AMPER API

AMPER API to otwarty interfejs **RestAPI** służący do wymiany danych z innymi systemami, takimi jak systemy ERP, porównywarki, sklepy internetowe.

External services integration

This page contains infromations and instructions on how to get data from AMPER for use in external services (like e-commerce site).

Example external e-commerce shop integration

Example project - generating CENEO.PL XML file

Prerequisites

1. AMPER API KEY
2. AMPER web serivce URL

To get those, please, contact with IT support of company you want to integrate with.

Example request

C#

```
var client = new
RestClient("https://{amper_ws_url}/external/v1/products/?api_key={amper_api_key}");
client.Timeout = -1;
var request = new RestRequest(Method.GET);
IRestResponse response = client.Execute(request);
Console.WriteLine(response.Content);
```

NodeJS

```
var request = require('request');
var options = {
  'method': 'GET',
  'url': 'http://{amper_ws_url}/external/v1/products/?api_key={amper_api_key}',
};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

PHP CURL

```
$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => 'http://{amper_ws_url}/external/v1/products/?api_key={amper_api_key}',
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => '',
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => 'GET',
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

Python

```
import requests

url = "http://{amper_ws_url}/external/v1/products/?api_key={amper_api_key}"

payload={}
headers = {}

response = requests.request("GET", url, headers=headers, data=payload)

print(response.text)
```

> replace `"amper_ws_url"` and `"amper_api_key"` with your data

Response

As a response you will get JSON. Example response with list of products

```
[
    {
        "id": 1037,
        "name": "BAHAMA zapach samochodowy NASZYJNIK Wild Hibiskus /1szt",
        "short_code": "850267",
        "description": "Zapachy inspirowane wakacjami oraz trendami konsumenckimi. Bahama & Co. to nie tylko akcesoria do samochodu, ale również atrakcyjne gadżety, stanowiące element dekoracyjny.",
        "ean": "978020137962",
        "sku": "850267",
        "vat": 23,
        "default_unit_of_measure": "szt.",
        "weight": "0.20",
        "default_price": "25.43"
    },
    {
        "id": 1034,
        "name": "VORTEX TKANINA wielofunkcyjna z mikrofibry /1szt",
        "short_code": "623765",
        "description": "Uniwersalne zastosowanie",
        "ean": "978020137962",
        "sku": "623765",
        "vat": 23,
        "default_unit_of_measure": "op.",
        "weight": "0.38",
        "default_price": "31.52"
    },
    {
        "id": 951,
        "name": "CALIFORNIA zapach samochodowy SCENT CONTROL Ice /1szt",
        "short_code": "851233",
        "description": "Papierowe zawieszki zapachowe: \nInnowacyjna technologia dostosowania intensywności zapachu – wystarczy odpowiednio ustawić zawieszkę. \nPapier pokryty zapachowymi olejkami.\nW zestawie sznurek umożliwiający wygodne zawieszanie zapachu. \nDostępne zapachy: Coronado cherry; Ice; New Car; Sea Breeze",
        "ean": "978020137962",
        "sku": "851233",
        "vat": 23,
        "default_unit_of_measure": "szt.",
```

```
        "weight": "0.33",
        "default_price": "18.38"
    },
]
```


Available methods

Here is a list of available methods. **HTTP** method that is allowed: **GET**

Products

With this method you can get list of products.

Methods

URL	HTTP allowed method	Description
/external/v1/products/	*GET*	get list of products
/external/v1/products/{id}/	*GET*	get product by ID

Categories

With this method you can get list of categories.

Methods

URL	HTTP allowed method	Description
/external/v1/categories/	*GET*	get list of categories
/external/v1/categories/{id}/	*GET*	get category by ID

Relation between Product and Categories

This method returns all bindings of product to categories

Methods

URL	HTTP allowed method	Description
/external/v1/product-categories/	*GET*	list of relations beteen product and category

Product files

This method returns all files attached to product

URL	HTTP allowed method	Description
/external/v1/product-files/	*GET*	list of files

Stocks

This method returns prodct's stocks

URL	HTTP allowed method	Description
/external/v1/stocks/	*GET*	list of stocks

Promotions

This method returns list of promotions

URL	HTTP allowed method	Description
/external/v1/promotions/	*GET*	list of promotions

Translator

Introduction

AmplifierApiClient .NET package provides ways to authenticate and communicate with Amplifier API. This includes access to B2B WS.

Nuget

You can add this package to your .NET project by using:

- **Package Manager Console in Visual Studio**
  `Install-Package AmperApiClient.NET`
- **.NET CLI**
  `dotnet add package AmperApiClient.NET`
- **add reference directly to your .csproj file**

```
    <ItemGroup>
          <PackageReference Include="AmperApiClient.NET" Version="1.x.x" />
      </ItemGroup>
```

Example project

Import

```
using Amplifier;
```

AmplifierJWTAuth

Constructors

Name	Description
Constructor(username, password, authUrl)	Creates an **AmplifierJWTAuth** with specified credentials

Methods

Return Type	Name	Description
Task<string>	**GetToken**	Fetches your JSON Web Token required for communication with backend of Amplifier

Constructor(username, password, authUrl)

```
using Amplifier;
```

```
string AUTH_URL = "your_ws_endpoint";
string USERNAME = "your_user_name";
string PASSWORD = "your_password";

AmplifierJWTAuth amplifierJWTAuth = new AmplifierJWTAuth(USERNAME, PASSWORD, AUTH_URL);
```

> replace `"your_ws_endpoint"`, `"your_user_name"` and `"your_password"` with your credentials

Name	Type	Description
your_ws_endpoint	*String*	Your dedicated AMPER WS endpoint
username	*String*	
password	*String*	

GetToken()

```
using Amplifier;
```

```
AmplifierJWTAuth amplifierJWTAuth = new AmplifierJWTAuth(USERNAME, PASSWORD, AUTH_URL);

string token = await amplifierJWTAuth.getToken();
```

