

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](#)
 - [Example external e-commerce shop integration](#)
- [Translator](#)
- [AMPER Channels](#)
- [Wymiana ofert i dokumentów w systemie AMPER](#)

External services integration

This page contains informations 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 service 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/	<i>GET</i>	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/	<i>GET</i>	list of files

Stocks

This method returns prodct's stocks

URL	HTTP allowed method	Description
/external/v1/stocks/	<i>GET</i>	list of files

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
<i>Task<string></i>	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	<i>String</i>	Your dedicated AMPER WS endpoint
username	<i>String</i>	
password	<i>String</i>	

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	<i>string</i>	target AMPER Web Service URL
ERPConnectionString	<i>string</i>	
JWTToken	<i>string</i>	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
<i>Task<string></i>	<u>ChangeOrderStatus(status, token)</u>	Asynchronously changes the status of an order
<i>Task<string></i>	<u>GetListOfComplaints</u>	Asynchronously fetches the list of complaints
<i>Task<string></i>	<u>GetListOfOrders(status)</u>	Asynchronously fetches list of orders with a specified status
<i>Task<string></i>	<u>GetOrder(token)</u>	Asynchronously fetches an order by it's token
<i>Task</i>	<u>SendAccountsAsync(accounts)</u>	Asynchronously sends the list of accounts
<i>Task</i>	<u>SendAddresses(addresses)</u>	Asynchronously Sends the list of addresses

Return Type	Name	Description
Task	<u>SendCategoryDiscountAsync(category discounts)</u>	Asynchronously sends the list of category discounts
Task	<u>SendCustomerProductLogisticMinimumAsync(relations)</u>	Asynchronously sends the list of logistic minimums for customers and products
Task	<u>SendCustomerProductsRelationsAsync(relations)</u>	Asynchronously sends the list of customer product relations
Task*	<u>SendDocumentsAsync(documents)</u>	Asynchronously sends the list of documents
Task	<u>SendFile(path, fileName, product_external_id, order)</u>	Asynchronously sends a file
Task	<u>SendPriceLevelsAsync(priceLevels)</u>	Asynchronously sends the list of price levels
Task	<u>SendPricesAsync(priceLevels)</u>	Asynchronously sends the list of prices
Task	<u>SendProductCategoriesAsync(categories)</u>	Asynchronously sends the list of product categories
Task	<u>SendProductCategoriesRelationsAsync(relations)</u>	Asynchronously sends the list of product category relations
Task	<u>SendProductsAsync(products)</u>	Asynchronously sends the list of products
Task	<u>SendRelatedProductsAsync(related_products)</u>	Asynchronously sends the list of related products
Task	<u>SendSettlementsAsync(settlements)</u>	Asynchronously sends the list of settlements
Task	<u>SendStockLocationsAsync(stockLocations)</u>	Asynchronously sends the list of location of the product stock
Task	<u>SendStocksAsync(stocks)</u>	Asynchronously sends the list of product stock

Constructor(b2BWSSConfig)

Parameters

Name	Type	Description
b2BWSSConfig	B2BWSSConfig	

ChangeOrderStatus(status, token)

Name	Type	Description
status	<i>String</i>	New status
token	<i>String</i>	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	<i>String</i>	Status of orders

Returns

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

GetOrder(token)

Parameters

Name	Type	Description
token	<i>String</i>	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	<i>String</i>	Path to a file
fileName	<i>String</i>	Name of a file after it's sent
product_external_id	<i>String</i>	Product's Id in client's database
order	<i>String</i>	

Returns

Task object representing the result of the operation

SendPriceLevelsAsync(priceLevels)

Parameters

Name	Type	Description
priceLevels	<i>List</i> < PriceLevel >	List of price levels

Returns

Task object representing the result of the operation

SendPricesAsync(priceLevels)

Parameters

Name	Type	Description
priceLevels	<i>List</i> < 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

SendProductCategoriesRelationAsynccsAsync(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 asynchronous 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	<i>string</i>	Category's Id in client's database
discount	<i>decimal</i>	
end_date	<i>string</i>	
external_id	<i>string</i>	Category discount's Id in client's database
order	<i>int</i>	
price_level_external_id	<i>string</i>	Price level's Id in client's database
start_date	<i>string</i>	

