OGOship API document

Version 1.3

Pekka Ylenius

pekka.ylenius@ogoship.com

Contents

Changes

Version	Date	Description
1.2	16.4.2018	Stockupdate API changes. New errorcodes.
1.3	19.4.2018	OGOship => OGOship

General info

All interfaces work using XML and JSON.

Use **ContentType** to select XML or JSON (“application/xml” or “application/json”).

You will receive a response in the same format.

Most of the samples are done using XML only. But you can still use JSON everywhere.

All API requests require authentication using SHA1 hash.

There is a backdoor which can be used when testing APIs manually.

Just use parameter SHA1=Dem0 to authenticate.

Later when that Merchant is really taken into use, you need to send correct hash…

Reference implementations and readymade libraries

We will publish a reference library for PHP. With the library it is easy to integrate any service with OGOship. When using that library you do not need to worry about using API or SHA1 tokens. You can use typed classes for products and orders. Those classed have get and set methods to all needed fields. When you have set all the needed properties then you just call the update function and then a product or an order will save its data to OGOship. Or you can ask for a product or order info by using your code of a product or an order.

How integration to your service should be done

There exists a list function which you should schedule your service to run periodically for example every 10 minutes. That function will return then all the changes done since your last request. A change can be a stock level change or an order status change. You will receive all changed orders and products with the same request.

Normally you would send new orders immediately when those can be shipped. Another option would be to schedule sending of those orders also periodically at the same time as requesting for changes. Choose whichever process works best for you.

Using the PHP library

First you initialize the library with “Merchant token” and “Secret token” and you will receive a live instance of the library.

Using that instance you can make actions to OGOship. Requests can be the same as described later in this document. Full specification of the library will be released with the library.

Also .Net library coming

We are developing a mobile application for Windows Phone (for example Nokia Lumias). A service API library used with that application will be published also when it is done.

Best practises

Best practice would be to have two different ways to call OGOship. Adding new orders would happen synchronously and reading order and stock changes would happen periodically for example every 30 minutes.

Levels of integration

We have defined levels of integration. It depends on merchants which level of integration they require. These are examples:

Minimum integration

- Add new orders “minimum”
    - Name and address of customer
    - List of order lines
    - One default shipping method
    - Error checking whether the order transfer has been successful.
- Follow existing orders
    - Inform customer when order is shipped (email from webstore or use email sending system of OGOship)
    - Copy parcel tracking info from order.

Standard integration (in addition to Minimum)

- Add new orders “full”
    - Send selected shipping method also
- Cancel unshipped orders
    - Allow merchant to cancel orders which are not shipped (=NEW / DRAFT-state).
- Update product stock info to webstore
    - Client is able to see if product is available in stock.
- Copy product info between webstore and OGOship (semi)automatically
    - The best way usually is that products are created to the webstore and added to OGOship via the API. Two way transfers also possible.

Advanced integration (in addition to Standard)

- Send stock updates of incoming shipments to OGOship.

User interface for the merchant

The user interface for the merchant should contain some information that is asked from OGOship. The goal of this information is that the merchant should not have to use OGOship’s web UI daily but should be able to maintain daily activities with the webstore’s UI.

The webstore UI should contain:

- Information whether the order has been added successfully to OGOship.
    - If the addition is unsuccessful, a clear notification should be displayed in a general order list and within the order details.
    - A message (e.g. email) to the merchant should be considered if an adding is unsuccessful
    - An error message for an unsuccessful adding (product code not found etc.).
- Order status at OGOship (NEW / SHIPPED / CANCELLED / PENDING and why is it PENDING / RETURNED).

- Is a product code present at OGOship.
- Display the <Comments2> field per order.

Functionalities that the merchant should be able to do:

- Orders:
  - Add an order to OGOship manually.
  - Cancel an order.
- Products:
  - Add a product and a product code to OGOship manually.
  - Update product information, e.g. product name, price.

Settings that can be altered by the merchant

- Are all products (product codes and info) added to OGOship or not.
- Are all product groups shipped from OGOship / which orders are added to OGOship.

Doing periodic updates from OGOship

LatestChanges request should be scheduled to run periodically every 30 minutes. That request will return all changed Orders and Products since previous request.