Returns

Personal JSON Web Token

B2BWSConfig

Properties

Name	Type	Description
B2BWSUrl	*string*	target AMPER Web Service URL
ERPConnectionString	*string*	
JWTToken	*string*	JSON Web Token fetched by AmplifierJWTAuth

B2BWSBackend

Constructors

Name	Description
Constructor(b2BWSConfig)	Creates an B2BWSBackend using config containing JWT and WS URL

Methods

Return Type	Name	Description
Task<string>	**ChangeOrderStatus(status, token)**	Asynchronously changes the status of an order
Task<string>	**GetListOfComplaints**	Asynchronously fetches the list of complaints
Task<string>	**GetListOfOrders(status)**	Asynchronously fetches list of orders with a specified status
Task<string>	**GetOrder(token)**	Asynchronously fetches an order by it's token
Task	**SendAccountsAsync(accounts)**	Asynchronously sends the list of accounts
Task	**SendAddresses(addresses)**	Asynchronously Sends the list of addresses
Task	**SendCategoryDiscountAsync(category_discounts)**	Asynchronously sends the list of category discounts
Task	**SendCustomerProductLogisticMinimumAsync(relations)**	Asynchronously sends the list of logistic minimums for customers and products

Return Type	Name	Description
Task	**SendCustomerProductsRelationAsyncsAsync(relations)**	Asynchronously sends the list of customer product relations
Task*	**SendDocumentsAsync(documents)**	Asynchronously sends the list of documents
Task	**SendFile(path, fileName, product_external_id, order)**	Asynchronously sends a file
Task	**SendPriceLevelsAsync(priceLevels)**	Asynchronously sends the list of price levels
Task	**SendPricesAsync(priceLevels)**	Asynchronously sends the list of prices
Task	**SendProductCategoriesAsync(categories)**	Asynchronously sends the list of product categories
Task	**SendProductCategoriesRelationAsyncsAsync(relations)**	Asynchronously sends the list of product category relations
Task	**SendProductsAsync(products)**	Asynchronously sends the list of products
Task	**SendRelatedProductsAsync(related_products)**	Asynchronously sends the list of related products
Task	**SendSettlementsAsync(settlements)**	Asynchronously sends the list of settlements
Task	**SendStockLocationsAsync(stockLocations)**	Asynchronously sends the list of location of the product stock
Task	**SendStocksAsync(stocks)**	Asynchronously sends the list of product stock

Constructor(b2BWSConfig)

Parameters

Name	Type	Description
b2BWSConfig	*B2BWSConfig*	

ChangeOrderStatus(status, token)

Name	Type	Description
status	*String*	New status
token	*String*	Order's token

Returns

Task object containing result of the operation

GetListOfComplaints()

Returns

Task object containing list of complaints in JSON format as a string

GetListOfOrders(status)

Parameters

Name	Type	Description
status	*String*	Status of orders

Returns

Task object containing list of orders in JSON format as a string

GetOrder(token)

Parameters

Name	Type	Description
token	*String*	Order's token

Returns

Task object containing order in JSON format as a string

SendAccountsAsync(accounts)

Parameters

Name	Type	Description

accounts	*List<Account>*	List of accounts

Returns

Task object representing the result of the operation

SendAddresses(addresses)

Parameters

Name	Type	Description
addresses	*List<Address>*	List of addresses

Returns

Task object representing the result of the operation

SendCategoryDiscountAsync(category_discounts)

Parameters

Name	Type	Description
category_discounts	*List<CategoryDiscount>*	List of category discounts

Returns

Task object representing the result of the operation

SendCustomerProductLogisticMinimumAsync(relations)

Parameters

Name	Type	Description
relations	*List<CustomerProductLogisticMinimum>*	List of logistic minimums for customers and products

Returns

Task object representing the result of the operation

SendCustomerProductsRelationAsyncsAsync(relations)

Parameters

Name	Type	Description
relations	*List<CustomerProductRelation>*	List of customer product relations

Returns

Task object representing the result of the operation

SendDocumentsAsync(documents)

Parameters

Name	Type	Description
documents	*List<Document>*	List of documents

Returns

Task object representing the result of the operation

SendFile(path, fileName, product_external_id, order)

Parameters

Name	Type	Description
path	*String*	Path to a file
fileName	*String*	Name of a file after it's sent
product_external_id	*String*	Product's Id in client's database
order	*String*	

Returns

Task object representing the result of the operation

SendPriceLevelsAsync(priceLevels)

Parameters

Name	Type	Description
priceLevels	*List<PriceLevel>*	List of price levels

Returns

Task object representing the result of the operation

SendPricesAsync(priceLevels)

Parameters

Name	Type	Description
priceLevels	*List<Price>*	List of prices

Returns

Task object representing the result of the operation

SendProductCategoriesAsync(categories)

Parameters

Name	Type	Description
categories	*List<ProductCategory>*	List of product categories

Returns Task object representing the result of the operation

SendProductCategoriesRelationAsyncsAsync(relations)

Parameters

Name	Type	Description
relations	*List<ProductCategoryRelation>*	List of product category relations

Returns

Task object representing the result of the operation

SendProductsAsync(products)

Parameters

Name	Type	Description
products	*List<Product>*	List of products

Returns

Task object representing the result of asynchrous operation

SendRelatedProductsAsync(related_products)

Parameters

Name	Type	Description
related_products	*List<RelatedProducts>*	List of related products

Returns

Task object representing the result of the operation

SendSettlementsAsync(settlements)

Parameters

Name	Type	Description
settlements	*List<Settlement>*	List of settlements

Returns

Task object representing the result of the operation

SendStockLocationsAsync(stockLocations)

Parameters

Name	Type	Description
stockLocations	*List<StockLocation>*	List of location of the product stock

