeercode in je favoriete taal

Stap 1: Inloggen en Token Verkrijgen

Om de API te kunnen gebruiken, heb je eerst een JWT token nodig. Dit token gebruik je voor alle verdere API calls.

Endpoint

```
POST https://webapi.psinfoodservice.com/v7/json/account/login
```

Request Headers

```
Content-Type: application/json
```

Request Body

```
{
  "username": "jouw_gebruikersnaam",
  "password": "jouw_wachtwoord"
}
```

Response Success - 200 OK

```
{
  "accesstoken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshtoken": "a8d9f7e6c5b4a3d2e1f0...",
  "expiresin": 3600
}
```

Belangrijk:

- De `accesstoken` is geldig voor het aantal seconden aangegeven in `expiresin` (standaard 3600 seconden = 1 uur)
- Bewaar de `refreshtoken` veilig - deze kun je gebruiken om een nieuw accesstoken aan te vragen zonder opnieuw in te loggen
- Bewaar tokens nooit in broncode of publieke repositories

Stap 2: Product Ophalen met Token

Nu je een token hebt, kun je productinformatie ophalen.

Endpoint

```
GET https://webapi.psinfoodservice.com/v7/json/productsheet/{psId}
```

of met specifieke taal:

```
GET
https://webapi.psinfoodservice.com/v7/json/productsheet/{language}/{psId}
```

Parameters:

- `{psId}`: Product ID (bijvoorbeeld: 1234567)
- `{language}`: (optioneel) Taalcode: `nl`, `en`, `de`, of `fr`

Request Headers

```
Authorization: Bearer {jouw_accesstoken}
Content-Type: application/json
```

Response Success - 200 OK

```
{
  "logistic": {
    "psId": 1234567,
    "gtince": "8712345678901",
    "gtinhe": "18712345678908",
    "productName": "Voorbeeld Product",
    ...
  },
  "product": {
    "articleNumber": "ART12345",
    "brand": "Voorbeeld Merk",
    ...
  },
  "specification": {
    ...
  }
}
```

Volledige Voorbeelden in Verschillende Talen

cURL (Command Line)

Stap 1: Login

```
curl -X POST https://webapi.psinfoodservice.com/v7/json/account/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "jouw_gebruikersnaam",
    "password": "jouw_wachtwoord"
  }'
```

Stap 2: Product ophalen

```
# Vervang {TOKEN} met de accesstoken uit stap 1
curl -X GET
https://webapi.psinfoodservice.com/v7/json/productsheet/1234567 \
  -H "Authorization: Bearer {TOKEN}" \
  -H "Content-Type: application/json"
```

C# (.NET 6+)

```
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Threading.Tasks;

public class Program
{
    private static readonly HttpClient client = new HttpClient();
    private const string BaseUrl =
"https://webapi.psinfoodservice.com/v7/json";

    public static async Task Main(string[] args)
    {
        try
        {
            // Stap 1: Login en token verkrijgen
            var loginData = new
            {
```

```
                username = "jouw_gebruikersnaam",
                password = "jouw_wachtwoord"
            };

            var loginResponse = await client.PostAsJsonAsync(
                $"{BaseUrl}/account/login",
                loginData
            );

            loginResponse.EnsureSuccessStatusCode();

            var tokenResponse = await
loginResponse.Content.ReadFromJsonAsync<TokenResponse>();
            Console.WriteLine($"Token verkregen, geldig voor
{tokenResponse.ExpiresIn} seconden");

            // Stap 2: Product ophalen met token
            client.DefaultRequestHeaders.Authorization =
                new AuthenticationHeaderValue("Bearer",
tokenResponse.AccessToken);

            int productId = 1234567;
            var productResponse = await client.GetAsync(
                $"{BaseUrl}/productsheet/{productId}"
            );

            productResponse.EnsureSuccessStatusCode();

            var productSheet = await
productResponse.Content.ReadAsStringAsync();
            Console.WriteLine("Product opgehaald:");
            Console.WriteLine(productSheet);
        }
        catch (HttpRequestException e)
        {
            Console.WriteLine($"Request fout: {e.Message}");
        }
    }
}

// DTOs
public class TokenResponse
{
    public string AccessToken { get; set; }
    public string RefreshToken { get; set; }
    public int ExpiresIn { get; set; }
}
```

JavaScript/Node.js (met fetch)

```
const baseUrl = 'https://webapi.psinfoodservice.com/v7/json';

async function getProduct() {
  try {
    // Stap 1: Login en token verkrijgen
    const loginResponse = await fetch(`${baseUrl}/account/login`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        username: 'jouw_gebruikersnaam',
        password: 'jouw_wachtwoord'
      })
    });

    if (!loginResponse.ok) {
      throw new Error(`Login failed: ${loginResponse.status}`);
    }

    const tokenData = await loginResponse.json();
    console.log(`Token verkregen, geldig voor ${tokenData.expiresin}
seconden`);

    // Stap 2: Product ophalen met token
    const productId = 1234567;
    const productResponse = await fetch(
      `${baseUrl}/productsheet/${productId}`,
      {
        method: 'GET',
        headers: {
          'Authorization': `Bearer ${tokenData.accesstoken}`,
          'Content-Type': 'application/json'
        }
      }
    );

    if (!productResponse.ok) {
```

```
      throw new Error(`Product ophalen mislukt: ${productResponse.status}`);
    }

    const productData = await productResponse.json();
    console.log('Product opgehaald:', productData);

    return productData;

  } catch (error) {
    console.error('Fout:', error.message);
    throw error;
  }
}

// Uitvoeren
getProduct();
```

Python (met requests library)

```
import requests
import json

BASE_URL = "https://webapi.psinfoodservice.com/v7/json"

def get_product():
    try:
        # Stap 1: Login en token verkrijgen
        login_data = {
            "username": "jouw_gebruikersnaam",
            "password": "jouw_wachtwoord"
        }

        login_response = requests.post(
            f"{BASE_URL}/account/login",
            json=login_data,
            headers={"Content-Type": "application/json"}
        )

        login_response.raise_for_status()
        token_data = login_response.json()

        access_token = token_data["accesstoken"]
        print(f"Token verkregen, geldig voor {token_data['expiresin']} seconden")

        # Stap 2: Product ophalen met token
        product_id = 1234567
        headers = {
            "Authorization": f"Bearer {access_token}",
            "Content-Type": "application/json"
        }

        product_response = requests.get(
            f"{BASE_URL}/productsheet/{product_id}",
            headers=headers
        )

        product_response.raise_for_status()
        product_data = product_response.json()

        print("Product opgehaald:")
        print(json.dumps(product_data, indent=2))

        return product_data

    except requests.exceptions.RequestException as e:
        print(f"Fout: {e}")
        raise

# Uitvoeren
if __name__ == "__main__":
    get_product()
```