LatestChanges request will return TimeStamp which should be saved somewhere and sent back when doing next request. Using that TimeStamp, the service can return only changes occurred since previous request.

Doing synchronous updates to OGOship

Adding new orders should be done synchronously after the customer has paid an order or otherwise approved it to be shipped.

Cancelling of order should be synchronous to make sure the order can really be cancelled. Orders which are in collecting or shipped state can't be cancelled automatically anymore.

Testing OGOship integration

You can create new account for testing purposes yourself. Just register to our https://my.ogoship.com/ website and create a new merchant. Creating new users or merchants is free.

Note! To be able to do proper integration testing, your user account needs special API tester permissions. To receive those, send an email to pekka.ylenius@ogoship.com and inform us of your OGOship username.

With proper tester permissions you can simulate whole process of collecting and shipping orders. If you specify TestOnly attribute to Order you are allowed to mark orders as collecting and shipped.

You can assign your products to a stock location called “Demo”. Those stock locations are allowed to be freely used for testing purposes. You need to add a few items to be available in stock before you can mark Orders containing those Products as Collecting or Shipped.

URLs and methods

We are using restful style in our APIs.

Usually URL stays the same and HTTP Method describes what should be done.

Here is list of HTTP methods. In this list a resource can be an order, a product or something else.

HTTP Methods	Description
GET	Get info about resource. This will not do any changes to resource.
PUT	Create new resource. This will not override existing resource.
POST	Update existing resource. This will not create new resource.
DELETE	Delete existing resource.

Responses

All responses from the server are inside a container called “Response”.

Also all responses have a node called Info. Info has the following attributes.

Name in XML	Name in JSON	Description
Success	@Success	“true” or “false” will tell if operation was successful.
Timestamp	@Timestamp	Unix style timestamp of server.
Error	@Error	Error message. This attribute is visible only if there was error in processing request.

Json Sample

```
{“Response”:{
“Info”:{
  “@Success”:”true”,
  “@Timestamp”:”1310653904.4377611”},
…
}}
```

XML Sample

```
<Response>
  <Info Success=”true” Timestamp=”1310592779.6092734” />
…
</Response>
```

Orders

Order status

Normally it is not possible for a merchant to change the status after an order is in collecting state. For testing purposes there exists a 'test only' option. When an order is in testing, then it is possible for a merchant to test all the statuses.

When new order is created it will go automatically to new state. If there is issue with shipping code when new order will be accepted but it will be put into draft state.

Status	Description
DRAFT	Can be used when manually creating an order. Order is editable in Web. Warehouse will not do anything for orders in this state.
RESERVED	Products are reserved from stock. Orders in this state will not be processed.
NEW	Order is waiting to be collected and shipped.
COLLECTING	Warehouse is collecting products. (It is not possible to cancel order anymore using Api.)
PENDING	Order is pending for some products.
CANCELLED	Order is cancelled.
SHIPPED	Order is shipped.

Adding new orders or updating existing orders

Similar schema can be used when adding new orders or when updating existing ones. When updating an order it is only needed to send the modified data. For example, you can just post a new status and everything else will be left as it is.

Existing orders can be edited before they are in collecting state.

Order URLs

All requests return full content of an order specified using URL

URL	/merchant/*MerchantID*/order/*Reference*?SHA1=*SHA1token*
Method	Use PUT to add new order Use POST to update existing order Use GET to just read info of order Use DELETE to remove order before it is collected
Reference	Reference number of order
MerchantID	Unique ID generated for merchant
SHA1	Is calculated from string "order,add,OrderID,SecretToken" , "order,update,OrderID,SecretToken" , "order,info,OrderID,SecretToken" and "order,remove,OrderID,SecretToken"