Returns

Task object representing the result of the operation

SendStocksAsync(stocks)

Parameters

Name	Type	Description
stocks	*List<Stock>*	List of product stock

Returns

Task object representing the result of the operation

Models

Account

Properties

Name	Type	Description
customers	*List<Customer>*	
external_id	*string*	Account's Id in client's database
name	*string*	
short_name	*string*	

Address

Properties

Name	Type	Description
city	*string*	
customer_external_id	*string*	Customer's Id in client's database
email	*string*	
external_id	*string*	Addres's Id in client's database
name	*string*	
phone	*string*	
postal_code	*string*	
street	*string*	
street_continuation	*string*	
voivodeship	*string*	

CategoryDiscount

Properties

Name	Type	Description
category_external_id	*string*	Category's Id in client's database
discount	*decimal*	
end_date	*string*	
external_id	*string*	Category discount's Id in client's database
order	*int*	
price_level_external_id	*string*	Price level's Id in client's database
start_date	*string*	

Complaint

Properties

Name	Type	Description

attachments	*List<object>*	
created_at	*DateTime*	
created_by	*DateTime*	
customer_external_id	*string*	Customer's Id in client's database
id	*int*	
lines	*List<ComplaintLine>*	
note	*string*	
notes	*List<object>*	
nr	*string*	
status	*string*	
updated_at	*int?*	
updated_by	*int?*	

ComplaintLine

Properties

Name	Type	Description
complaint	*id*	
description	*string*	
id	*id*	
name	*string*	
order	*string*	
product_external_id	*string*	Product's Id in client's database
product_id	*int*	
purchase_date	*string*	

Customer

Properties

Name	Type	Description
city	*string*	
comments	*string*	
discount	*decimal*	
doc_export	*bool*	
external_id	*string*	Customer's Id in client's database
ftp_export	*bool*	
login	*string*	
mail_export	*bool*	
name	*string*	
offer_export	*bool*	
overdue_limit	*decimal*	
password	*string*	
phone	*string*	
postal_code	*string*	
price_level_external_id	*string*	Price level's Id in client's database
primary_email	*string*	
short_name	*string*	
street	*string*	
tax_id	*string*	
template	*string*	
trade_credit_limit	*decimal*	

CustomerProductLogisticMinimum

Properties

Name	Type	Description
customer_external_id	*string*	Customer's Id in client's database
external_id	*string*	Id of logistic minimum for customer and product in client's database
logistic_minimum	*decimal*	

Name	Type	Description
product_external_id	*string*	Product's Id in client's database

CustomerProductRelation

Properties

Name	Type	Description
customer_external_id	*string*	Customer's Id in client's database
external_id	*string*	Customer product relation's Id in client's database
product_external_id	*string*	Product's Id in client's database

Document

Properties

Name	Type	Description
customer	*string*	
date	*string*	
description	*string*	
document_lines	*List<DocumentLine>*	
due_date	*string*	
external_id	*string*	Document's Id in client's database
number	*string*	
value_gross	*decimal*	
value_net	*decimal*	

DocumentLine

Properties

Name	Type	Description
document	*string*	
group	*string*	
make	*string*	
manufacturer	*string*	
price_gross	*decimal*	
price_net	*decimal*	
product	*decimal*	
product_ean	*decimal*	
product_name	*string*	
product_symbol	*string*	
quantity	*decimal*	
quantity_aggregate	*decimal*	
unit	*string*	
unit_aggregate	*string*	
value_gross	*decimal*	
value_net	*decimal*	
vat	*int*	

Order

Properties

Name	Type	Description
billing_address	*string*	
created	*DateTime*	
customer_external_id	*string*	Customer's Id in client's database
customer_note	*string*	
discount_amount	*object*	
lines	*List<OrderLine>*	
paid	*object*	

Name	Type	Description
products_total_gross	*string*	
products_total_net	*string*	
shipment	*int*	
shipment_type	*int*	
shipping_address	*int*	
shipping_price_gross	*string*	
shipping_price_net	*string*	
status	*string*	
token	*string*	
total_gross	*decimal*	
total_net	*decimal*	
updated	*DateTime*	
user_email	*string*	

OrderLine

Properties

Name	Type	Description
attributes	*List<object>*	
id	*int*	
product	*int*	
product_external_id	*string*	Product's Id in client's database
product_name	*string*	
product_sku	*string*	
quantity	*int*	
tax_rate	*string*	
unit_price_gross	*string*	
unit_price_net	*string*	

Price

Properties

Name	Type	Description
discount	*decimal*	
end_date	*string*	
external_id	*string*	Price's Id in client's database
order	*int*	
price	*decimal*	
price_level_external_id	*string*	Price level's Id in client's database
product_external_id	*string*	Product's Id in client's database
start_date	*string*	

PriceLevel

Properties

Name	Type	Description
external_id	*string*	Price level's Id in client's database
name	*string*	
order	*int*	

Product

Properties

Name	Type	Description
attributes	*List<ProductAttributes>*	
available_on	*string*	
can_be_split	*bool*	

Name	Type	Description
cumulative_converter	*decimal*	
cumulative_unit_of_measure	*string*	
cumulative_unit_ratio_splitter	*decimal*	
default_price	*decimal*	
default_unit_of_measure	*string*	
description	*string*	
external_id	*string*	Product's Id in client's database
friendly_name	*string*	
is_featured	*bool*	
is_published	*bool*	
name	*string*	
short_code	*string*	
short_description	*string*	
sku	*string*	
unit_roundup	*bool*	
vat	*int*	
weight	*decimal*	

ProductAttributes

Properties

Name	Type	Description
atr_name	*string*	
atr_val	*string*	
key	*string*	

Constructors

Name	Description
Constructor(String, String, String)	

Constructor(String, String, String)

ProductCategory

Properties

Name	Type	Description
description	*string*	
external_id	*string*	Product category's Id in client's database
name	*string*	
order	*int*	
parent_external_id	*string*	Category's parent category Id in client's database
seo_tags	*string*	

