Wholesale API Manual

Version 2.3

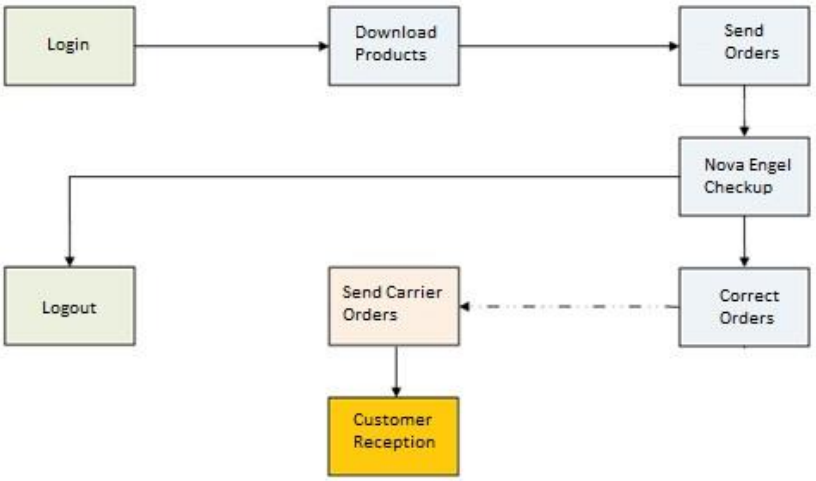
Content

Introduction

Nova Engel offers this new API System as a method to receive and send information to its clients. This is an API that uses web services (ideal for an important volume of data), through http protocol.

Process

The process can be summarized in the following scheme:



1. **Login**: Identifies the clients in the system. Each client has a username and a password that allows the access to the website and the API services.

2. **Download Products**: Allows to update the information about the products offered by Nova Engel to its clients. We recommend you to frequently download all the products data to be informed about the new items and the available stock at any time.

3. **Send Orders**: Allows to send to Nova Engel all the requested orders to be delivered. Each order must be identified by a PO reference. ! Warning! This reference can only contain numbers.

4. **Logout:** Logout of the system.

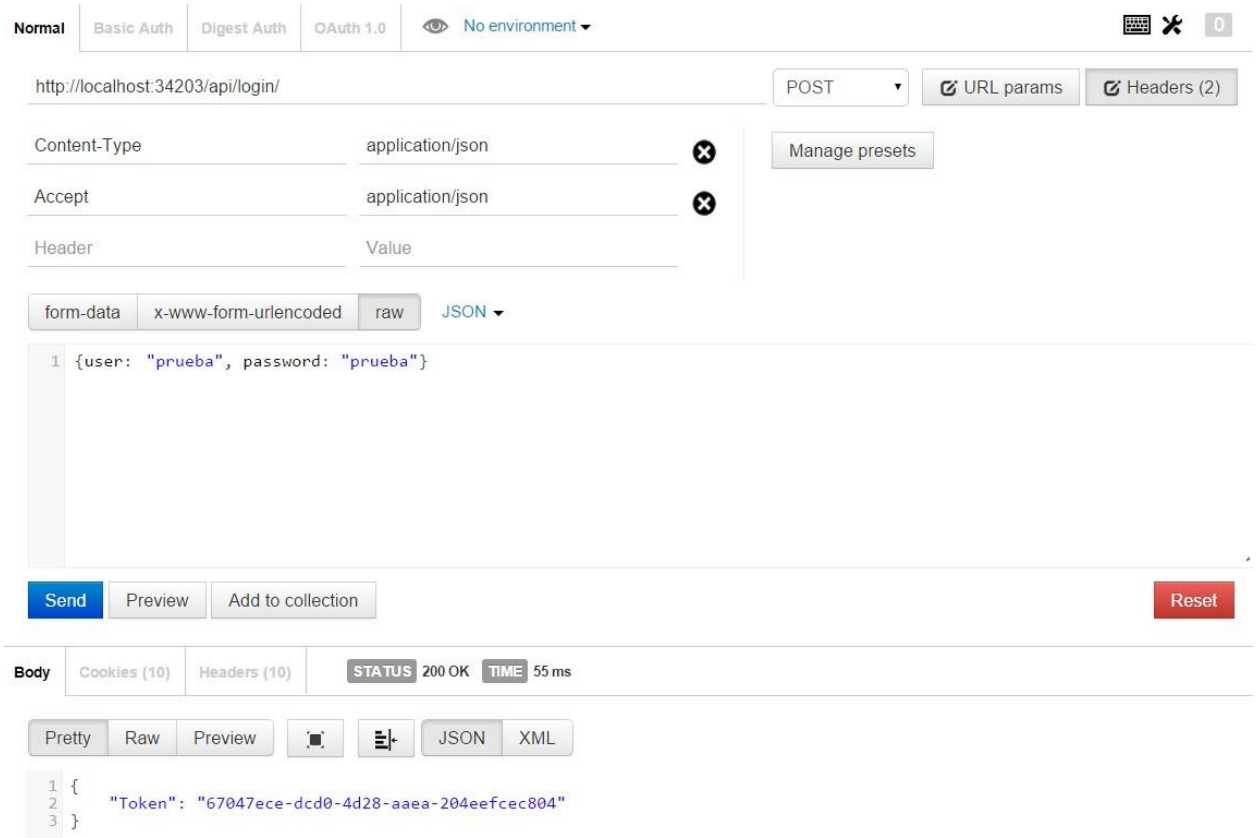
The API (REST)