PHP

```
<?php

$baseUrl = "https://webapi.psinfoodservice.com/v7/json";

function getProduct() {
    global $baseUrl;

    try {
        // Stap 1: Login en token verkrijgen
        $loginData = [
            'username' => 'jouw_gebruikersnaam',
            'password' => 'jouw_wachtwoord'
```

```php
        ];

        $loginOptions = [
            'http' => [
                'method' => 'POST',
                'header' => 'Content-Type: application/json',
                'content' => json_encode($loginData)
            ]
        ];

        $loginContext = stream_context_create($loginOptions);
        $loginResponse = file_get_contents(
            "$baseUrl/account/login",
            false,
            $loginContext
        );

        if ($loginResponse === false) {
            throw new Exception("Login failed");
        }

        $tokenData = json_decode($loginResponse, true);
        $accessToken = $tokenData['accesstoken'];

        echo "Token verkregen, geldig voor {$tokenData['expiresin']} seconden\n";

        // Stap 2: Product ophalen met token
        $productId = 1234567;

        $productOptions = [
            'http' => [
                'method' => 'GET',
                'header' => "Authorization: Bearer $accessToken\r\n" .
                            "Content-Type: application/json"
            ]
        ];

        $productContext = stream_context_create($productOptions);
        $productResponse = file_get_contents(
            "$baseUrl/productsheet/$productId",
            false,
            $productContext
        );

        if ($productResponse === false) {
            throw new Exception("Product ophalen mislukt");
        }

        $productData = json_decode($productResponse, true);

        echo "Product opgehaald:\n";
        echo json_encode($productData, JSON_PRETTY_PRINT) . "\n";

        return $productData;

    } catch (Exception $e) {
        echo "Fout: " . $e->getMessage() . "\n";
        throw $e;
    }
}

// Uitvoeren
getProduct();

?>
```

XML Format Gebruiken

Als je XML formaat wilt gebruiken in plaats van JSON, vervang dan `/json/` door `/xml/` in de URLs:

Login (XML):

```
POST https://webapi.psinfoodservice.com/v7/xml/account/login
Content-Type: application/json
```

Product ophalen (XML):

```
GET https://webapi.psinfoodservice.com/v7/xml/productsheet/1234567
Authorization: Bearer {token}
```

Let op: De request body voor login blijft JSON, maar de response zal in XML formaat zijn.

Token Vernieuwen (Refresh Token)

Wanneer je accesstoken bijna verlopen is, kun je een nieuw token aanvragen zonder opnieuw in te loggen:

Endpoint

```
POST https://webapi.psinfoodservice.com/v7/json/account/refreshtoken
```

Request Body

```
{
  "accessToken": "je_huidige_accesstoken",
  "refreshToken": "je_refreshtoken"
}
```

Response

```
{
  "accesstoken": "nieuw_accesstoken",
  "refreshtoken": "nieuw_refreshtoken",
  "expiresin": 3600
}
```

Troubleshooting

Probleem: 401 Unauthorized bij login

Mogelijke oorzaken:

- Incorrecte gebruikersnaam of wachtwoord
- IP adres is geblokkeerd na meerdere foutieve login pogingen (24 uur blokkade)
- Account is niet actief

Oplossing:

- Controleer je credentials
- Wacht 24 uur als je IP geblokkeerd is
- Neem contact op met support: info@psinfoodservice.com

Probleem: 401 Unauthorized bij product ophalen

Mogelijke oorzaken:

- Token is verlopen
- Token is niet correct opgenomen in de Authorization header
- Formaat van de Authorization header is incorrect