ProductCategoryRelation

Properties

Name	Type	Description
category_external_id	*string*	Category's Id in client's database
external_id	*string*	Product and category relation's Id in client's database
product_external_id	*string*	Product's Id in client's database

ProductImage

Name	Type	Description
alt	*string*	
product_id	*int*	
image	*string*	

Name	Type	Description
order	*int*	
thumbnail_width	*int*	

RelatedProduct

Properties

Name	Type	Description
external_id	*string*	Related product's Id in client's database

RelatedProducts

Properties

Name	Type	Description
external_id	*string*	Product's Id in client's database
related_products	*List<RelatedProduct>*	List of related products

Settlement

Properties

Name	Type	Description
customer	*string*	
date	*string*	
due_date	*string*	
external_id	*string*	Settlement's Id in client's database
number	*string*	
value	*decimal*	

Name	Type	Description
value_to_pay	*decimal*	

Stock

Properties

Name	Type	Description
external_id	*string*	
product_external_id	*string*	Product's Id in client's database
quantity	*decimal*	
quantity_allocated	*decimal*	
stock_level_external_id	*string*	Stock location's Id in client's database

StockLocation

Properties

Name	Type	Description
external_id	*string*	
name	*string*	

AMPER Channels

How to use amper-channels microservice (websockets)

Short instruction how to implement communication with microservice.

Frontend

Each of the front service (admin, b2b, msf, etc.) should firstly request a websocket authorization token through.

GET https://channels.ampli-solutions.com/authorize

```
Authorization Header: Bearer TOKEN
```

replace TOKEN with WS token

Code implementation - package [socket.io]

```
<script src="/socket.io/socket.io.js"></script>
```

For all the other methods of including socket.io package to your project, please refer to documentation: [client-api]

After receiving a token, use it to get your websocket connection

```
const url = 'https://channels.ampli-solutions.com/'
const token = token received from "authorize" API
const keycloakID = 'your keycloak'
const channel = 'msf' (channel used by your project) (b2b, admin, msf, etc)
const socket = io(url, {
auth: {
token:token,
keycloakID:keycloakID,
```

```
□□channel:channel
□}
})
```

Add a **notification** type event listener to your socket:

```
socket.on('notification', function(msg) {
□//here goes your action code that runs after notification is received
□(msg) - notification message object (string lub json)
})
```

Backend

You have 3 variants of notifications:

1. notifying every user of given channel (msg, b2b, admin, etc.),
2. notifiyng every channel that a given keycloak user is using,
3. notifying a single channel of given keycloak user

API Endpoint: POST https://channels.ampli-solutions.com/notify

BODY:

```
{
□"API_KEY": "5b28dc2f-...-7fbb705907c5", //current api key//Required
□"msg" : string/json, //Required
□"keycloak": "26259ede-....-9c58927efaae",//keycloak of the target user for
notification(variant 2 and 3)
□"channel": "msf"//(b2b, admin, etc)//when you want to use variant 1 or 2 notification
}
```

Summary: Variant 1 requires only the **channel** variable in the body, Variant 2: only **keycloak** Variant 3: both **channel** and **keycloak**

Wymiana ofert i dokumentów w systemie AMPER

Dokument opisuje dostępne możliwości konfigurowania ofert i formatów dokumentów na potrzeby wymiany danych z systemami obcymi.

U?ytkownik

Użytkownik ma możliwość eksportowania swoich dokumentów do dowolnego z dostępnych formatów.
Dostępne są 3 formy eksportu:

- w przeglądarce
- na serwer FTP
- na maila
- cyklicznie raz na dobę (dotyczy wysłki mailem i FTP)

Eksport w przegl?darce

W zakładce "moje dokumenty" użytkownik ma opcję pobrania dokumentu w wybranym formacie.

Opis techniczny:
Aby pobrać wybrany dokument w wybranym formacie należy użyć następującej metody z serwera ampli-ws:

```
GET /documents/:id_dokumentu/export/:id_szablonu
```

Jeżeli dokument oraz szablon istnieją, plik w wybranym formacie zostanie pobrany.

Eksport na serwer FTP

Użytkownik w zakładce Moje Konto -> Ustawienia ustawia dane dostępowe do serwera FTP takie jak:
adres serwera, port, nazwa użytkownika, hasło, katalog do którego mają być wrzucane pliki oraz czy transmisja ma być szyfrowana.
Dodatkowo konieczne jest wybranie do jakiego formatu mają być eksportowane dokumenty.
Od tego momentu co określony okres czasu wszystkie dokumenty niewyeksportowane poprawnie

będą
wrzucane na serwer FTP klienta.

Domyślnie dla każdego użytkownika jest zakładane konto FTP na serwerze: **ftp://ftp.amplifier.pl**.
Należy pamiętać, że dokumenty na tym serwerze FTP są automatycznie kasowane po 90 dniach.
Dane dostępowe do konta FTP są zapisywane wraz z utworzeniem konta.

Opis techniczny:
Eksportem i wysyłaniem plików zajmuje się serwer ampli-ws.
Serwer udostępnia zestaw standardowych operacji pod adresem: `/ftp-config`

Aby eksport był możliwy `Customer` musi mieć przypisany obiekt `FTPConfig`.

```
{
  "id": 1,
  "address": "localhost",
  "port": 2121,
  "username": "samplelogin",
  "password": "samplepass",
  "directory": "export",
  "is_secure": false,
  "customer": 3,
  "template": 2  // id szablonu do eksportu
}
```

Pole `directory` nie jest wymagane, pliki będą wtedy eksportowane do katalogu głównego.
W przypadku gdy transmisja ma być szyfrowana z wykorzystaniem TLS należy ustawić `is_secure` na `true`.
UWAGA! Jeżeli `is_secure` będzie ustawione niepoprawnie połączenie nie powiedzie się.