The clients will work through standard **HTTP** operations (post and get methods). It is an easy way to work as it accepts JSON format and XML format. The available methods in the web service are explained below. The connexion is https://b2b.novaengel.com/ . The exact syntax of a call depends entirely on the platform from which they are called.

Login

This call identifies us in the system. It accepts a username and a password. If it all works properly, you will receive a string with a token (identifier) that will be used to call the next methods. If for some reason it fails, you will receive a null value.

Method: *Post*
Call: *https://b2b.novaengel.com/api/**login***
Parameters from body: *{user: "test", password: "test"}* **Returned Value:** *LoginModel*



Logout

This call overrides the token we are using and log out of the system. It is not essential, but it is advisable for security reasons.

Method: *Post*
Call: *https://b2b.novaengel.com/api/logout**/token***
Parameters from URI: *token*
Returned Value: *No value returned*

Products List

This call gives us all of the available products.
To be functional, it is necessary to enter the token previously provided by the Login. Then you can choose the language (es, en, fr…) in order to translate relevant information (product description, set content and categories).
In case of not specifying the language, the program will set one as default.
If for some reason it fails, you will receive an empty list of ProductModel.

Method: *Get*
Call: *https://b2b.novaengel.com/api/products/availables/**token/language***
Parameters from URI: *token, language*
Returned Value: *List<ProductModel>*


```
[
  {
    "Id": 101028,
    "EANs": [
      "8410412000659"
    ],
    "Description": "SOLAR ACEITE LUMINOSO bronceador SPF0 200 ml",
    "SetContent": null,
    "Price": 4.31,
    "PVR": 8,
    "Stock": 81,
    "BrandId": "1004",
    "BrandName": "Babaria",
    "Gender": "Unisex",
    "Families": [
      "Solar"
    ],
    "Kgs": 0.2,
    "Width": 67,
    "Height": 165,
    "Depth": 46,
    "Vat": 21,
    "Date": "2018-05-21T15:00:38.28",
    "Content": "200 ml",
    "ProductLine": "00000001058",
    "PriceQuantity": [
      {
        "Quantity": 1,
        "Price": 4.31
      },
      {
        "Quantity": 3,
        "Price": 4.07
      },
      {
        "Quantity": 12,
        "Price": 3.95
      }
    ],
    "ItemId": "1004-00659"
  },
```

Brand Products List

This call gives us all the available products of a brand according to our selection.
To be functional, it is necessary to enter the token previously provided by the Login. Then you can choose the language (es, en, fr…) in order to translate relevant information (product description, set content and categories).
In case of not specifying the language, the program will set one as default.
If for some reason it fails, you will receive an empty list of ProductModel.