Order schema

Property	Required	Type	Description
Shipping	No	string	Name of shipping method

ShippingCode	No	string	Code of shipping method which you have enabled at Edit Merchant page. This is required if more than 1 shipping method is selected.
PickUpPointCode	No	string	Code of pickup point. Used with “Matkahuolto Lähellä” and “Itella SmartPost”.
Reference	N/A	string	Reference number of order
Status	No	string	Status of order
PriceTotal	No	decimal	Price is needed for “Postiennakko” and “Matkaennakko”
PriceCurrency	No	string	Currency of PriceTotal and UnitPrice (of OrderLine).
TrackingNumber	N/A	string	Warehouse will assign tracking number when available.
Comments	No	string	Write any additional comments about order.
Comments2	N/A	string	Comments from OGOship.
TestOnly	No	boolean	“true”/”false” Set to true for testing purposes.
Customer	Yes	Customer	See details below
OrderLines	Yes	OrderLine[]	Required for new order. If given when updating order then all order lines will be replaced with the ones sent with update. See details below.
Documents	No	Document[]	Not required. If given when updating order then all documents will be replaced with the ones sent with update. See details below.
PaymentType	No	string	Free text name of payment type.
CashOnDelivery	No	CashOnDelivery	This element is required for cash on delivery orders

CashOnDelivery schema

Property	Required	Type	Description
Reference	No	string	Bank reference (Order reference number + required validation digits are used if not specified)
Amount	Yes	decimal	Amount requested from customer
Currency	No	string	Currency of amount. (ISO 4117 Code). Default value EUR will be used if not specified.

Customer schema

Property	Required	Type	Description
Name	Yes	string	
Company	No	string	
Address1	Yes	string	
Address2	No	string	
City	Yes	string	
Country	Yes	string	Use two-letter codes: ISO 3166-1 alpha-2
Zip	No	number	
Phone	No	string	
Email	No	string	

OrderLine schema

Property	Required	Type	Description
Code	Yes	string	Code of product
Code2	No	string	2nd part of product code
Quantity	Yes	integer	Quantity of products
UnitPrice	No	decimal	Sales price of single product. (Price including VAT.)
VatPercentage	No	int	Percentage of VAT included in sales price (UnitPrice).
ProductInfoUrl	No	string	Full url of product info page. If there are lots of similar products then warehouse staff can use this page to verify products before shipping.
ProductPictureUrl	No	String	Full url of product picture. If there are lots of similar products then warehouse staff can use this page to verify products before shipping.

Document schema

Property	Required	Type	Description
Type	Yes	string	Name of type of document, e.g. "receipt". Documents with type "receipt" will be automatically printed and attached to all deliveries. (This can be changed).
URL	Yes	string	Full url of document.

Sample

Content sent using order PUT or POST methods could look like this:

```
<Order>
  <Shipping>Posti</Shipping>
  <PriceTotal>1000.00</PriceTotal>
  <PriceCurrency>EUR</PriceCurrency>
  <Comments>Tähän voi kirjoittaa tekstiä.</Comments>
  <TestOnly>true</TestOnly>
  <Documents>
    <Document>
      <Type>receipt</Type>
      <URL>http://Foo.bar.com/doc1.pdf</URL>
    </Document>
    <Document>
      <Type>instructions</Type>
      <URL>http://Foo.bar.com/instructions.pdf</URL>
    </Document>
  </Documents>
  <Customer>
    <Name>Heikki Heikkinen</Name>
    <Company>Heikki Heikkinen</Company>
    <Address1>Kotikati 1</Address1>
    <Address2>Yläkerta</Address2>
    <City>Nowhere</City>
    <Country>fi</Country>
    <Zip>12345</Zip>
    <Phone>+358 50 123 2345</Phone>
    <Email>foo@bar.com</Email>
  </Customer>
  <CashOnDelivery>
    <Reference>10003</Reference>
    <Amount>1000.50</Amount>
    <Currency>EUR</Currency>
  </CashOnDelivery>
  <OrderLines>
    <OrderLine>
      <Code>PUKA</Code>
      <Quantity>1</Quantity>
      <UnitPrice>13.00</UnitPrice>
      <VatPercentage>24</VatPercentage>
    </OrderLine>
    <OrderLine>
      <Code>bb</Code>
      <Quantity>3</Quantity>
      <UnitPrice>999.00</UnitPrice>
      <VatPercentage>24</VatPercentage>
    </OrderLine>
  </OrderLines>
</Order>
```