W pliku `documents/tasks.py` zdefiniowana jest funkcja `export_all_to_ftp()`.
Odpowiada ona za operację eksportu, a przebiega ona w sposób następujący:
Dla każdego użytkownika posiadającego konfigurację FTP wykonywany jest eksport dokumentów.
Eksportowane są tylko te dokumenty, dla których nie istnieje wpis w historii FTP, bądź wpis zawiera informację o błędzie.

Eksport na maila

Użytkownik w zakładce Moje Konto -> Ustawienia ustawia adres email na który mają
być wysyłane dokumenty oraz format do którego mają być eksportowane.

Od tego momentu co określony okres czasu wszystkie dokumenty niewyeksportowane poprawnie będą wysyłane na ten adres.

Opis techniczny:
Eksportem i wysyłaniem plików zajmuje się serwer ampli-ws.
Serwer udostępnia zestaw standardowych operacji pod adresem: `/mail-config`

Aby eksport był możliwy `Customer` musi mieć przypisany obiekt `MailConfig`.

```
{
  "id": 1,
  "address": "customer@example.com",
  "customer": 3,
  "template": 2  // id szablonu do eksportu
}
```

W pliku `documents/tasks.py` zdefiniowana jest funkcja `export_all_to_mail()`.
Odpowiada ona za operację eksportu, a przebiega ona w sposób następujący:
Dla każdego użytkownika posiadającego konfigurację email wykonywany jest eksport dokumentów.
Eksportowane są tylko te dokumenty, dla których nie istnieje wpis w historii Mail, bądź wpis zawiera informację o błędzie.
Każdy plik wysyłany jest w oddzielnej wiadomości jako załącznik.

Historia eksportów

Każde zdarzenie eksportu jest zapisywane w bazie danych.
Zapisywane jest id dokumentu, data i godzina, cel eksportu (przeglądarka, FTP, mail) oraz informacja o błędzie.
W przypadku eksportów automatycznych (FTP, mail) dokumenty, dla których poprzedni eksport się nie powiódł
zostaną wyeksportowane ponownie.

Administrator

Administrator systemu (nie sklepu) ma możliwość podglądu, definiowania, edycji oraz usuwania szablonów.

Opis techniczny:
Serwer ampli-ws udostępnia zestaw standardowych operacji pod adresem: `/document-templates`

Aby jakikolwiek dokument mógł zostać wyeksportowany wymagane jest dodanie przynajmniej jednego szablonu.
Przykładowy obiekt `DocumentTemplate` wygląda następująco:

```
{
  "id": 2,
  "name": "Testowy Format",
  "extension": ".txt",  // konieczne jest umieszczenie kropki razem z roszerzeniem
  "content": "Numer dokumentu: {{ document.number }}, data: {{ document.date }}"
}
```

Pole `content` zawiera w sobie standardowy szablon Django.
Mogą być używane wszystkie standardowe dla tego typu szablonów elementy.
Jako kontekst przekazywany jest obiekt typu `Document` pod nazwą `document`.

Dost?pne pola dla nag?ówka dokumentu

Dostęp: **document.nazwa_pola**

number - Numer dokumentu
date - Data dokumentu
due_date - Termin dokumentu
description - Opis
value_net - Wartość netto
value_gross - Wartość brutto

Tagi własne

- **get_vat_value_net**
- **get_vat_value_gross**
- **get_vat_value_vat**

Sposób użycia:

```
{% get_vat_value_net document 23 %}
```

zwraca wartość netto dla stawki VAT 23%

Dost?pne pola dla pozycji dokumentu

Dostęp: **line.nazwa_pola**

product_name - Nazwa
product_symbol - Symbol
product_vat - Stawka VAT produktu
product_ean - Kod EAN produktu
unit - Jednostka miary
quantity - Ilość
unit_aggregate - Jednostka miary zbiorcza"
quantity_aggregate - Ilość zbiorcza
price_net - Cena netto
price_gross - Cena brutto
value_net - Wartość netto
value_gross - Wartość brutto
manufacturer - Producent
make - Marka
group - Grupa
value_vat - Wartość VAT pozycji

Dost?pne pola dla kontrahenta

Dostęp: **customer.nazwa_pola**

name - Nazwa kontrahenta
short_name - Kod/nazwa skrócona
tax_id - Numer NIP

Dost?pne pola dla produktu

Dostęp: **line.product.nazwa_pola**

name - Nazwa produktu
friendly_name - Nazwa skrócona/nazwa zrozumiała dla użytkownika
short_description - Krótki opis produktu
description - Opis
short_code - Kod produktu
sku - SKU
vat - VAT
available_on - Dostępny od
is_published - Czy opulikowany
is_featured - Czy promowany
default_unit_of_measure - Domyślna jednostka miary
cumulative_unit_of_measure - Zbiorcza jednostka miary
cumulative_converter - Przelicznika na jednostkę zbiorczą
can_be_split - Czy jednostka miary zbiorcza podzielna
cumulative_unit_ratio_splitter - Współczynnik podzielności (uzupełniany tylko kiedy

can_be_split==True)
unit_roundup - Zaokrąglenie jednostki miary
weight - Waga
default_price - Cena 100
default_image.image.url - url do domyślnego obrazka produktu
product_b2b_url - url do produktu w systemie AMPER B2B
available_on_stock - czy produkt dostępny na stanie, zwraca True/False
stocks_available - podaje rzyczywisty stan produktu na magazynie
stocks_available_in_words - podaje stan magazynu w formie słownej `none/low/medium/high`
stock_availability_customer - podaje stan produktu na magazynie do którego jest przypisany kontrahent z uwzględnieniem flagi `Zwracaj rzeczywistą wartość stanów w API i ofertach`
cn_code - kod CN
category_path - ścieżka kategorii głównej do jakiej jest przypisany produkt
main_category - nazwa kategorii głównej