Method: *Get*
Call: *https://b2b.novaengel.com/api/products/brand/**token/brandId/language***
Parameters from URI: *token, brandId, languague*
Returned value: *List<ProductModel>*


```
[
  {
    "Id": 87061,
    "EANs": [
      "8410190613430"
    ],
    "Description": "AGUA FRESCA edt vapo 230 ml",
    "SetContent": null,
    "Price": 29.76,
    "PVR": 57,
    "Stock": 71,
    "BrandId": "10",
    "BrandName": "Adolfo Dominguez",
    "Gender": "Hombre",
    "Families": [
      "Perfumes"
    ],
    "Kgs": 0.55,
    "Width": 107,
    "Height": 142,
    "Depth": 99,
    "Vat": 21,
    "Date": "2017-03-09T06:16:39.697",
    "Content": "230 ml",
    "ProductLine": "5200",
    "PriceQuantity": [
      {
        "Quantity": 1,
        "Price": 29.76
      },
      {
        "Quantity": 3,
        "Price": 29.01
      },
      {
        "Quantity": 12,
        "Price": 28.15
      }
    ],
    "ItemId": "10-13430"
  }
]
```

Paging Product List

This call gives us a page with the available products. To be functional, it is necessary to enter the token previously provided by the Login, the page number that we want to download (beginning by 0) and the number of products of each page. If the number of the returned products is lower than the requested orders, it means that we have downloaded all the data.
Then you can choose the language (es, en, fr…) in order to translate relevant information (product description, set content and categories). In case of not specifying the language, the program will set one as default.
If for some reason it fails, you will receive an empty list of ProductModel.

Method: *Get*
Call: *https://b2b.novaengel.com/api/products/paging/**token/pages/elements/language***
Parameters from URI: *token, pages, elements, languaje*
Returned value: *List<ProductModel>*


```
[
  {
    "Id": 101028,
    "EANs": [
      "8410412000659"
    ],
    "Description": "SOLAR ACEITE LUMINOSO bronceador SPF0 200 ml",
    "SetContent": null,
    "Price": 4.31,
    "PVR": 8,
    "Stock": 81,
    "BrandId": "1004",
    "BrandName": "Babaria",
    "Gender": "Unisex",
    "Families": [
      "Solar"
    ],
    "Kgs": 0.2,
    "Width": 67,
    "Height": 165,
    "Depth": 46,
    "Vat": 21,
    "Date": "2018-05-21T15:00:38.28",
    "Content": "200 ml",
    "ProductLine": "00000001058",
    "PriceQuantity": [
      {
        "Quantity": 1,
        "Price": 4.31
      },
      {
        "Quantity": 3,
        "Price": 4.07
      },
      {
        "Quantity": 12,
        "Price": 3.95
      }
    ],
    "ItemId": "1004-00659"
  },
```

Product Image

This call provides us an URL to access temporally the image of the requested product.
To be functional, it is necessary to enter the token previously provided by the Login and the ID of the requested product. If for some reason it fails, you will receive an empty string.

Method: *Get*
Call: *https://b2b.novaengel.com/api/products/image/**token/productId***
Parameters from URI: *token, productId*
Returned value: *string*

Stock

This call gives us all the available products. The information is minimal, therefor this is an ideal method to maintain the stock of the products updated.