And response received back could look like this:

```
<Response>
  <Info Success=”true” Timestamp=”1310592779.6092734” />
  <Order>
    <Shipping>Posti</Shipping>
    <Reference>1</Reference>
    <Status>NEW</Status>
    <PriceTotal>1000.00</PriceTotal>
    <TrackingNumber>AS123123B22</TrackingNumber>
    <Comments>Tähän voi kirjoittaa tekstiä.</Comments>
    <Customer>
      <Name>Heikki Heikkinen</Name>
      <Company>Heikki Heikkinen</Company>
      <Address1>Kotikati 1</Address1>
      <Address2>Yläkerta</Address2>
      <City>Nowhere</City>
      <Country>fi</Country>
      <Zip>12345</Zip>
      <Phone>+358 50 123 2345</Phone>
      <Email>foo@bar.com</Email>
    </Customer>
    <OrderLines>
      <OrderLine>
        <Code>PUKA</Code>
        <Quantity>1</Quantity>
      </OrderLine>
      <OrderLine>
        <Code>bb</Code>
        <Quantity>3</Quantity>
      </OrderLine>
    </OrderLines>
    <Documents>
      <Doc1>http://Foo.bar.com/doc1.doc</Doc1>
      <Document>http://Foo.bar.com/doc2.doc</Document>
    </Documents>
  </Order>
</Response>
```

Products

Product URLs

All requests return full info of product specified using URL

URL	/merchant/*MerchantID*/Product/ProductCode?SHA1=*SHA1token* /merchant/*MerchantID*/Product/ProductCode/ProductCode2?SHA1=*SHA1token*
Method	Use PUT to add new product Use POST to update existing product Use GET to just read info of product Use DELETE to remove product. Count of products in stock must be zero.
ProductCode	Product code which must be unique
ProductCode2	Product code part 2. (If specified then code + code2 must be unique.)
MerchantID	Unique ID generated for merchant
SHA1	Is calculated from string "product,add,ProductCode,SecretToken" , "product,update,ProductCode,SecretToken" , "product,info,ProductCode,SecretToken" and "product,remove,ProductCode,SecretToken"

Adding new products or updating existing products

Similar schema can be used when adding new products or when updating existing ones. When updating a product it is only needed to send the modified data. For example, you can just post a new name and everything else will be left as it is.

Note. Currently we don't take copies of product info when saving orders. So when renaming products also product lists in old orders will be changed. This might change in the future if there is any need to keep the old orders untouched even if product info has changed.

Product schema

Property	Required	Type	Description
Name	Yes	string	Display name of product
Description	No	string	Additional information about product
ShortDescription	No	string	Short description of product
Code	No	string	Part1 of product code
Code2	No	String	Part2 of product code. (Not required)
SupplierCode	No	string	Supplier given code of this product
Supplier	No	string	Name of supplier
Group	No	string	Group of product
EANCode	No	string	EAN code of product
Width	No	int	Width of product (mm)
Height	No	int	Height of product (mm)
Depth	No	int	Depth of product (mm)
Weight	No	int	Weight of product (g)
AlarmLevel	No	int	Merchant can receive reports if stock is below this alarm level
Stock	N/A	int	Count of products in stock
StockAvailable	N/A	int	Count of products available for orders

Price	No	decimal	Supply price of product for calculating value of stock
SalesPrice	No	decimal	Sales price of product
VatPercentage	No	int	Vat percentage included into Price and SalesPrice
Currency	No	string	Currency of Price and SalesPrice
Reserved	N/A	int	Count of products reserved for not shipped orders
InfoUrl	No	string	Url of product page at webstore. This helps warehouse staff to recognize products. Very important!
PictureUrl	No	string	Url of product picture at webstore. This helps warehouse staff to recognize products. Very important!
StockUpdate	N/A	int	Quantity of new products coming to stock.
StockUpdateTime	N/A	int	Unix style time estimate of new stock update coming.
EditTime	N/A	int	Unix style timestamp of last change made to this product.
CountryOfOrigin	No	string	2 char iso code of country of origin for customs info
CustomsDescription	No	string	Infotext for customs documents
AdditionalPicture	No	AdditionalPicture	Additional pictures of product