Tagi własne

- **attribute_value**
- **unit_of_measure_converter**

Sposób użycia:

```
{% attribute_value product [id atrybutu] %}
{% unit_of_measure_converter product [nazwa jednostki miary] %}
```

W szablonie należy zadeklarować tagi produktu: `{% load product_extras %}`

Kolumny dostępne tylko do typu szablonu **Oferty**

offer_best_promotion.price - najlepsza cena promocyjna
offer_best_promotion.promotion_name - nazwa promocji z jakiej pochodzi cena promocyjna
offer_best_promotion.promotion_id - id promocji
offer_best_promotion.promotion_url - link URL do promocji w B2B
offer_best_promotion.promotion_min_order_quantity - minimalna ilość zamówienia w promocji
offer_promotion_prices_list - lista cen promocyjnych z programi zwracana w formie `2880.00;1.39;1440.00;1.43;720.00;1.50;` . Czyli produkt występuje w trzech progach promocyjnych: 1.39 PLN za zakup 2880, 1.43 PLN za zakup 1440, 1.50 PLN za zakup 720
top_offer_starting_from_a_single_piece - najlepsza cena promocyjna dla minimum jednej sztuki

Operacje na liczbach

W definicji szablonu na samym początku należy dodać deklarcję:
`{% load mathfilters %}`

W szablonie można wtedy używać następujących operacji matematycznych:

- **sub** – odejmowanie
- **mul** – mnożenie
- **div** – dzielenie
- **intdiv** – dzielenie całkowite (podłoga)
- **abs** – wartość bezwzględna
- **mod** – modulo
- **addition** – zamiennik filtra add ze wsparciem dla typów float/decimal

Przykład:

```
{% load mathfilters %}
{% with answer=42 %}
42 * 0.5 = {{ answer|mul:0.5 }}
{% endwith %}
{% with numerator=12 denominator=3 %}
12 / 3 = {{ numerator|div:denominator }}
{% endwith %}
```

Dokumentacja sk?adni u?ywanej przy tworzeniu szablonów

https://docs.djangoproject.com/en/3.2/topics/templates/

Przyk?ad

```
{% for line in document.document_lines.all
%}{{line.product_name}};{{line.product_symbol}};{{line.quantity}};{{line.price_net}};{{line.va
lue_net}}
{% endfor %}
```

Promocje - integracja z systemami zewn?trznymi

? Dokumentacja techniczna struktura i logika promocji

? 1. Struktura pojedynczej promocji

Każdy element tablicy to *jedna promocja* w systemie AMPER B2B.


```
{
  "id": 22960679,
  "name": "promocja na palete",
  "description": "",
  "start": "2025-10-23T00:00:00+02:00",
  "end": "2028-10-20T23:59:00+02:00",
  "short_code": "promocja-na-palete",
  "default_image": { "image": "...", "alt": "..." },
  "keep_bundle_multiplayer_same": false,
  "use_promotion_price_for_threshold": false,
  "calculate_using_cumulative_units": true,
  "multiply_reward": 0,
  "color_in_cart": null,
  "start_using_default_price": false,
  "is_active": true,
  "conditions": [ ... ]
}
```


? Kluczowe pola

Pole	Typ	Opis
id	int	ID promocji
name	string	Nazwa promocji
description	string	Opis (opcjonalny)
start, end	datetime	Zakres obowiązywania promocji
short_code	string	Jednoznaczny kod identyfikacyjny
default_image	object	Domyślna grafika promocji

Pole	Typ	Opis
keep_bundle_multiplayer_same	bool	Czy utrzymać wielokrotność w zestawach
use_promotion_price_for_threshold	bool	Czy threshold/próg liczony wg ceny promocyjnej
calculate_using_cumulative_units	bool	Czy przeliczać jednostki po konwerterach/jednostkach zbiorczych
multiply_reward	int	Czy nagroda mnożona przez ilość pakietów
color_in_cart	string/null	Kolor w koszyku
start_using_default_price	bool	Czy liczyć od ceny domyślnej
is_active	bool	Czy aktywna
conditions	array	Lista warunków aktywujących promocję

? 2. Warunki promocji (conditions)

Każda promocja może zawierać **jeden lub wiele warunków**.

Struktura warunku:

```
{
  "threshold_min": 1.0,
  "threshold_max": 0.0,
  "measure": "quantity",
  "rewards": [ ... ]
}
```

? Pola warunku

Pole	Typ	Opis
threshold_min	float	Minimalny próg osiągnięcia promocji
threshold_max	float	Maksymalny próg (0 = brak limitu)
measure	string	Typ pomiaru: "quantity" lub "value"
rewards	array/null	Lista nagród (ceny, gratisy, rabaty)

? 3. Nagrody (rewards)

Każda nagroda jest powiązana do konkretnego produktu.

```
{
  "given_price": 8.0,
  "reward_value": 0.0,
  "value_type": "3",
  "quantity": 0.0,
  "product": {
    "id": 817,
    "ean": "817",
    "name": "BATERIA ENERGIZER...",
    "short_code": "426684-CC",
    "default_unit_of_measure": "bl.",
    "cumulative_unit_of_measure": "karton",
    "cumulative_converter": "24.00",
    "can_be_split": true,
    "cumulative_unit_ratio_splitter": "24.00",
    "unit_roundup": false
  }
}
```


? Pola nagrody

Pole	Typ	Opis
given_price	float	Cena promocyjna produktu
reward_value	float	Wartość rabatu (jeśli dotyczy)
value_type	string	Typ rabatu (np. stała cena, gratis, procent)
quantity	float	Ilość nagrody (np. liczba gratisów)
product	object	Opis produktu objętego promocją

? 4. Struktura produktu w nagrodzie (product)

Pole	Typ	Opis
id	int	ID produktu
ean	string	Kod EAN
name	string	Pełna nazwa
short_code	string	Kod skrócony
default_unit_of_measure	string	Jednostka bazowa
cumulative_unit_of_measure	string	Jednostka zbiorcza

Pole	Typ	Opis
cumulative_converter	string	Przelicznik jednostek
can_be_split	bool	Czy można dzielić jednostki
cumulative_unit_ratio_splitter	string	Współczynnik dzielenia
unit_roundup	bool	Czy zaokrąglać w górę

? 5. Logika liczenia promocji

5.1. Typ progu – measure

Dwa tryby:

1) quantity

Liczymy ilość sztuk / opakowań / jednostek.