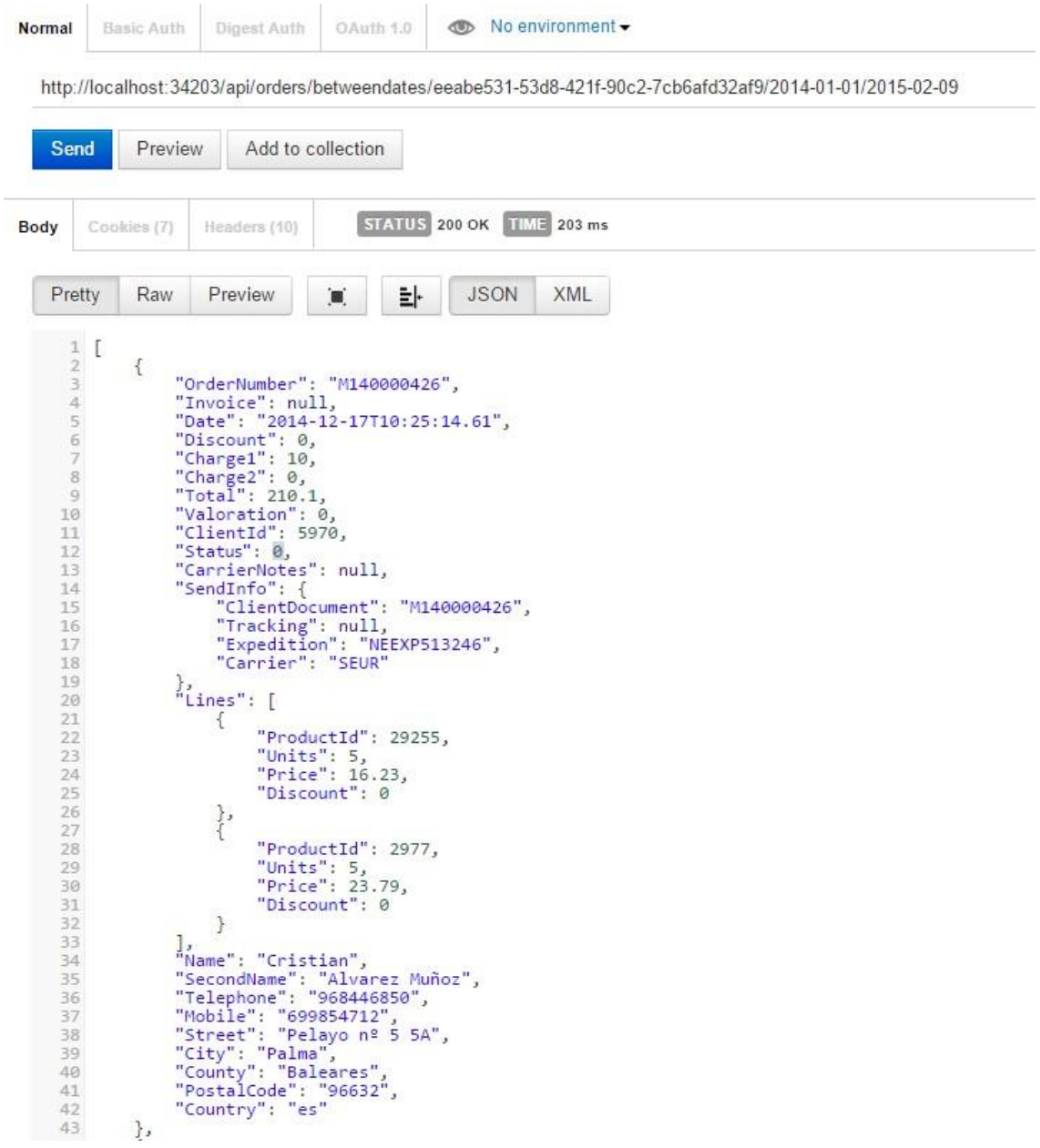
Method: *Get*
Call: *https://b2b.novaengel.com/api/stock/update/**token***
Parameters from URI: *token*
Returned value: *List<StockProductModel>*


```
[
    {
        "Id": 106118,
        "Stock": 0
    },
    {
        "Id": 106119,
        "Stock": 0
    },
    {
        "Id": 68540,
        "Stock": 5
    },
    {
        "Id": 68545,
        "Stock": 194
    },
    {
        "Id": 68562,
        "Stock": 3
    },
    {
        "Id": 68572,
        "Stock": 477
```

Orders Between Dates

This call lets us know the number of orders between the dates of our selection and their general status. We will need the token, the start date and the end date. If for some reason it fails, you will receive an empty list of OrderModel.

Method: *Get*
Call: *https://b2b.novaengel.com/api/orders/betweendates/**token/initiation/final***
Parameters from URI: *token, initiation (date), final (date).*
Returned value: *List<OrderModel>*




```
[
    {
        "OrderNumber": "M140000426",
        "Invoice": null,
        "Date": "2014-12-17T10:25:14.61",
        "Discount": 0,
        "Charge1": 10,
        "Charge2": 0,
        "Total": 210.1,
        "Valoration": 0,
        "ClientId": 5970,
        "Status": 0,
        "CarrierNotes": null,
        "SendInfo": {
            "ClientDocument": "M140000426",
            "Tracking": null,
            "Expedition": "NEEXP513246",
            "Carrier": "SEUR"
        },
        "Lines": [
            {
                "ProductId": 29255,
                "Units": 5,
                "Price": 16.23,
                "Discount": 0
            },
            {
                "ProductId": 2977,
                "Units": 5,
                "Price": 23.79,
                "Discount": 0
            }
        ],
        "Name": "Cristian",
        "SecondName": "Alvarez Muñoz",
        "Telephone": "968446850",
        "Mobile": "699854712",
        "Street": "Pelayo nº 5 5A",
        "City": "Palma",
        "County": "Baleares",
        "PostalCode": "96632",
        "Country": "es"
    },
```

Send Orders

It allows us to place the orders that we want to be delivered. The parameters accept the Token provided by the Login and a list with the orders to be placed. We will obtain information for each order in case of error. To send the orders correctly, it has to be in JSON format.

