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

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 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
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

Return Type	Name	Description
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
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?*	

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	*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*	

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	*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*	
cumulative_converter	*decimal*	
cumulative_unit_of_measure	*string*	

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	*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*	
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*	

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