Oplossing:

```
# Correcte format:
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

# NIET:
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Probleem: 403 Forbidden bij product ophalen

Mogelijke oorzaken:

- Je hebt geen toegang tot dit specifieke product
- Product ID bestaat niet
- Je account heeft niet de juiste permissies

Oplossing:

- Controleer of het product ID correct is
- Controleer of je account toegang heeft tot dit product
- Neem contact op met support voor toegangsvragen

Probleem: 404 Not Found

Mogelijke oorzaken:

- Product bestaat niet in het systeem
- Verkeerd endpoint gebruikt
- Type fout in URL

Oplossing:

- Controleer de URL spelling
- Verifieer dat het product ID bestaat
- Controleer de API documentatie voor correcte endpoints

Probleem: 400 Bad Request

Mogelijke oorzaken:

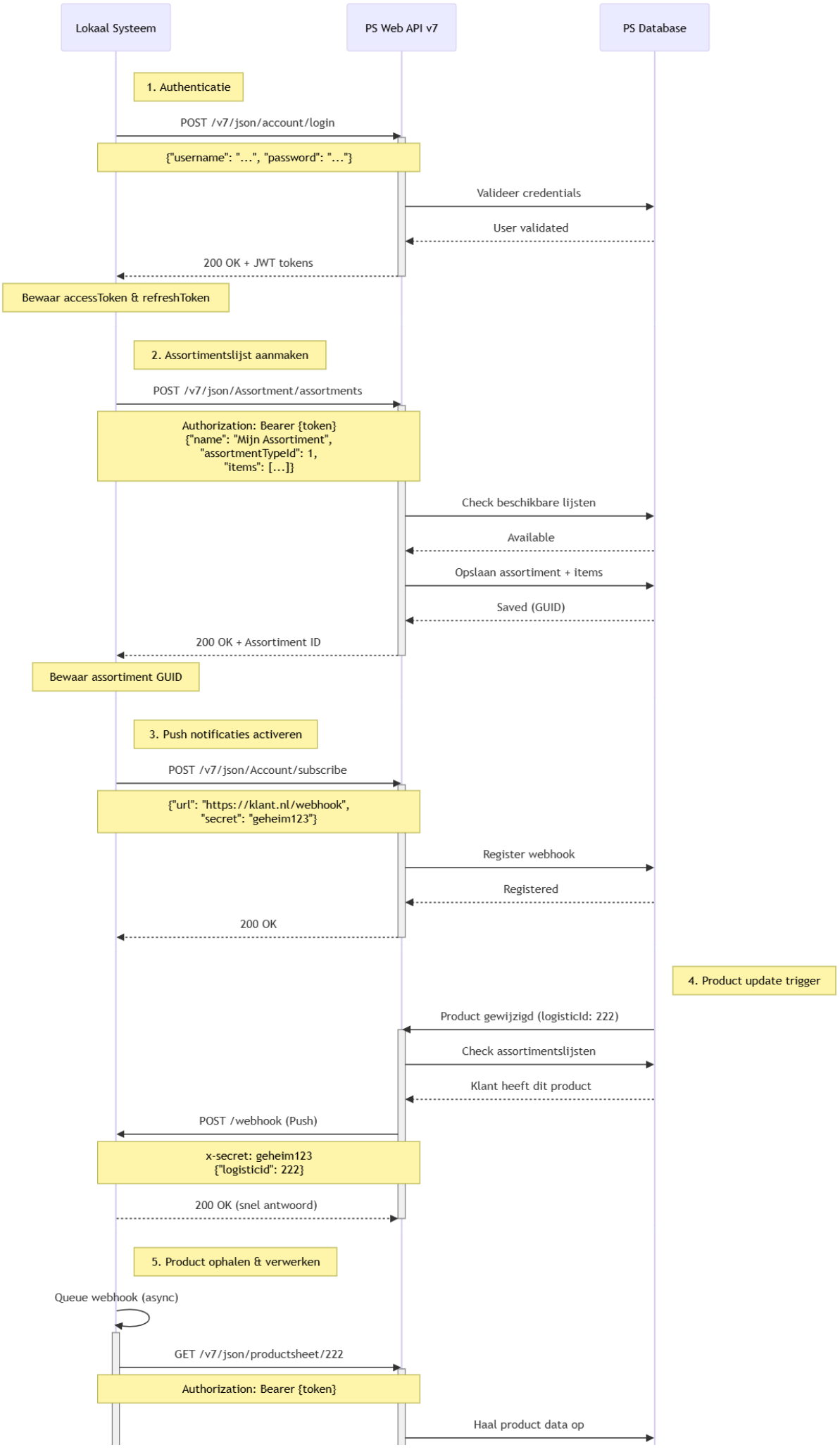
- Ongeldige JSON in request body
- Missende verplichte velden
- Verkeerde data types
- Ongeldige taalcode (moet zijn: nl, en, de, of fr)

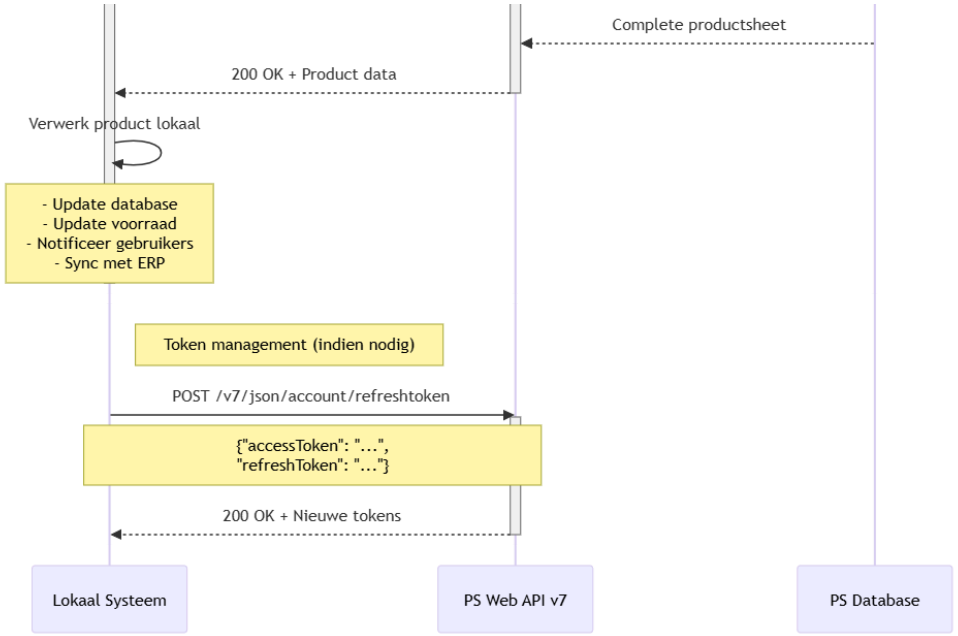
Oplossing:

```
// Correct login request:
{
  "username": "string",
  "password": "string"
}

// Incorrecte voorbeelden (geen lege strings toegestaan):
{
  "username": "",
  "password": "test"
}
```

Best Practices voor Getting Started


Lokaal Systeem
PS Web API v7
PS Database
1. Authenticatie
POST /v7/json/account/login
{"username": "...", "password": "..."}
Valideer credentials
User validated
200 OK + JWT tokens
Bewaar accessToken & refreshToken
2. Assortimentslijst aanmaken
POST /v7/json/Assortment/assortments
Authorization: Bearer {token}
{"name": "Mijn Assortiment",
"assortmentTypeId": 1,
"items": [...]}
Check beschikbare lijsten
Available
Opslaan assortiment + items
Saved (GUID)
200 OK + Assortiment ID
Bewaar assortiment GUID
3. Push notificaties activeren
POST /v7/json/Account/subscribe
{"url": "https://klant.nl/webhook",
"secret": "geheim123"}
Register webhook
Registered
200 OK
4. Product update trigger
Product gewijzigd (logisticId: 222)
Check assortimentslijsten
Klant heeft dit product
POST /webhook (Push)
x-secret: geheim123
{"logisticid": 222}
200 OK (snel antwoord)
5. Product ophalen & verwerken
Queue webhook (async)
GET /v7/json/productsheet/222
Authorization: Bearer {token}
Haal product data op



1. **Token Management**
   - Sla tokens veilig op (bijv. environment variables, secure storage)
   - Implementeer token refresh logic voor verlopen tokens
   - Log tokens NOOIT in console of log bestanden
2. **Error Handling**
   - Implementeer try-catch blokken voor alle API calls
   - Handle verschillende HTTP status codes afzonderlijk
   - Log errors voor debugging, maar zonder gevoelige informatie
3. **Testing**
   - Test eerst met Postman of cURL voordat je code schrijft
   - Gebruik de Swagger documentatie voor endpoint details
   - Test zowel success als error scenarios
4. **Rate Limiting**
   - De API heeft rate limiting per endpoint
   - Implementeer exponential backoff bij rate limit errors
   - Cache resultaten waar mogelijk
5. **Productie Deployment**
   - Gebruik HTTPS altijd (nooit HTTP)
   - Bewaar credentials in omgevingsvariabelen
   - Implementeer logging en monitoring
   - Test grondig in staging omgeving

Volgende Stappen

Nu je je eerste API call succesvol hebt uitgevoerd, kun je verder met:

- **Sectie 5**: Leer meer over alle beschikbare endpoints
- **Sectie 6**: Ontdek productsheet functionaliteit in detail
- **Sectie 9**: Implementeer assortimentslijsten
- **Sectie 10**: Configureer webhooks voor real-time notificaties
- **Sectie 12**: Best practices voor productie gebruik

Voor meer gedetailleerde informatie over specifieke endpoints, zie de Swagger documentatie:
Swagger UI

---

4. Authenticatie & Beveiliging

Authenticatie Flow

Het authenticatie proces verloopt via de volgende stappen:

1. Login met gebruikersnaam/wachtwoord
2. Ontvangst van JWT token
3. Token gebruiken voor vervolg aanroepen

Autorisatie token

Tokens bevatten de volgende informatie:

- Gebruikersidentiteit
- Machtigingen
- Geldigheidsduur

Belangrijk: Tokens hebben een beperkte geldigheidsduur en moeten worden vernieuwd via het authenticatie proces.

---

5. API Endpoints & Functionaliteit

Beschikbare Endpoints

Basis URL structuur:

```
https://webapi.psinfoodservice.com/v7/{format}/{endpoint}/{parameters}
```

Waar:

- `{format}` = 'json' of 'xml'
- `{endpoint}` = specifieke functionaliteit
- `{parameters}` = aanvullende parameters

Belangrijke Endpoints

Productsheet

```
GET /ProductSheet/{id}
GET /ProductSheet/{language}/{id}
```

Master Data

```
GET /master/all
GET /master/{type}
```

Updates

```
POST /Update/EAN
```

Mijn Producten

```
GET /myproducts
```

Data Formaten

XML Formaat

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <logistic>
```

```
    <!-- Logistieke gegevens -->
  </logistic>
  <product>
    <!-- Algemene productgegevens -->
  </product>
  <specification>
    <!-- Product specificaties (optioneel) -->
  </specification>
</root>
```

JSON Formaat

```
{
  "logistic": {
    // Logistieke gegevens
  },
  "product": {
    // Algemene productgegevens
  },
  "specification": {
    // Product specificaties (optioneel)
  }
}
```

Request & Response Voorbeelden

Updates Controleren

Request:

```
{
  "searchCriteria": [
    "87000000000",
    "87000000001",
    "87000000002"
  ],
  "lastUpdatedAfter": "2023-12-22",
  "isPrivateLabel": true,
  "isPubliclyVisible": true,
  "targetMarket": "All"
}
```

Response:

```
[
  {
    "LogisticId": 1690881,
    "TargetMarketId": 1,
    "LastChanged": "2023-08-24T12:53:24.823"
  }
]
```

---

6. Productsheet Functionaliteit

Structuur

De productsheet bestaat uit drie hoofdsecties:

1. Logistiek gedeelte (verplicht)

- Verpakkingsgegevens
- Logistieke informatie

2. Productgedeelte (verplicht)

- Algemene productgegevens
- Productidentificatie

3. Specificatiegedeelte (optioneel)

- Food/Non-food specifieke gegevens
- Aanvullende productinformatie

Taalondersteuning

- Standaard: alle beschikbare talen
- Specifieke taal via URL parameter
- Vertaalde velden via masterlijsten

Output Opties

- **all**: complete productinformatie
- **summary**: beknopte samenvatting
- **productcontent**: alleen productinhoud
- **logistics**: alleen logistieke gegevens

---

7. Updates & Synchronisatie

Updates Controleren

Het update proces omvat:

1. Controle op wijzigingen sinds datum
2. Vergelijking met bestaande data
3. Verwerking van verschillen
4. Terugkoppeling resultaten

Data Insturen

Bij het insturen van data wordt het volgende gecontroleerd:

- Validiteit van de data
- Referentiële integriteit
- Verplichte velden
- Datatype correctheid

Validatie Proces

1. Syntactische validatie
2. Semantische validatie
3. Business rules validatie
4. Master data referentie controle

---

8. Master Data Management

Beschikbare Masterlijsten

- Eenheden
- Allergenen
- Landen
- Vertalingen
- Verpakkingsmaterialen
- Aanvullende productdetails

Gebruik en Implementatie

1. Ophalen van masterlijsten
2. Implementatie van updates
3. Verwerking in eigen systeem

4. Synchronisatie strategie

Caching Strategie

- Implementeer lokale cache
- Regelmatige updates
- Versie controle
- Cache invalidatie

---

9. Assortimentslijsten

Doel en Functionaliteit

Assortimentslijsten maken het mogelijk om:

- Producten te koppelen aan afnemers
- Assortiment te beheren
- Relaties te leggen tussen leveranciers en afnemers
- Artikelen gestructureerd te beheren in categorieën

Klanten kunnen meerdere assortimentslijsten aanmaken en beheren, artikelen toevoegen en verwijderen, en assortimenten opvragen en bekijken.

API Endpoints

Alle assortimentslijsten endpoints vereisen authenticatie met JWT token en de Read rol. De endpoints filteren automatisch op basis van de RelationId uit het token.

9.1 POST /v7/{format}/Assortment/assortments

Een nieuwe assortimentslijst aanmaken

Maakt een nieuwe assortimentslijst aan met optioneel items. Controleert eerst of er nog beschikbare lijsten zijn voor de relatie.