JSON format, from Body:


```
[
  {
    "OrderNumber": "568746", "Valoration": 0,
    "CarrierNotes": "closed monday"
    , "Lines": [
      {
        "ProductId": 2977, "Units": 5
      },
      {
        "ProductId": 15162, "Units": 5
      }
    ],
    "Name": "Cristian", "SecondName": "Alvarez
    Muñoz", "Telephone": "968446850", "Mobile":
    "699854712",
    "Street": "Pelayo nº 5 5A",
    "City": "Palma", "County": "Baleares",
    "PostalCode": "96632", "Country":
    "ES",
  }
]
```


The information provided will contain the possible mistakes found in the orders and the orders which have been sent correctly.

Method:	*Post*
Call:	*https://b2b.novaengel.com/api/orders/send/**token***
Parameters from URI:	*token*
Parameters from Body:	*orderList (**json format**)*
Returned value:	*List<OrdersInfoModel*

Normal | Basic Auth | Digest Auth | OAuth 1.0 | No environment

http://localhost:34203/api/orders/send/03a96a4c-ca49-405b-b14d-92acf650f5a9/ POST

form-data | x-www-form-urlencoded | raw | JSON

```
[
    {
        "OrderNumber": "568746",
        "Valoration": 0,
        "TransportNotes": "cerrado lunes",
        "Lines": [
            {
                "ProductId": 29255,
                "Units": 5
            },
            {
                "ProductId": 2977,
                "Units": 5
            }
        ],
        "Name": "Cristian",
        "SecondName": "Alvarez Muñoz",
        "Telephone": "968446850",
        "Mobile": "699854712",
        "Street": "Pelayo nº 5 5A",
        "City": "Palma",
        "County": "Baleares",
        "PostalCode": "96632",
        "Country": "ES",
    }
]
```

Send | Preview | Add to collection

Body | Cookies (10) | Headers (10) | STATUS 200 OK | TIME 247 ms

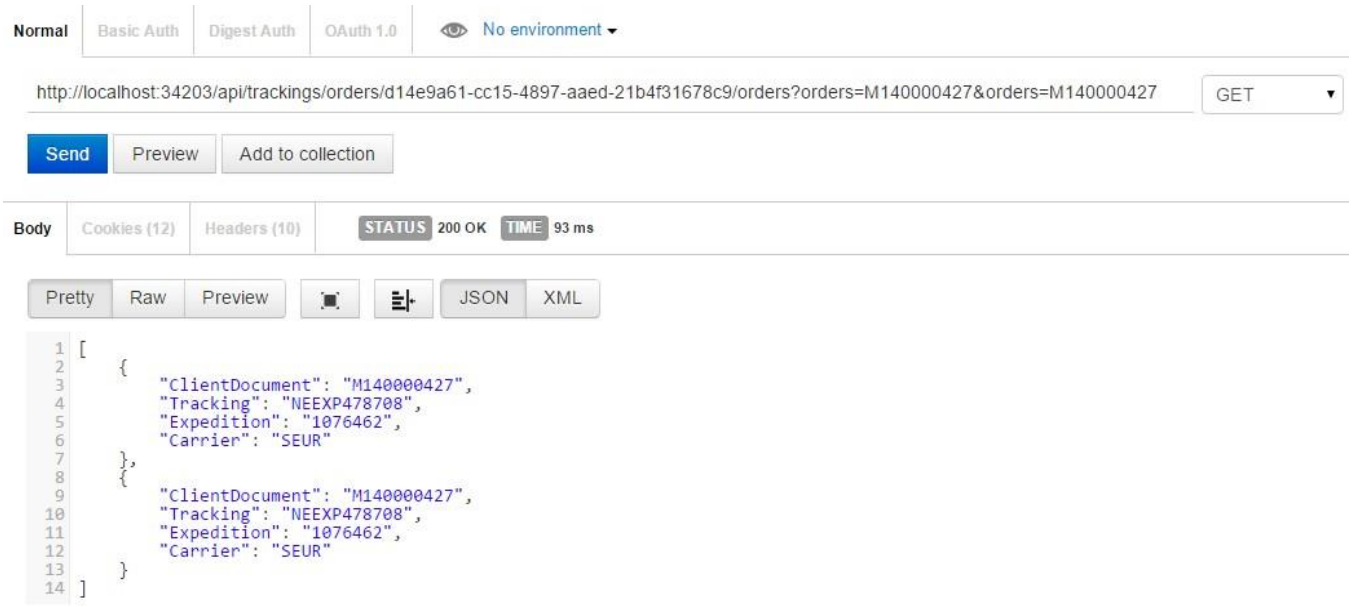
Pretty | Raw | Preview | JSON | XML

```
[
    {
        "Message": "Orders booked 1"
    }
]
```

