nutribl
private label supplements

Orders API

Introduction

We have built an API through which our trade customers can submit orders to us, and this document sets out how this API can be used.

Overview

This section gives a high-level overview of how to use the API to submit orders to Nutribl.

Any customer wishing to use this API must apply to us for a set of API credentials. We will give you a username and password, and these credentials will need to be used to call our Authenticate method.

The Authenticate method will return a JSON object including a token that must be used with any further requests. This token will expire seven days after it is created, so can be cached and reused in future requests without the need to call Authenticate.

The JSON object also contains an id value - this is your internal customer id. This must be cached for later use also.

Once a valid token has been received, you can call the SendOrder method. This method expects the body of the request to contain a collection of OrderWireFormat objects, one for each ordered item.

One of the data items required in the OrderWireFormat is CustomerId - this is your Nutribl customer id which is returned as part of the Authenticate response

In the event of a failure, you will receive a BadRequest response with a message describing the error.

Once we have received order lines via the API, these are then stored in our database. A scheduled task which runs periodically (currently every fifteen minutes but this is subject to change) will then import these orders into our Warehouse Management System. Part of this process may include taking payment from your recorded payment mechanism.

In the event that any errors occur at this stage, you will be notified by email.

In order to check whether an order has been shipped, and to get a tracking number (if one is available - not all shipping methods will provide one), you can use the CheckShippingStatus method. This will return an object containing a shipped on date (if this is null, then the order has not yet shipped), the Nutribl internal order number and a tracking number if a) the order has shipped and b) the shipping method chosen by you for the order facilitates one.

Test Implementation

We provide a test implementation of the API for you to use to build and test your code. When we create an API User account for you, we will provide credentials for both live and test environments.

We will also provide a personalised Postman collection for you to use in your development. This should simplify your tasks and help understand how the API functions.

nutribl
private label supplements

Authenticate Method

URL (live)	https://webapi2.mercante.co.uk/Users/Authenticate
URL (test)	http://webapi.mercante.co.uk/Users/Authenticate
Method	POST
Header	Content-Type = application/json

The request needs a parameter adding with content-type = application/json which is a JSON object representing your credentials:

```
{
    "Username":"yourusername",
    "Password":"yourpassword"
}
```

The response object is as follows:

```
{
    "id": 123456,
    "username": "yourusername",
    "password": null,
    "token": "jwttoken"
}
```

The following is c# code snippet for authenticating and caching the token, as well as logging errors (assuming you have a logging class implemented) using RestSharp and Newtonsoft.Json:

```
public class ApiUser
{
        public int Id { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string Token { get; set; }
}

var client = new
RestClient("https://webapi2.mercante.co.uk/Users/Authenticate");
client.Timeout = -1;
var request = new RestRequest(Method.POST);
request.AddHeader("Content-Type", "application/json");
ApiUser credentials = new ApiUser
{
        Password = “yourpassword”,
        Username = “yourusername”
};
string json = JsonConvert.SerializeObject(credentials);
Parameter p = new Parameter
```

```
{
      Name = "undefined",
      Value = json,
      Type = ParameterType.RequestBody
};
request.AddParameter(p);
var response = client.Execute(request);
if (response.IsSuccessful)
{
      ApiUser tokenUser =
JsonConvert.DeserializeObject<ApiUser>(response.Content);
      if (!string.IsNullOrEmpty(tokenUser.Token))
      {
            token = tokenUser.Token;
                    var cacheEntryOptions = new
      MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromHours(1));
      _memoryCache.Set(“yourcachekey”, token, cacheEntryOptions);
      }
}
else
{
      _logger.Error($"Api Authentication Error. Request returned
{response.Content}");
      _logger.Error($"Api Authentication Error. Request returned
{response.ErrorMessage}");
      _logger.Error($"Api Authentication Error. Request returned
{response.StatusDescription}");
      if (response.ErrorException != null)
      {
            _logger.Error($"Api Authentication Error. Request returned
{response.ErrorException.Message}");
      }
      throw new Exception("Call failed because the Api call was not
authenticated");
}
```

Token Caching

Under normal circumstances, the token is cached for seven days. However, if the API is restarted for any reason, then our copy of the token will be deleted, so it’s always worth ensuring that you expire your cached version of the token more frequently than every seven days, and you write defensive code that will fail gracefully and retry if you inadvertently send us an expired token.