Parameters:

- `{format}` : Responseformaat (json/xml)

Request Body:

```
{
  "id": "guid",
  "name": "string (verplicht)",
  "assortmentTypeId": "int (verplicht, > 0)",
  "items": [
    {
      "id": 0,
      "articleNumber": "string",
      "articleName": "string",
      "articleBrand": "string",
      "gtince": "string",
      "gtinhe": "string",
      "relationGln": "string",
      "relationName": "string",
      "relationArticleNumber": "string"
    }
  ]
}
```

Response Codes:

- **200 OK**: Assortimentslijst succesvol aangemaakt
- **400 Bad Request**: Ongeldige input data
- **409 Conflict**: "No available lists" - geen beschikbare lijsten meer

Implementatie Details:

- Controller: `AssortmentController.Assortments()` (regel 57-75)
- Use Case: `ISetAssortmentUseCase`
- Request Size Limit: 100 MB

9.2 GET /v7/{format}/Assortment/assortments

Alle assortimentslijsten ophalen

Haalt alle assortimentslijsten op die horen bij de ingelogde relatie, inclusief alle items per lijst.

Parameters:

- `{format}`: Responseformaat (json/xml)

Response Body:

```
[
  {
    "id": "guid",
    "name": "string",
    "assortmentType": {
      "id": 0,
      "name": [
        {
          "language": "string",
          "value": "string"
        }
      ]
    },
    "items": [
      {
        "id": 0,
        "articleNumber": "string",
        "articleName": "string",
        "articleBrand": "string",
        "gtince": "string",
        "gtinhe": "string",
        "relationGln": "string",
        "relationName": "string",
        "relationArticleNumber": "string"
      }
    ]
  }
]
```

Response Codes:

- **200 OK**: Lijst met assortimenten
- **204 No Content**: Geen assortimenten gevonden

Implementatie Details:

- Controller: `AssortmentController.GetAssortmentLists()` (regel 157-167)
- Use Case: `IGetAssortmentListsUseCase`

9.3 PUT /v7/{format}/Assortment/assortments/items

Items toevoegen aan een assortimentslijst

Voegt nieuwe items toe aan een bestaande assortimentslijst. Bestaande items blijven behouden.

Parameters:

- `{format}`: Responseformaat (json/xml)

Request Body:

```
{
  "id": "guid (verplicht)",
  "items": [
    {
```

```
      "id": 0,
      "articleNumber": "string",
      "articleName": "string",
      "articleBrand": "string",
      "gtince": "string",
      "gtinhe": "string",
      "relationGln": "string",
      "relationName": "string",
      "relationArticleNumber": "string"
    }
  ]
}
```

Response Codes:

- **200 OK**: Items succesvol toegevoegd
- **400 Bad Request**: Ongeldige input data
- **409 Conflict**: "Geen assortimentslijst met de Id {id} gevonden."

Implementatie Details:

- Controller: `AssortmentController.AddAssortmentItems()` (regel 81-99)
- Use Case: `IAddAssortmentItemsUseCase`
- Request Size Limit: 100 MB

9.4 DELETE /v7/{format}/Assortment/assortments/{id}/items

Items verwijderen uit een assortimentslijst

Verwijdert specifieke items uit een bestaande assortimentslijst. De lijst zelf blijft bestaan.

Parameters:

- `{format}`: Responseformaat (json/xml)
- `{id}`: GUID van de assortimentslijst (route parameter, verplicht)

Request Body:

```
{
  "id": "guid",
  "items": [
    {
      "id": 0,
      "articleNumber": "string",
      "articleName": "string",
      "articleBrand": "string",
      "gtince": "string",
      "gtinhe": "string",
      "relationGln": "string",
      "relationName": "string",
      "relationArticleNumber": "string"
    }
  ]
}
```

Response Codes:

- **200 OK**: Items succesvol verwijderd
- **400 Bad Request**: Ongeldige input data
- **409 Conflict**: "Geen assortimentslijst met de naam {name} gevonden."

Implementatie Details:

- Controller: `AssortmentController.RemoveAssortmentItems()` (regel 104-126)
- Use Case: `IRemoveAssortmentItemsUseCase`

9.5 GET /v7/{format}/Assortment/assortments/{id}/items

Een specifieke assortimentslijst ophalen

Haalt een specifieke assortimentslijst op inclusief alle items en metadata.

Parameters:

- `{format}` : Responseformaat (json/xml)
- `{id}` : GUID van de assortimentslijst (verplicht)

Response Body:

```
{
  "id": "guid",
  "name": "string",
  "assortmentType": {
    "id": 0,
    "name": [
      {
        "language": "string",
        "value": "string"
      }
    ]
  },
  "items": [
    {
      "id": 0,
      "articleNumber": "string",
      "articleName": "string",
      "articleBrand": "string",
      "gtince": "string",
      "gtinhe": "string",
      "relationGln": "string",
      "relationName": "string",
      "relationArticleNumber": "string"
    }
  ]
}
```

Response Codes:

- **200 OK**: Assortimentslijst gevonden
- **204 No Content**: Geen assortimentslijst gevonden met dit ID

Implementatie Details:

- Controller: `AssortmentController.GetAssortmentList()` (regel 141-152)
- Use Case: `IGetAssortmentUseCase`

9.6 DELETE /v7/{format}/Assortment/assortment/{id}

Een volledige assortimentslijst verwijderen

Verwijdert een volledige assortimentslijst inclusief alle items permanent.

Parameters:

- `{format}` : Responseformaat (json/xml)
- `{id}` : GUID van de assortimentslijst (verplicht)

Response Codes:

- **200 OK**: Assortimentslijst succesvol verwijderd
- **400 Bad Request**: Ongeldig ID
- **401 Unauthorized**: Niet geautoriseerd
- **403 Forbidden**: Geen toegang tot deze lijst

Implementatie Details:

- Controller: `AssortmentController.RemoveAssortment()` (regel 131-136)
- Use Case: `IRemoveAssortmentUseCase`

Data Models

AssortmentDto (Output)

Representeert een complete assortimentslijst met alle metadata en items.

Structuur:

```
public class AssortmentDto {
    public Guid Id { get; set; }
    public string Name { get; set; }
    public AssortmentTypeDto AssortmentType { get; set; }
    public List<AssortmentItemDto> Items { get; set; }
}
```

Velden:

- **Id**: Unieke identifier van de assortimentslijst
- **Name**: Naam van de assortimentslijst
- **AssortmentType**: Type/categorie van het assortiment
- **Items**: Lijst van artikelen in dit assortiment

AssortmentInsertDto (Input)

Voor het aanmaken van een nieuwe assortimentslijst.