Get Tracking

This method allows us to get the tracking information provided by the transporter. If for some reason it fails, we will receive an empty list of TrackingModel.

Method: *Get*
Call: *https://b2b.novaengel.com/api/trackings/orders/**token/orders***
Parameters from URI: *token, orders*
Parameters from Body: *orderList (**xml** **format**)*
Returned value: *List<TrackingModel>*



The Data

Below we will explain the data structure used in the API.

LoginModel

It assigns a token that we will need to make the next calls.

```
public class LoginModel
{
    public string Token { get; set; }
}
```

ProductModel

Contains the information about Nova Engel's products. The format is:

```
public class ProductModel
{
    public int Id { get; set; }
    public List<string> EANs { get; set; }
    public string Description { get; set; }
    public string SetContent { get; set; }
    public decimal Price { get; set; }
    public decimal PVR { get; set; }
    public int Stock { get; set; }
    public string BrandId { get; set; }
    public string BrandName { get; set; }
    public string Gender { get; set; }
    public List<string> Families { get; set; }
    public decimal Kgs { get; set; }
    public decimal Vat { get; set; }
    public int Height { get; set; }
    public int Width { get; set; }
    public int Depth { get; set; }
    public List<PrecioCantidad> PriceQuantity { get; set; }
    public string Date { get; set; }
    public string Content { get; set; }
    public string ProductLine { get; set; }
    public string ItemId { get; set; }
    public List<PropertyModel> Properties { get; set; }
```

- **Id:** Unique identifier of the product.
- **EANs:** EANs list of the product.
- **Description:** Product description.
- **SetContent:** Content of the Set.
- **Price:** Nova Engel's sale price.
- **PVR:** Recommended retail price of the product.
- **Stock:** Current stock of Nova Engel.
- **BrandId:** Brand Identifier of the product.
- **BrandName:** Name of the Brand.
- **Gender:** Gender of the product. It can be Unisex, Woman or Man.
- **Families:** List of the families the product belongs.

- **Vat:** Product VAT.
- **Kgs:** Product weigth in Kilograms.
- **Width:** Product width in millimeters.
- **Height:** Product heigth in millimeters.
- **Depth:** Product length in millimeters.
- **PriceQuantity:** Price list by quantity.
- **Date:** Date product data creation in our system.
- **Content:** Content of the product.
- **ProductLine:** Product range.
- **ItemId:** Other identifier of the product.
- **Properties:** Product's properties list.

PropertyModel

Contains products' properties of the sunglasses family (caliber, arms and bridge).

```
public class PropertyModel
{
    public string Name { get; set; }
    public string Value { get; set; }
}
```

- **Name:** Nombre de la propiedad.
- **Value:** Valor de la propiedad.

StockProductModel

It contains the updated stock of the products. The format is:

```
public class StockProductModel
{
    public int Id { get; set; }
    public int Stock { get; set; }
    public int Price { get; set; }
}
```

- **Id:** Unique identifier product.
- **Stock:** Current stock of Nova Engel.
- **Price:** Current price of the product.