SendOrder

URL (live)	https://webapi2.mercante.co.uk/Order/SendOrder
URL (test)	http://webapi.mercante.co.uk/Order/SendOrder
Method	POST
Header	Content-Type = application/json

nutribl
private label supplements

This method requires that the JWT bearer token returned by Authenticate is added as an Authorization header. In C# using RestSharp, this looks like this:

```
request.AddHeader("Authorization", $"bearer {yourtoken}");
```

The body of the request needs to contain an array of OrderWireFormat objects. A C# class to represent this look like this:

```
public class OrderWireFormat
{
      public string Sku { get; set; }
      public string Size { get; set; }
      public string Color { get; set; }
      public int Quantity { get; set; }
      public string FirstName { get; set; }
      public string LastName { get; set; }
      public string Company { get; set; }
      public string Address1 { get; set; }
      public string Address2 { get; set; }
      public string City { get; set; }
      public string State { get; set; }
      public string PostCode { get; set; }
      public string Country { get; set; }
      public string ShippingMethod { get; set; }
      public Guid BatchGuid { get; set; }
      public int CustomerID { get; set; }
}
```

The meaning of these properties is as follows:

SKU
The Nutribl SKU of the item you wish to order

Size
If you sell a product that needs this property, you will be informed of how you should populate it. Leave empty otherwise

Color
If you sell a product that needs this property, you will be informed of how you should populate it. Leave empty otherwise

Quantity
The quantity you would like us to dispatch

FirstName, LastName etc.
This represents the address you want us to deliver to

ShippingMethod
This is the Nutribl shipping method you wish us to use to send the item. The method name must exactly match one of the following methods:

- 1st Class TR
- 2nd Class TR
- UK Courier TR
- 1st Class Recorded TR

- 2nd Class Recorded TR
- Europe Airmail TR
- Europe Airmail Tracked TR
- ROW Airmail TR
- ROW Signed Airmail TR

BatchGuid

BatchGuid must be a newly generated Globally Unique Identifier - it must be the same for all order lines submitted as a single order so if you are sending three orders, one with two lines, one with three lines and one with one line, each group of lines will need a distinct BatchGuid, but you can still submit all six lines in a single call. If you use the same BatchGuid for ALL lines, then they will be treated as a single order.

If you wish to use the CheckShippingStatusMethod you will need to store this against your order as BatchGuid is a parameter for the CheckShippingStatusMethod.

CustomerId

This is the id returned by the Authenticate method

A complete C# code snippet to use the SendOrder method is as follows:

```
var client = new RestClient("https://webapi2.mercante.co.uk/Order/SendOrder");
var request = new RestRequest(Method.POST);
request.AddHeader("Authorization", $"bearer {yourtoken}");
request.AddHeader("Content-Type", "application/json");

List<OrderWireFormat> orderItems = new List<OrderWireFormat>();
var batchGuid = Guid.NewGuid();

foreach (var item in itemsToFulfil)
{
      orderItems.Add(new OrderWireFormat
      {
            Sku = sku,
            Size = size,
            Color = colour,
            Quantity = item.Quantity,
            FirstName = order.ShippingAddress.FirstName,
            LastName = order.ShippingAddress.LastName,
            Company = order.ShippingAddress.Company,
            Address1 = order.ShippingAddress.Address1,
            Address2 = order.ShippingAddress.Address2,
            City = order.ShippingAddress.City,
            PostCode = order.ShippingAddress.ZipPostalCode,
            State = order.ShippingAddress.StateProvince.Name,
            Country = order.ShippingAddress.Country.Name,
            BatchGuid = batchGuid,
            ShippingMethod = "your chosen shipping method",
            CustomerID = Id // from authenticate methid
      });
}
string json = JsonConvert.SerializeObject(orderItems);
Parameter p = new Parameter
{
    Name = "undefined",
    Value = json,
    Type = ParameterType.RequestBody
};
request.AddParameter(p);
var response = client.Execute(request);
if (!response.IsSuccessful)
{
    _logger.Error($"Failed to send order {order.Id} to THC. Message received was
{response.Content}");
}
```

CheckShippingStatus

URL (live)	https://webapi2.mercante.co.uk/Order/CheckShippingStatus
URL (test)	http://webapi.mercante.co.uk/Order/CheckShippingStatus
Method	GET
Header	Content-Type = application/json
Parameter	batchGuid

This method requires that the JWT bearer token returned by Authenticate is added as an Authorization header. In C# using RestSharp, this looks like this:

```
request.AddHeader("Authorization", $"bearer {yourtoken}");
```

This method returns a JSON object representing the shipping data:

```
{
   "orderNumber": 797017,
   "shippedOn": "2020-09-24T15:54:09",
   "shippingTrackingNumber": "",
   "batchGuid": "3be96cc5-a740-48a2-ae8c-xxxxxxxxxxxx"
}
```

In C#, a call to this method can be achieved as follows:

```
public class ShippingDetails
{
    public int OrderNumber { get; set; }
    public DateTime? ShippedOn { get; set; }
    public string ShippingTrackingNumber { get; set; }
    public Guid BatchGuid { get; set; }
}

var client = new
      RestClient($"https://webapi2.mercante.co.uk/Order/CheckShippingStatus?batc
hGuid={yourbatchguid}");
var request = new RestRequest(Method.GET);
request.AddHeader("Authorization", $"bearer {yourtoken}");
request.AddHeader("Content-Type", "application/json");
var response = client.Execute(request);
ShippingDetails entity = new ShippingDetails();
if (response.IsSuccessful)
{
    entity = JsonConvert.DeserializeObject<ShippingDetails>(response.Content);
}
```