Structuur:

```
public class AssortmentInsertDto {
    public Guid Id { get; set; }
    public string Name { get; set; }
    public int AssortmentTypeId { get; set; }
    public List<AssortmentItemUpdateDto> Items { get; set; }
}
```

Velden:

- **Id**: Optionele GUID (wordt gegenereerd indien niet meegegeven)
- **Name**: Naam van de assortimentslijst (verplicht)
- **AssortmentTypeId**: ID van het assortimentstype (verplicht, > 0)
- **Items**: Lijst van items om toe te voegen (optioneel)

AssortmentItemDto

Representeert een artikel binnen een assortimentslijst.

Structuur:

```
public class AssortmentItemDto {
    public int Id { get; set; }
    public string ArticleNumber { get; set; }
    public string ArticleName { get; set; }
    public string ArticleBrand { get; set; }
    public string GTINCE { get; set; }
    public string GTINHE { get; set; }
    public string RelationGln { get; set; }
    public string RelationName { get; set; }
    public string RelationArticleNumber { get; set; }
}
```

Velden:

- **Id**: Interne item ID
- **ArticleNumber**: Artikelnummer van de leverancier
- **ArticleName**: Naam van het artikel
- **ArticleBrand**: Merk van het artikel
- **GTINCE**: GTIN van de consumer eenheid (EAN)
- **GTINHE**: GTIN van de handelseenheid

- **RelationGln**: GLN van de gerelateerde partij
- **RelationName**: Naam van de gerelateerde partij
- **RelationArticleNumber**: Artikelnummer bij de relatie

AssortmentTypeDto

Representeert een type/categorie assortiment met meertalige namen.

Structuur:

```
public class AssortmentTypeDto {
    public int Id { get; set; }
    public List<TranslationDto> Name { get; set; }
}
```

Velden:

- **Id**: Unieke identifier van het type
- **Name**: Lijst van vertalingen van de typenaam

Implementatie

Stappen voor implementatie:

1. **Lijst samenstellen**
   - Bepaal het type assortiment (AssortmentTypeId)
   - Verzamel artikelgegevens
   - Valideer verplichte velden
2. **Lijst aanmaken**
   - POST naar /v7/json/Assortment/assortments
   - Controleer beschikbaarheid van lijsten
   - Bewaar het gegenereerde ID
3. **Items beheren**
   - Voeg items toe via PUT /assortments/items
   - Verwijder items via DELETE /assortments/{id}/items
   - Update items door verwijderen en opnieuw toevoegen
4. **Lijst opvragen**
   - GET voor alle lijsten: /v7/json/Assortment/assortments
   - GET voor specifieke lijst: /v7/json/Assortment/assortments/{id}/items
5. **Lijst verwijderen**
   - DELETE voor volledige lijst: /v7/json/Assortment/assortment/{id}

Verwerking

Het systeem verwerkt de assortimentslijsten door:

1. **Validatie**
   - Controle op verplichte velden (Name, AssortmentTypeId)
   - Validatie van data types
   - Controle op beschikbare lijsten
2. **Vergelijking met bestaande data**
   - Identificatie van nieuwe artikelen
   - Matching met bestaande producten
   - Detectie van duplicaten
3. **Relaties verwerken**

- Koppeling van artikelen aan relaties
- Bijwerken van relatie-informatie
- Synchronisatie met master data

4. **Terugkoppeling**
   - Success/error response codes
   - Gedetailleerde foutmeldingen
   - Logging van bewerkingen

Authenticatie & Autorisatie

Vereisten voor alle endpoints:

- JWT Token in Authorization header
- Rol: Read rol vereist
- Token bevat: RelationId en UserId voor filtering

Token structuur:

```
Authorization: Bearer <jwt-token>
```

De RelationId uit het token wordt gebruikt om:

- Toegang te beperken tot eigen assortimenten
- Aantal beschikbare lijsten te controleren
- Data te filteren per relatie

Rate Limiting & Size Limits

POST en PUT endpoints:

- Request size limit: **100 MB**
- Maakt grote bulk uploads mogelijk
- Geschikt voor complete assortimentslijsten

Aanbevelingen:

- Splits zeer grote lijsten (> 10.000 items) in meerdere batches
- Implementeer error handling voor timeouts
- Monitor response tijden

Gebruik Tips & Best Practices

Nieuwe lijst aanmaken:

1. Gebruik eerst POST om een lijst aan te maken
2. Bewaar het gegenereerde ID voor latere referentie
3. Valideer het response voor confirmatie

Items beheren:

1. Gebruik PUT om items incrementeel toe te voegen
2. Gebruik DELETE /items om specifieke items te verwijderen
3. Voor bulk updates: verwijder oude items en voeg nieuwe toe

Lijsten ophalen:

1. Gebruik GET zonder {id} voor overzicht van alle lijsten
2. Gebruik GET met {id} voor details van specifieke lijst

3. Cache resultaten lokaal indien mogelijk

Lijst verwijderen:

1. Gebruik DELETE /{id} om de volledige lijst te verwijderen
2. Let op: dit is een permanente actie
3. Implementeer confirmatie in user interface

Foutafhandeling:

1. Implementeer retry logic voor netwerk fouten
2. Log error responses voor debugging
3. Valideer input data voor verzending
4. Check beschikbare lijsten voor aanmaken

Performance:

1. Batch items waar mogelijk (tot 100 MB)
2. Gebruik pagination voor grote lijsten
3. Implementeer caching van masterdata
4. Monitor API call frequentie

---

10. Webhooks

Overzicht

Webhooks bieden de mogelijkheid om real-time notificaties te ontvangen wanneer producten in uw assortimentslijst worden gewijzigd. In plaats van periodiek de API te pollen voor updates, ontvangt u automatisch een push bericht zodra er een wijziging plaatsvindt.

Hoe werkt het?

Wanneer een product uit uw assortimentslijst wordt geüpdatet, stuurt de Web API automatisch een POST request naar uw geregistreerde webhook URL.