ClientOrderModel

It contains the order to send to Nova Engel. The structure is:

```
public class ClientOrderModel
    {
        public string OrderNumber { get; set; }
        public decimal Valoration { get; set; }
        public string CarrierNotes { get; set; }
        public IList<ClientOrderLineModel> Lines { get; set; }
```

```
        public string Name { get; set; }
        public string SecondName { get; set; }
        public string Telephone { get; set; }
        public string Mobile { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public string County { get; set; }
        public string PostalCode { get; set; }
        public string Country { get; set; }
```

- **OrderNumber**: Order number assigned by the client. The max. Length is 15 characters. This reference can only contain numbers.
- **Valoration**: Optional parameter with the estimation.
- **CarrierNotes**: Notes for the transporter.
- **Lines**: List of products and quantity of the order.
- **Name**: Recipient's first name.
- **SecondName**: Recipient's last name.
- **Telephone**: Phone number, in case the transporter need further information (optional).
- **Mobile**: Cellphone number, in case the transporter need further information (optional).
- **Street**: Delivery address.
- **City**: City of delivery.
- **County:** County of delivery.
- **PostalCode:** Post code.
- **Country:** Country of delivery.

ClientOrderLineModel

Details of each order. It indicates the product and the quantity ID:

```
public class ClientOrderLineModel
{
    public int ProductId { get; set; }
    public int Units { get; set; }
}
```

- **ProductId**: Id of the product.
- **Units**: Number of products.

OrderModel

It contains the order information.

```
public class OrderModel
    {
        public string OrderNumber { get; set; }
        public string Invoice { get; set; }
        public DateTime Date { get; set; }
        public decimal Discount { get; set; }
        public decimal Charge1 { get; set; }
        public decimal Charge2 { get; set; }
        public decimal Total { get; set; }
        public decimal Valoration { get; set; }
```

```
        public int ClientId { get; set; }
        public EstadoPedido Status { get; set; }
        public string CarrierNotes { get; set; }
        public TrackingModel SendInfo { get; set; }
        public IList<OrderLineModel> Lines { get; set; }
        public string Name { get; set; }
        public string SecondName { get; set; }
        public string Telephone { get; set; }
        public string Mobile { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public string County { get; set; }
        public string PostalCode { get; set; }
        public string Country { get; set; }
    }
```

- **OrderNumber**: Order number assigned.
- **Invoice**: Order invoice.
- **Date**: Order date.
- **Discount**: Order discount.
- **Charge1**: First charge.
- **Charge2**: Second Charge.
- **Total**: Total amount of the order.
- **Valoration**: Optional parameter with the estimation.
- **ClientId**: Id of the client.
- **Status**: Status of the order.
- **CarrierNotes**: Notes for the carrier.
- **SendInfo**: Tracking information of the order.
- **Lines**: List of products and quantity to be delivered.
- **Name**: Recipient's first name.
- **SecondName**: Recipient's last name.
- **Telephone**: Phone number, in case the transporter need further information (optional).
- **Mobile**: Cellphone number, in case the transporter need further information (optional).
- **Street**: Shipment address.
- **City**: City of delivery.
- **County:** County of delivery.
- **PostalCode:** Post code.
- **Country:** Country of delivery.

Status can take the following values: `{ ShippingSlope= 1, Invoiced = 2, Sent = 3, StockError = 4, CarrierError = 5, Canceled = 6, Received = 7}`

OrderLineModel

Order lines information. It includes the discount and the associated price.

```
public class OrderLineModel
{

    public int ProductId { get;
    set; }
    public int Units { get; set; }
    public decimal Price { get;
    set; }
    public decimal Discount { get; set; }
}
```

- **ProductId**: product's identifier.
- **Units**: Assigned units.
- **Price**: Applied price.
- **Discount**: Applied discount.

OrdersInfoModel

Information about the errors related to the orders' importation. The structure is:

```
public class OrdersInfoModel
{
    public string Message { get; set; }
     public string OrderNumber { get; set; }
 }
```

- **Message**: Message of the error.
- **OrderNumber**: Order Number.

TrackingModel

Indicates the information related to the order tracking.

```
public class TrackingModel
    {
        public string ClientDocument { get; set; }
        public string Tracking { get; set;
        }
        public string Expedition { get;
        set; }
        public string Carrier { get; set; }
    }
```

- **ClientDocument**: Order number assigned by the client.
- **Tracking**: Tracking number assigned by the transporter.
- **Expedition**: Expedition number in our system.
- **Carrier**: Transporter in charge of the expedition.

All methods previously mentioned have been tested with Google Chrome that you can get by clicking here.

The application POSTMAN have been also used.

To perform tests, we remind you that we have two headers:

Content-Type	application/json
Accept	application/json