AdditionalPicture schema

Property	Required	Type	Description
Url	Yes	string	Url of image

Sample

Content sent using product PUT or POST methods could look like this:

```
<Product>
  <Name>Leijona</Name>
  <Description>-</Description>
  <SupplierCode>LE-1</SupplierCode>
  <Group>ANIMALS</Group>
  <EANCode>123456</EANCode>
  <Width>1000</Width>
  <Height>600</Height>
  <Depth>400</Depth>
  <Weight>100000</Weight>
  <AlarmLevel>6</AlarmLevel>
  <Price>100.00</Price>
  <SalesPrice>1000.00</SalesPrice>
  <VatPercentage>24</VatPercentage>
  <Currency>EUR</Currency>
  <CountryOfOrigin>FI</CountryOfOrigin>
  <CustomsDescription>Animal</CustomsDescription>
  <InfoUrl>http://foo.bar/lei</InfoUrl>
  <PictureUrl>http://foo.bar/image.jpg</PictureUrl>
</Product>
```

And response received back could look like this:

```
<Response>
  <Info Success=”true” Timestamp=”1310592779.6092734” />
  <Product>
    <Code>LE-1</Code>
    <Code2></Code2>
    <SupplierCode>LEI</SupplierCode>
    <Supplier>Zoo</Supplier>
    <EANCode>123456</EANCode>
    <Width>1000</Width>
    <Height>600</Height>
    <Depth>400</Depth>
    <Weight>100000</Weight>
    <AlarmLevel>6</AlarmLevel>
    <EditTime>1310592779</EditTime>
    <Name>Leijona</Name>
    <Description>-</Description>
    <Culture></Culture>
    <Stock>7</Stock>
    <Reserved>2</Reserved>
    <StockAvailable>5</StockAvailable>
    <Group>ANIMALS</Group>
    <Stock>7</Stock>
    <InfoUrl>http://foo.bar/lei</InfoUrl>
    <PictureUrl>http://foo.bar/image.jpg</PictureUrl>
    <Price>100.00</Price>
    <SalesPrice>1000.00</SalesPrice>
    <VatPercentage>24</VatPercentage>
    <Currency>EUR</Currency>
    <StockUpdate>12</StockUpdate>
    <StockUpdateTime>1310592779</StockUpdateTime>
    <CountryOfOrigin>FI</CountryOfOrigin>
    <CustomsDescription>Animal</CustomsDescription>
    <AdditionalPicture><Url>http://foo.bar/image.jpg</Url></AdditionalPicture>
    <AdditionalPicture><Url>http://foo.bar/image2.jpg</Url></AdditionalPicture>
  </Product>
</Response>
```

List or update all products

All requests return full info of product specified using URL

URL	/merchant/*MerchantID*/Products?SHA1=*SHA1token*
Method	Use PUT or POST to add new and update existing products Use GET to just read info of products
MerchantID	Unique ID generated for merchant
SHA1	Is calculated from string “product,all,SecretToken”

Products which exist already are just updated. Products which do not exist are added.

Sample

Content sent using product PUT or POST methods could look like this:

```
<Products>
  <Product>
    <Name>Leijona</Name>
    <Code>LE-1</Code>
    <Code2></Code2>
    <Description>-</Description>
    <SupplierCode>LE-1</SupplierCode>
    <EANCode>123456</EANCode>
    <Width>1000</Width>
    <Height>600</Height>
    <Depth>400</Depth>
    <Weight>100000</Weight>
    <AlarmLevel>6</AlarmLevel>
    <Price>100.00</Price>
    <SalesPrice>1000.00</SalesPrice>
    <VatPercentage>24</VatPercentage>
    <Currency>EUR</Currency>
    <CountryOfOrigin>FI</CountryOfOrigin>
    <CustomsDescription>Animal</CustomsDescription>
    <InfoUrl>http://foo.bar/lei</InfoUrl>
    <PictureUrl>http://foo.bar/image.jpg</PictureUrl>
  </Product>
<Products>
```

Json version would look like this

```
{
  "Products": {
    "Product": [
      {
        "Code": "LE-1",
        "Name": "Leijona",
        "Currency": "EUR"
      },
      {
        "Code": "LE-2",
        "Name": "Leijona2",
        "Currency": "EUR"
      }
    ]
  }
}
```