Webhook Endpoints

Subscribe - Aanmelden voor push berichten

Endpoint:

```
POST /v7/{format}/Account/subscribe
```

Beschrijving:

Registreer een webhook URL om notificaties te ontvangen bij productwijzigingen.

Parameters:

- `{format}`: Responseformaat (json/xml)

Request Body:

```
{
  "url": "https://uw-domein.nl/webhook/productupdate",
  "secret": "uw-geheime-sleutel"  // optioneel
}
```

Response Codes:

- **200 OK**: Webhook succesvol geregistreerd
- **400 Bad Request**: Ongeldige URL of request

- **401 Unauthorized**: Niet geautoriseerd

Unsubscribe - Afmelden voor push berichten

Endpoint:

```
POST /v7/{format}/Account/unsubscribe
```

Beschrijving:

Verwijder de geregistreerde webhook URL om geen notificaties meer te ontvangen.

Parameters:

- `{format}` : Responseformaat (json/xml)

Response Codes:

- **200 OK**: Webhook succesvol verwijderd
- **400 Bad Request**: Geen webhook geregistreerd
- **401 Unauthorized**: Niet geautoriseerd

Webhook Payload

Wanneer een product in uw assortimentslijst wordt gewijzigd, ontvangt u een POST request op uw geregistreerde webhook URL met de volgende payload:

Request Method: POST

Content-Type: application/json

Headers:

```
{
  "x-secret": "uw-geheime-sleutel" // indien secret opgegeven bij subscribe
}
```

Body:

```
{
  "logisticid": 222
}
```

Gebruik van Secret

Wanneer u bij subscribe een `secret` opgeeft, wordt deze waarde in de `x-secret` header van elk push bericht meegestuurd. Gebruik dit om de authenticiteit van inkomende webhook berichten te verifiëren.

Verwerking van Webhook Berichten

Na ontvangst van een webhook bericht kunt u de volledige productgegevens ophalen met het ontvangen `logisticid` :

Stap 1: Ontvang webhook notificatie

```
{
  "logisticid": 222
}
```

Stap 2: Haal productgegevens op via de API

```
GET https://webapi.psinfoodservice.com/v7/json/productsheet/{logisticid}
Authorization: Bearer {your_accesstoken}
```

Implementatie Voorbeeld

Webhook Endpoint (C# / ASP.NET Core, an open-source web development framework | .NET Core)

```csharp
[ApiController]
[Route("webhook")]
public class WebhookController : ControllerBase
{
    private readonly IProductService _productService;
    private readonly ILogger<WebhookController> _logger;

    public WebhookController(IProductService productService, ILogger<WebhookController> logger)
    {
        _productService = productService;
        _logger = logger;
    }

    [HttpPost("productupdate")]
    public async Task<IActionResult> HandleProductUpdate([FromBody] WebhookPayload payload)
    {
        // Valideer webhook secret
        if (!ValidateWebhookSecret())
        {
            _logger.LogWarning("Webhook ontvangen met ongeldige secret");
            return Unauthorized();
        }

        _logger.LogInformation($"Webhook ontvangen voor logisticid: {payload.LogisticId}");

        try
        {
            // Haal de geüpdatete productgegevens op
            var product = await _productService.GetProductByIdAsync(payload.LogisticId);

            // Verwerk de update in uw systeem
            await _productService.ProcessProductUpdateAsync(product);

            return Ok();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"Fout bij verwerken webhook voor logisticid: {payload.LogisticId}");
            return StatusCode(500);
        }
    }

    private bool ValidateWebhookSecret()
    {
        // Haal de verwachte secret op uit configuratie
        var expectedSecret = _configuration["Webhook:Secret"];

        if (string.IsNullOrEmpty(expectedSecret))
        {
            _logger.LogWarning("Geen webhook secret geconfigureerd");
            return true; // Optioneel: sta requests toe als geen secret is geconfigureerd
        }

        // Haal de secret uit de header
        if (!Request.Headers.TryGetValue("x-secret", out var receivedSecret))
        {
            return false;
        }

        // Vergelijk secrets (gebruik constant-time vergelijking voor beveiliging)
        return CryptographicOperations.FixedTimeEquals(
            Encoding.UTF8.GetBytes(expectedSecret),
            Encoding.UTF8.GetBytes(receivedSecret.ToString())
        );
    }
}

public class WebhookPayload
{
    public int LogisticId { get; set; }
}
```

Webhook Endpoint (Node.js / Express)

```javascript
const express = require('express');
```

```
const crypto = require('crypto');
const app = express();

app.use(express.json());

// Configuratie
const WEBHOOK_SECRET = process.env.WEBHOOK_SECRET || 'uw-geheime-sleutel';
const ACCESS_TOKEN = process.env.ACCESS_TOKEN;

// Middleware voor webhook secret verificatie
function validateWebhookSecret(req, res, next) {
    const receivedSecret = req.headers['x-secret'];

    if (!WEBHOOK_SECRET) {
        console.warn('Geen webhook secret geconfigureerd');
        return next(); // Optioneel: sta requests toe als geen secret is geconfigureerd
    }

    if (!receivedSecret) {
        console.warn('Webhook ontvangen zonder secret header');
        return res.status(401).send('Unauthorized');
    }

    // Gebruik constant-time vergelijking voor beveiliging
    const expectedBuffer = Buffer.from(WEBHOOK_SECRET);
    const receivedBuffer = Buffer.from(receivedSecret);

    if (expectedBuffer.length !== receivedBuffer.length ||
        !crypto.timingSafeEqual(expectedBuffer, receivedBuffer)) {
        console.warn('Webhook ontvangen met ongeldige secret');
        return res.status(401).send('Unauthorized');
    }

    next();
}

app.post('/webhook/productupdate', validateWebhookSecret, async (req, res) => {
    const { logisticid } = req.body;

    console.log(`Webhook ontvangen voor logisticid: ${logisticid}`);

    try {
        // Haal de geüpdatete productgegevens op via de API
        const productResponse = await fetch(
`https://webapi.psinfoodservice.com/v7/json/productsheet/${logisticid}`,
            {
                headers: {
                    'Authorization': `Bearer ${ACCESS_TOKEN}`,
                    'Content-Type': 'application/json'
                }
            }
        );

        if (!productResponse.ok) {
            throw new Error(`API error: ${productResponse.status}`);
        }

        const productData = await productResponse.json();

        // Verwerk de update in uw systeem
        await processProductUpdate(productData);

        res.status(200).send('OK');
    } catch (error) {
        console.error(`Fout bij verwerken webhook: ${error.message}`);
        res.status(500).send('Error');
    }
});

app.listen(3000, () => {
    console.log('Webhook server draait op poort 3000');
});

async function processProductUpdate(productData) {
    // Implementeer hier uw eigen logica
    console.log('Product update verwerken:', productData);
}
```