2) value

Liczymy wartość koszyka dla produktów spełniających warunki.

5.2. Przeliczanie jednostek (calculate_using_cumulative_units)

Jeśli **true** – ilość do thresholdu liczona jest tak:

```
ilość_w_sztukach * cumulative_converter
```

Przykład:

- produkt: 6-pack → cumulative_converter = 6
- klient kupuje 2 opakowania
- do thresholdu liczymy: **12 szt.**

5.3. Multiplikacja nagród (multiply_reward)

Jeśli multiply_reward > 0:

- osiągnięcie kolejnych progów zwiększa liczbę nagród

Przykład:

threshold = 10 szt

multiply_reward = 1

klient kupił 30 szt → **3x nagroda**

? 6. Typowe przypadki

? Sta?a cena za produkt

Promocje typu:

> reward → value_type = 3, given_price > 0

Przykład: “Stała cena 5 zł przy zakupie min 10 szt”

? Gratis

Promocja ma quantity > 0 oraz given_price = 0.

? Progi wielokrotno?ci

Promocje:

- Próg wielokrotności - produkt kwota
- Próg wielokrotności - kategoria kwota

Mechanika:

- measure = value
- wiele warunków z tym samym threshold
- wiele nagród równoległych

? 7. Przyk?ad – schematy strukturalne

Promocja (Promo)

```
Promo
 ├── conditions [Condition]
 │     ├── threshold_min
 │     ├── threshold_max
 │     ├── measure (quantity/value)
 │     └── rewards [Reward]
 │           ├── given_price
 │           ├── reward_value
 │           ├── value_type
 │           ├── quantity
 │           └── product (Product)
 └── metadata...
```

? 8. U?ycie

Algorytm wyliczania promocji:

1. Pobierz wszystkie aktywne promocje dla klienta
2. Dla każdej promocji:
   - sprawdź daty start/end
   - oblicz threshold dla danego measure
   - uwzględnij cumulative conversions
3. Jeśli warunki spełnione:
   - zastosuj nagrody:
     - nadpisz cenę (given_price)
     - dodaj gratisy
     - nadaj rabat procentowy/kwotowy
4. Zapisz informację o zastosowanej promocji w koszyku.

? 9. U?ycie po stronie frontend (koszyk)

Front powinien:

- wyróżnić produkty w promocji (color_in_cart)
- pokazać cenę promocyjną given_price
- pokazać opis promocji
- dodać gratisy jako osobne linie (jeśli quantity > 0)
- opcjonalnie pokazać osiągnięty próg

? 10. Podsumowanie

AMPER to elastyczny model promocji zawierający:

✔ wiele progów

✔ wiele typów nagród

✔ wielokrotność progów

✔ konwersje jednostek

✔ stałe ceny

✔ gratisy

✔ rabaty

✔ pakiety

Tworzenie zamówienia "create-order"

Przegl?d

Endpoint `POST /create-order` tworzy zamówienie na podstawie podanych danych klienta i linii produktów. Przed wywołaniem należy uzyskać token JWT przez endpoint autoryzacyjny.

---

1. Autoryzacja - uzyskanie tokena

1.1 Pobranie tokena (logowanie)

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

Body żądania:

```
{
  "username": "login_uzytkownika",
  "password": "haslo"
}
```

Przykład (curl):

```
curl -X POST https://<host>/auth/ \
  -H "Content-Type: application/json" \
  -d '{"username": "jan.kowalski", "password": "tajnehaslo"}'
```

Odpowiedź (200 OK):

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cC...",
  "expires_in": 300,
```

```
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5c...",
  "token_type": "Bearer",
  "not-before-policy": 0,
  "session_state": "abc123...",
  "scope": "amper"
}
```

Pole	Opis
`access_token`	Token JWT do autoryzacji żądań API
`expires_in`	Czas ważności tokena w sekundach (domyślnie 300 s = 5 min)
`refresh_token`	Token do odświeżenia sesji bez ponownego logowania
`refresh_expires_in`	Czas ważności refresh tokena w sekundach

1.2 Od?wie?enie tokena

Gdy `access_token` wygaśnie, użyj `refresh_token` zamiast ponownie logować użytkownika.

```
POST /auth/token-refresh/
Content-Type: application/json
```

Body żądania:

```
{
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5c..."
}
```

Przykład (curl):

```
curl -X POST https://<host>/auth/token-refresh/ \
  -H "Content-Type: application/json" \
  -d '{"refresh_token": "eyJhbGciOiJIUzI1NiIsInR5c..."}'
```

Odpowiedź: Taki sam format jak przy logowaniu — nowy `access_token` i `refresh_token`.

---

2. Tworzenie zamówienia

Endpoint