And response received back could look like this:

```
<Response>
  <Info Success=”true” Timestamp=”1310592779.6092734” />
  <Product>
    <Code>LE-1</Code>
    <Code2></Code2>
    <SupplierCode>LEI</SupplierCode>
    <Supplier>Zoo</Supplier>
    <EANCode>123456</EANCode>
    <Width>1000</Width>
    <Height>600</Height>
    <Depth>400</Depth>
    <Weight>100000</Weight>
    <AlarmLevel>6</AlarmLevel>
    <EditTime>1310592779</EditTime>
    <Name>Leijona</Name>
    <Description>-</Description>
    <Culture></Culture>
    <Stock>7</Stock>
    <Reserved>2</Reserved>
    <StockAvailable>5</StockAvailable>
    <Group>ANIMALS</Group>
    <Stock>7</Stock>
    <InfoUrl>7</InfoUrl>
    <PictureUrl>7</PictureUrl>
    <Price>100.00</Price>
    <SalesPrice>1000.00</SalesPrice>
    <VatPercentage>24</VatPercentage>
    <Currency>EUR</Currency>
    <StockUpdate>12</StockUpdate>
    <StockUpdateTime>1310592779</StockUpdateTime>
    <CountryOfOrigin>FI</CountryOfOrigin>
    <CustomsDescription>Animal</CustomsDescription>
    <AdditionalPicture><Url>http://foo.bar/image.jpg</Url></AdditionalPicture>
    <AdditionalPicture><Url>http://foo.bar/image2.jpg</Url></AdditionalPicture>
  </Product>
</Response>
```

List all products (simple response)

URL	/merchant/*MerchantID*/Products?SHA1=*SHA1token&ResponseType=Simple*
Method	Use PUT or POST to add new and update existing products Use GET to just read info of products
MerchantID	Unique ID generated for merchant
SHA1	Is calculated from string “product,all,SecretToken”

Response

And response received back could look like this:

```
<Response>
  <Info Success=”true” Timestamp=”1310592779.6092734” />
  <Product>
    <Code>LE-1</Code>
    <Code2></Code2>
    <EditTime>1310592779</EditTime>
    <Stock>7</Stock>
    <Reserved>2</Reserved>
    <StockAvailable>5</StockAvailable>
    <StockUpdate>12</StockUpdate>
    <StockUpdateTime>1310592779</StockUpdateTime>
  </Product>
</Response>
```

Latest changes

To follow all changes happening in OGOship your service needs to poll the latest changes method. You can for example schedule a task to query for latest changes every 15minutes. For requests you need to send a timestamp of the previous request as parameter. That way you can be sure that you do not miss any changes.

This method retrieves all latest changes from Orders, Products or both.

Changes URL

URL	/Merchant/*MerchantID*/LatestChanges? SHA1=*SHA1token&TimeStamp=Timestamp* /Merchant/*MerchantID*/Product/LatestChanges? SHA1=*SHA1token&TimeStamp=Timestamp* /Merchant/*MerchantID*/Order/LatestChanges? SHA1=*SHA1token&TimeStamp=Timestamp*
Method	GET
Timestamp	Only give changes after timestamp
MerchantID	Unique ID generated for merchant
SHA1	Is calculated from string “changes,TimeStamp,SecretToken”

Response follows same Order and Product schemas specified earlier.

Here is one possible sample response:

```
<Response>
  <Info Success=”true” Timestamp=”1310595606.1569428” />
  <Orders>
    <Order>…</Order>
    <Order>…</Order>
    <Order>
      <Shipping>Posti</Shipping>
      <Reference>11</Reference>
      <Status>NEW</Status>
      <Customer>
        <Name>Heikki Heikkinen</Name>
        <Company>koti ab</Company>
        <Address1>Kotikati 1</Address1>
        <Address2>Yläkerta</Address2>
        <City>Nowhere</City>
        <Country>fi</Country>
        <Zip>12345</Zip>
        <Phone>+358 50 123 2345</Phone>
        <Email>foo@bar.com</Email>
      </Customer>
      <OrderLine>
```

```
        <Code>PUKA</Code>
        <Quantity>1</Quantity>
      </OrderLine>
      <OrderLine>
        <Code>bb</Code>
        <Quantity>3</Quantity>
      </OrderLine>
    </Order>
  </Orders>
  <Products>
    <Product>
      <Name>Leijona</Name>
      <Description>-</Description>
      <ManufacturerCode>LE-1</ManufacturerCode>
      <EANCode>123456</EANCode>
      <Width>1000</Width>
      <Height>600</Height>
      <Depth>400</Depth>
      <Weight>100000</Weight>
      <AlarmLevel>6</AlarmLevel>
    </Product>
  </Products>
</Response>
```