Environment variables (.env bestand):
```
WEBHOOK_SECRET=uw-geheime-sleutel
ACCESS_TOKEN=uw-access-token
```

Best Practices voor Webhooks

1. **Beveiliging**
   - Gebruik HTTPS voor uw webhook endpoint
   - Valideer binnenkomende requests
   - Implementeer rate limiting op uw endpoint
2. **Betrouwbaarheid**
   - Retourneer snel een 200 OK response
   - Verwerk de daadwerkelijke update asynchroon
   - Implementeer een queue voor binnenkomende webhooks
3. **Foutafhandeling**
   - Log alle binnenkomende webhook requests
   - Implementeer retry logic bij mislukte API calls
   - Monitor uw webhook endpoint voor fouten
4. **Idempotentie**
   - Ontwerp uw verwerking idempotent (dezelfde update meerdere keren verwerken geeft hetzelfde resultaat)
   - Houd bij welke updates al verwerkt zijn
5. **Timeout Handling**
   - Reageer binnen 30 seconden op webhook requests
   - Gebruik background processing voor langdurige operaties

---

11. Error Handling

HTTP Status Codes

Code	Betekenis	Wanneer
200	OK	Succesvolle operatie
204	No Content	Geen data gevonden
400	Bad Request	Ongeldige aanvraag of parameters
401	Unauthorized	Niet geautoriseerd, token ontbreekt/ongeldig
403	Forbidden	Toegang geweigerd, onvoldoende rechten
404	Not Found	Resource niet gevonden
409	Conflict	Conflict met bestaande data
500	Internal Server Error	Server fout

Foutmeldingen

Foutmeldingen bevatten:

- Unieke foutcode

- Beschrijvende melding in duidelijke taal
- Aanvullende details over de fout
- Logging referentie voor support

Voorbeeld error response:

```
{
  "error": "ValidationError",
  "message": "Geen assortimentslijst met de Id {guid} gevonden.",
  "statusCode": 409,
  "timestamp": "2024-01-15T10:30:00Z"
}
```

Troubleshooting

Veel voorkomende problemen:

1. Connectie problemen

- Controleer netwerk connectiviteit
- Verificeer SSL/TLS configuratie
- Check firewall settings
- Test met ping/traceroute

2. Authenticatie problemen

- Controleer API key/token validiteit
- Verificeer verlopen credentials
- Check IP whitelist settings
- Valideer gebruikersrechten

3. Validatie fouten

- Controleer verplichte velden
- Valideer data types
- Check data formaat (JSON/XML)
- Verify character encoding

4. Rate limiting

- Monitor aantal API calls
- Implementeer backoff strategie
- Gebruik caching waar mogelijk
- Optimaliseer batch sizes

---

12. Best Practices

Implementatie Richtlijnen

1. **Gebruik de juiste endpoints**
   - Kies het juiste HTTP method (GET/POST/PUT/DELETE)
   - Gebruik correcte URL structuur
   - Specificeer gewenst format (json/xml)
2. **Implementeer error handling**
   - Try-catch blokken
   - Retry logic voor netwerk fouten
   - Logging van errors

    - User-friendly foutmeldingen
3. **Valideer input data**
    - Client-side validatie
    - Controleer verplichte velden
    - Valideer data types
    - Sanitize user input
4. **Monitor performance**
    - Track response tijden
    - Log API call volume
    - Monitor error rates
    - Set up alerts
5. **Documenteer aanpassingen**
    - Code comments
    - API documentatie
    - Change logs
    - Version control

Performance Optimalisatie

1. **Implementeer caching**
    - Cache masterdata lokaal
    - Use HTTP caching headers
    - Implementeer client-side cache
    - Set appropriate TTL
2. **Beperk aantal aanroepen**
    - Batch requests waar mogelijk
    - Gebruik pagination
    - Filter data server-side
    - Minimize payload size
3. **Optimaliseer dataverkeer**
    - Gebruik compressie (gzip)
    - Minimaliseer response data
    - Select only needed fields
    - Use efficient data formats
4. **Monitor response tijden**
    - Set performance baselines
    - Track trends over time
    - Identify bottlenecks
    - Optimize slow endpoints

Caching Strategieën

Master data caching:

- Cache duration: 24 uur
- Update trigger: versie controle
- Invalidatie: op nieuwe versie

Product data caching:

- Cache duration: 1-4 uur
- Update trigger: timestamp controle
- Invalidatie: op change detection

Token management:

- Refresh voor expiratie
- Secure storage
- Auto-renewal
- Error recovery

Cache invalidatie:

- Time-based (TTL)
- Event-based (updates)
- Manual (force refresh)
- Version-based (API updates)

---

13. Ondersteuning

Contact Informatie

Email technische vragen: ict@psinfoodservice.com

Email commerciële vragen: info@psinfoodservice.com

Telefoon: +31 085 044 18 96

Support tijden:

- Maandag t/m vrijdag: 09:00 - 17:00 CET
- Spoed support: via e-mail

FAQ

Veel gestelde vragen en antwoorden zijn beschikbaar op de support website.

Veelgestelde onderwerpen:

- Authenticatie problemen
- Rate limiting
- Data formaat vragen
- Implementatie tips
- Error handling

Aanvullende Bronnen

Swagger documentatie:
Swagger UI

Beschikbare resources:

- Postman collectie voor testing
- Code voorbeelden (C#, PHP, Python)
- Implementatie guides

Appendix

Glossary

Term	Definitie
JWT	JSON Web Token - authenticatie token formaat
GTIN	Global Trade Item Number - product identificatie
GLN	Global Location Number - locatie identificatie
EAN	European Article Number - barcode standaard
TTL	Time To Live - cache geldigheidsduur
DTO	Data Transfer Object - data transport model

Laatste update: Jan, 2026

Documentatie versie: 1.1.0

API versie: 7.0.0.1