```
POST /create-order
Authorization: Bearer <access_token>
Content-Type: application/json
```

Wymagane uprawnienia

Token musi należeć do użytkownika z zakresem `B2B` lub `TRANSLATOR`.

2.1 Parametry ??dania

Body żądania (JSON):

Pole	Typ	Wymagane	Opis
`customer_id`	`integer`	Tak*	ID klienta w systemie
`customer_external_id`	`string`	Tak*	Zewnętrzny ID klienta (alternatywa dla `customer_id`)
`lines`	`array`	Tak	Lista linii zamówienia (patrz niżej)
`playground`	`boolean`	Nie	`true` — symulacja bez zapisu zamówienia (domyślnie `false`)
`description`	`string`	Nie	Opis/notatka do zamówienia
`shipment_type_id`	`integer`	Nie	ID typu wysyłki

> * Wymagane jest podanie `customer_id` **lub** `customer_external_id`.

Pola obiektu w tablicy `lines`:

Pole	Typ	Wymagane	Opis
`product_id`	`integer`	Tak*	ID produktu w systemie
`product_external_id`	`string`	Tak*	Zewnętrzny ID produktu (alternatywa dla `product_id`)

Pole	Typ	Wymagane	Opis
`quantity`	`decimal`	Tak	Ilość (zaokrąglana do kroku jednostki produktu)
`suggested_price`	`decimal`	Tak	Sugerowana cena jednostkowa
`promotion_id`	`integer`	Nie	ID konkretnej promocji do zastosowania
`order_line_promotion_id`	`integer`	Nie	ID promocji linii zamówienia

> * Wymagane jest podanie `product_id` **lub** `product_external_id`.

2.2 Przyk?ady ??da?

Przykład z wewnętrznymi ID:

```
curl -X POST https://<host>/create-order \
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cC..." \
  -H "Content-Type: application/json" \
  -d '{
    "customer_id": 42,
    "lines": [
      {
        "product_id": 101,
        "quantity": 10,
        "suggested_price": 99.99
      },
      {
        "product_id": 205,
        "quantity": 5,
        "suggested_price": 49.50,
        "promotion_id": 7
      }
    ],
    "description": "Zamówienie testowe"
  }'
```

Przykład z zewnętrznymi ID (integracja ERP):

```
curl -X POST https://<host>/create-order \
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cC..." \
  -H "Content-Type: application/json" \
  -d '{
    "customer_external_id": "ERP-KLIENT-0042",
    "lines": [
      {
        "product_external_id": "SKU-ABC-001",
        "quantity": 10,
        "suggested_price": 99.99
      }
    ]
  }'
```

Przykład trybu playground (symulacja bez zapisu):

```
curl -X POST https://<host>/create-order \
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cC..." \
  -H "Content-Type: application/json" \
  -d '{
    "customer_id": 42,
    "playground": true,
    "lines": [
      {
        "product_id": 101,
        "quantity": 10,
        "suggested_price": 99.99
      }
    ]
  }'
```

2.3 Odpowied?

Sukces — 201 Created:

```
{
  "order_id": 12345,
  "lines": [
    {
```

```
      "product_id": 101,
      "product_external_id": "SKU-ABC-001",
      "quantity": 10,
      "suggested_price": 99.99,
      "default_price": 100.00,
      "customer_price": 95.00,
      "given_price": 90.00,
      "promotion": {
        "price": 90.00,
        "promotion_min_order_quantity": 5,
        "promotion_condition_item_id": 123,
        "promotion_condition_id": 456,
        "relation_id": 789,
        "promotion_name": "Letnia wyprzedaż",
        "promotion_id": 999,
        "promotion_url": "https://<host>/promotions/999/"
      }
    }
  ]
}
```

Pole	Opis
`order_id`	ID utworzonego zamówienia; `0` gdy `playground=true` lub brak linii
`lines`	Linie zamówienia z przeliczonymi cenami i promocjami
`lines[].default_price`	Cena katalogowa
`lines[].customer_price`	Cena indywidualna klienta
`lines[].given_price`	Cena końcowa (po uwzględnieniu promocji; `0` gdy brak promocji)
`lines[].promotion`	Najlepsza dostępna promocja lub `null` gdy nie dotyczy

> **Uwaga:** W odpowiedzi zwracane są tylko linie produktów, które są dostępne dla danego klienta i mają prawidłowe dane cenowe. Linie z niedostępnymi produktami są pomijane.

---

3. Kody b??dów

Kod	Opis
`201 Created`	Zamówienie zostało pomyślnie utworzone
`401 Unauthorized`	Brak lub nieprawidłowy token `Authorization`
`403 Forbidden`	Token ważny, ale brak wymaganego zakresu (`B2B` lub `TRANSLATOR`)
`404 Not Found`	Nieprawidłowe dane logowania przy pobieraniu tokena

---

4. Pe?ny przep?yw integracji

```
1. POST /auth/              → otrzymujesz access_token + refresh_token
        |
        ▼
2. POST /create-order       → zamówienie zostaje utworzone
   Authorization: Bearer <access_token>
        |
        ▼
3. Gdy access_token wygaśnie (expires_in sekund):
   POST /auth/token-refresh/ → otrzymujesz nowy access_token
   { "refresh_token": "..." }
```

Rekomendowane podejście w kliencie:

- Przechowuj `access_token` i `refresh_token` w pamięci sesji.
- Przed każdym żądaniem sprawdzaj, czy token nie wygasł (`expires_in`).
- W przypadku odpowiedzi `401` automatycznie wywołaj endpoint odświeżenia tokena.
- Gdy `refresh_token` również wygaśnie, ponownie pobierz token przez `/auth/`.

---

5. Uwagi implementacyjne

- **Tryb playground** (`"playground": true`) pozwala przetestować logikę cenową i promocyjną bez tworzenia zamówienia w bazie danych. Zwrócone `order_id` będzie równe `0`.
- **Zewnętrzne ID** (`customer_external_id`, `product_external_id`) są tłumaczone na wewnętrzne ID systemu przez moduł translatora. Używaj ich przy integracji z zewnętrznymi systemami (ERP, WMS).
- **Ilość** (`quantity`) jest automatycznie zaokrąglana do kroku jednostki produktu (np. jeśli krok to 5, zamówienie 7 szt. zostanie zaokrąglone do 10).

- **Promocje** są wybierane automatycznie (najlepsza dostępna dla klienta), chyba że podasz `promotion_id` w linii zamówienia.