StockUpdate

StockUpdates may be made before shipping products to OGOship. By using stock updates we can check that nothing is lost during shipping. A stock update is also needed so that the staff at OGOship is able to know whose products arrive. While doing stock updates you can also create new products to OGOship. You can use stock updates to easily see which products are under warning level and insert those to new stock updates.

Status	Desription
Draft	Can be used when manually creating StockUpdate. Warehouse will not do anything for StockUpdates in Draft state. Merchant is able to edit StockUpdate when it is this state.
Waiting	Warehouse is waiting for products to arrive. Merchant is able to cancel shipment in waiting state.
Received	Warehouse has received shipment. All stock values have not been updated yet.
In Stock	Warehouse has fully handled receiving products and updated stocks.
Cancelled	StockUpdate is cancelled.

Stock update URLs

All requests return full content of stock update specified using URL

URL	/merchant/*MerchantID*/stockupdate/*StockUpdateID*
Method	Use PUT to add new stock update Use POST to update existing stock update Use GET to just read info of stock update Use DELETE to remove order before it is collected

StockUpdateID	Reference number of stock update
MerchantID	Unique ID generated for merchant
SHA1	No sha hash needed for stock update

Stock update schema

Property	Required	Type	Description
Status	No	string	Status of stock update
Reference	N/A	string	Reference number of stock update is used from url
ReceiveDate	Yes	integer	Date when products will be delivered to warehouse (unix type datetime)
Supplier	Yes	string	Supplier of products
Comments	No	string	Additional comments for warehouse
Products	Yes	ProductInfo	Info about products to added to stock

ProductInfo schema

Property	Required	Type	Description
Code	Yes	string	Unique code of product
Name	No/Yes	string	Name of product. Required for new products.
Quantity	Yes	integer	Number of products ordered
Received	N/A	integer	Number of products received by warehouse
Comments	No	string	Additional comments for warehouse
Price	No	decimal	Price of product for calculating value of stock
SupplierCode	No	string	ProductCode for supplier

Sample

Content sent using order PUT or POST methods could look like this:

```
<StockUpdate>
  <Status>NEW</Status>
  <Reference>MyStockUpdate</Reference>
  <Supplier>Apple</Supplier>
  <ReceiveDate>1232343456</ReceiveDate>
  <Products>
    <ProductInfo>
      <Code>LI-1</Code>
      <Name>Lion big</Name>
      <Quantity>10</Quantity>
      <SupplierCode>LLI</SupplierCode>
    </ProductInfo>
    <ProductInfo>
      <Code>BLI</Code>
      <Name>Lion baby</Name>
      <Quantity>30</Quantity>
      <SupplierCode>BLI</SupplierCode>
    </ProductInfo>
  </Products>
</StockUpdate>
```

Add new Merchant

It is possible to add a new merchant though the API to make it very easy for new customers to start using OGOship.

The WebStore might know most of the needed information for registration already. Therefore, in the best possible case, only a click or two are needed from the merchant.

Add merchant URLs

Requests returns full content of newly created merchant

URL	/merchant
Method	Use PUT to add new stock update

New merchant schema

Property	Required	Type	Description
Name	Yes	string	Name of merchant
StreetLine1	No	string	First line of street address of merchant
StreetLine2	No	string	Second line of street address of merchant
PostCode	No	string	Post code of merchant
City	No	string	City of merchant
Country	No	string	Country of merchant
BankAccount	No	string	Bank account for Cash On Delivery orders (IBAN)
BicCode	No	string	Bic Code of bank
Id	N/A	string	MerchantID of new merchant
SecretToken	N/A	string	Secret token generated for a merchant
User	Yes	NewUser	Users which are linked to new merchant
ResponsiblePerson	Yes	ContactDetails	Contact details of person who will sign final agreement between OGOship and new merchant.