Complaint

Properties

Name	Type	Description
attachments	<i>List<object></i>	
created_at	<i>DateTime</i>	
created_by	<i>DateTime</i>	
customer_external_id	<i>string</i>	Customer's Id in client's database
id	<i>int</i>	
lines	<i>List<ComplaintLine></i>	
note	<i>string</i>	
notes	<i>List<object></i>	
nr	<i>string</i>	
status	<i>string</i>	
updated_at	<i>int?</i>	

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

CustomerProductRelation

Properties

Name	Type	Description
customer_external_id	<i>string</i>	Customer's Id in client's database
external_id	<i>string</i>	Customer product relation's Id in client's database
product_external_id	<i>string</i>	Product's Id in client's database

Document

Properties

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

DocumentLine

Properties

Name	Type	Description
document	<i>string</i>	
group	<i>string</i>	
make	<i>string</i>	
manufacturer	<i>string</i>	

Name	Type	Description
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	
products_total_gross	string	
products_total_net	string	
shipment	int	
shipment_type	int	

Name	Type	Description
shipping_address	<i>int</i>	
shipping_price_gross	<i>string</i>	
shipping_price_net	<i>string</i>	
status	<i>string</i>	
token	<i>string</i>	
total_gross	<i>decimal</i>	
total_net	<i>decimal</i>	
updated	<i>DateTime</i>	
user_email	<i>string</i>	

OrderLine

Properties

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

Price

Properties

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

PriceLevel

Properties

Name	Type	Description
external_id	<i>string</i>	Price level's Id in client's database
name	<i>string</i>	
order	<i>int</i>	

Product

Properties

Name	Type	Description
attributes	<i>List</i> < <i>ProductAttributes</i> >	
available_on	<i>string</i>	
can_be_split	<i>bool</i>	
cumulative_converter	<i>decimal</i>	
cumulative_unit_of_measure	<i>string</i>	

Name	Type	Description
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	<i>string</i>	
external_id	<i>string</i>	Product category's Id in client's database
name	<i>string</i>	
order	<i>int</i>	
parent_external_id	<i>string</i>	Category's parent category Id in client's database
seo_tags	<i>string</i>	

ProductCategoryRelation

Properties

Name	Type	Description
category_external_id	<i>string</i>	Category's Id in client's database
external_id	<i>string</i>	Product and category relation's Id in client's database
product_external_id	<i>string</i>	Product's Id in client's database

ProductImage

Name	Type	Description
------	------	-------------

alt	<i>string</i>	
product_id	<i>int</i>	
image	<i>string</i>	
order	<i>int</i>	
thumbnail_width	<i>int</i>	

RelatedProduct

Properties

Name	Type	Description
external_id	<i>string</i>	Related product's Id in client's database

RelatedProducts

Properties

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

Settlement

Properties

Name	Type	Description
customer	<i>string</i>	
date	<i>string</i>	

Name	Type	Description
due_date	<i>string</i>	
external_id	<i>string</i>	Settlement's Id in client's database
number	<i>string</i>	
value	<i>decimal</i>	
value_to_pay	<i>decimal</i>	

Stock

Properties

Name	Type	Description
external_id	<i>string</i>	
product_external_id	<i>string</i>	Product's Id in client's database
quantity	<i>decimal</i>	
quantity_allocated	<i>decimal</i>	
stock_level_external_id	<i>string</i>	Stock location's Id in client's database

StockLocation

Properties

Name	Type	Description
external_id	<i>string</i>	
name	<i>string</i>	

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. notifying 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.

Użytkownik

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 przeglądarce

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 rozszerzeniem
  "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`.

Dostępne pola dla nagłówka 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%

Dostępne 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

Dostępne pola dla kontrahenta

Dostęp: **customer.nazwa_pola**

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

Dostępne 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 opublikowany

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 rzeczywisty 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

Operacje na liczbach

W definicji szablonu na samym początku należy dodać deklarację:

```
{% 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 składni używanej przy tworzeniu szablonów

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

Przykład

```
{% for line in document.document_lines.all
```

```
%}{{line.product_name}};{{line.product_symbol}};{{line.quantity}};{{line.price_net}};{{line.value_net}}
```

```
{% endfor %}
```