NewUser schema

Property	Required	Type	Description
UserName	No	string	Username of user. If username is empty or missing then email will be used as username.
Email	Yes	string	Email of user
Password	No	string	Password of user. If password is missing or empty then new password will be generated and sent as email to user.

ContactDetails schema

Property	Required	Type	Description
Name	Yes	string	Name of person
Email	Yes	string	Email of person
Phone	Yes	string	Phone number of person

Sample

Content sent using merchant PUT method could look like this:

```
  <Merchant>
    <Name>Api test</Name>
    <StreetLine1>Street 1</StreetLine1>
    <StreetLine2>Upper floor</StreetLine2>
    <PostCode>12345</PostCode>
    <City>City</City>
    <Country>Finland</Country>
    <BankAccount>12 124122 234345 345345</BankAccount>
    <BicCode>NDEAFIN</BicCode>
    <Id>321c73bd-9a40-4bca-898d-d22a0002d8b6</Id>
    <SecretToken>482e3233-0858-4a3f-ac03-e6d5208814d2</SecretToken>
    <User>
      <UserName>user</UserName>
      <Email>foo@bar.com</Email>
      <Password>foAz733!</Password>
    </User>
    <User>
      <UserName>user2</UserName>
      <Email>foo2@bar.com</Email>
      <Password>foAz7323!</Password>
    </User>
    <ResponsiblePerson>
      <Name>First Lastname</Name>
      <Email>foo@bar.com</Email>
      <Phone>+35840123456</Phone>
    </ResponsiblePerson>
  </Merchant>
```

And response received back could look like this:

```
<Response>
  <Info Success="true" Timestamp="1384634732.3910432" />
  <Merchant>
    <Name>Api test</Name>
    <StreetLine1>Street 1</StreetLine1>
    <StreetLine2>Upper floor</StreetLine2>
    <PostCode>12345</PostCode>
    <City>City</City>
    <Country>Finland</Country>
    <BankAccount>12 124122 234345 345345</BankAccount>
    <BicCode>NDEAFIN</BicCode>
    <Id>321c73bd-9a40-4bca-898d-d22a0002d8b6</Id>
    <SecretToken>482e3233-0858-4a3f-ac03-e6d5208814d2</SecretToken>
    <User>
      <UserName>user</UserName>
      <Email>foo@bar.com</Email>
    </User>
    <User>
      <UserName>user2</UserName>
      <Email>foo2@bar.com</Email>
    </User>
  </Merchant>
</Response>
```

Errorcodes

Code	Description
Annn	Authentication specific errors
Mnnn	Merchant specific errors
Onnn	Order specific errors
Pnnn	Product specific errors
Unnn	Stock update specific errors
A001	Merchant not specified!
A002	Merchant not in correct format!
A030	SHA1 not specified!
A031	SHA1 Hash not valid!
M009	Merchant not found!
M010	Merchant is suspended!
O010	Order with ID xxx not found!
O011	Order with same ID already exists!
O015	Order cannot be updated anymore!
O016	Not valid next state for Order!
O020	Order reference not specified!
O030	No 'OrderLine' elements found.
O031	Element 'OrderLine/Code' is missing.
O032	Element 'OrderLine/Code' is empty or whitespace.
O033	Element 'OrderLine/Quantity' is missing.
O034	OrderLine/Quantity element is not integer.
P010	Product with code "xxx" not found!
P011	Product with same ID already exists!
P016	Product with code "xxx" cannot be removed!
P020	Product code not specified!
O040	CashOnDelivery 'Amount' is missing.
O041	CashOnDelivery 'Amount' is not valid.
O050	Customer 'Name' is missing.
O051	Customer 'Address1' is missing.
O052	Customer 'City' is missing.
O053	Customer 'Country' is missing.
O055	'Customer' is missing.
P021	Update product only once in single request.
U015	StockUpdate can't be updated anymore.
U016	Status nnn is not valid next status when stockupdate in mmm status. Valid statuses are: (a,b,c)

U030	No 'ProductInfo' elements found.
U031	Element 'ProductInfo/Code' is missing.
U032	Element 'ProductInfo/Code' is empty or whitespace.
U043	Reference is required.
U044	ReceiveDate is required.
U045	Supplier is required.
U061	Element 'ProductInfo/Name' is missing for new product.