eMAG Marketplace
API Documentation v4.4.4

09.12.2021

Version	Date modified	Changes
4.1.2	25.09.2017	Added return_reason and return_type description details Added type key on order read and order filtering Updated product validation_status values Added info about start_date limitations during special campaigns
4.1.3	13.02.2018	Updated product stock maximum value Updated order attachment types Added EAN method for attaching offers on existing products Added billing_name and billing_phone on order/read Added awb_format on awb/save
4.2.0	01.03.2018	Added a limit for the number of entities per request New API endpoints available with 1st of May 2018 Removed awb_format on awb/save
4.2.1	20.03.2018	Added time based throttling for save requests starting with 1st of May 2018 Added offer_validation_status key on product_offer/read and filter
4.2.2	08.05.2018	Removed product statuses from return requests Updated return request status change permissions API endpoints availability change, now beginning with 1st of August 2018 Removed "Invoices", "Details" and "Observation" keys from order fields
4.3.0	03.10.2018	Removed mandatory label from product url Added AWBs key on rma/read Extended AWB read method: AWB number, status, type, courier name, courier ID and courier account ID Added new method for downloading AWB .zpl format types Added type key on rma/read Added attachments, cashed_co and cashed_cod keys on order/read Added callback URLs section Added rma_id on awb/save
4.3.1	05.02.2019	Added new resource for saving volume measurments Added other currencies option on product_offer/save Added locker delivery option on order/read and awb/save Updated the time based throttling for save requests
4.3.2	18.02.2019	Order status matrix update Important changes in permisions for updating orders (3.5) and removing products from an order (3.5.1) Added supply_lead_time key on product_offer/save
4.3.4	01.10.2019	Added rate limiting for invalid requests Added new order cancelation reasons Added "locker_name" key on order/read Adjusted the maximum number of elements returned for all read requests Adjust measurements values decimal length
4.3.5	08.10.2019	Changed the warranty requirement regime
4.3.6	03.12.2019	Part Number input changes
4.3.7	28.09.2020	Added new REST resource for updating stock only Added new resource for proposing offers in campaigns
4.3.8	13.10.2020	Added new invoice resources
4.3.9	23.11.2020	Added new resource for reading customer invoice data
4.4.0	26.04.2021	Added "source_language" key on product_offer/save for publishing product content in a different language than the platforms default
4.4.1	28.04.2021	Added voucher discounts distribution on order/read
4.4.2	16.06.2021	Added "detailed_payment_method" key on order/read Added Draft publishing API details Changed product ownership logic
4.4.3	08.12.2021	Added new API for reading commission at offer level Added "return_reason" and "observations" keys at product level on rma/read and rma/save Changed reading_rma and saving_rma examples (7.4)
4.4.4	09.12.2021	Added "language" parameter on category/read

Table of Contents

1. eMAG Marketplace API

eMAG Marketplace API is developed by eMAG for Marketplace partners in order to allow them to use their own CRM's / ERP's. This document explains the methods for calling the API.

The API can be used in order to:
- send products and offers
- process orders

1.1. Conventions

We define MARKETPLACE_API_URL constant of being the API URL of the platform (ex: https://marketplace-api.emag.ro/api-3)

We define MARKETPLACE_URL constant of being the URL of the platform (ex: https://marketplace.emag.ro)

We define DEFAULT_CURRENCY constant of being the default currency of the platform (ex: RON).

All API parameters are key-sensitive.

Platform	Romania	Bulgaria
MARKETPLACE_URL	https://marketplace.emag.ro	https://marketplace.emag.bg
MARKETPLACE_API_URL	https://marketplace-api.emag.ro/api-3	https://marketplace-api.emag.bg/api-3
Protocol	HTTPS	HTTPS
Locale	ro_RO	bg_BG
Default currency	RON	BGN

Platform	Hungary	Poland
MARKETPLACE_URL	https://marketplace.emag.hu	https://marketplace.emag.pl
MARKETPLACE_API_URL	https://marketplace-api.emag.hu/api-3	https://marketplace-api.emag.pl/api-3
Protocol	HTTPS	HTTPS
Locale	hu_HU	pl_PL
Default currency	HUF	PLN

To access the API, simply make a Basic Authorization request with your username, password and a base64 computed hash. Please note that user should be granted API rights in order to access the API.

```
$hash = base64_encode($username . ':' . $password);
```

1.2. Request, resources and actions

A Marketplace API call is represented by sending a request to API URL of platform. Every request consists of a POST to an URL like:

MARKETPLACE_API_URL/**resource/action**
Ex: https://marketplace-api.emag.ro/api-3/product_offer/save

RESOURCES AND AVAILABLE ACTIONS

Resource	Resource URL	Available actions				
product_offer	MARKETPLACE_API_URL/product_offer	read	save	count		match
measurements	MARKETPLACE_API_URL/measurements		save			
offer_stock	MARKETPLACE_API_URL/offer_stock/{resourceId}					
campaign_proposals	MARKETPLACE_API_URL/campaign_proposals		save			
order	MARKETPLACE_API_URL/api-3/order	read	save	count	acknowledge	
order/attachments	MARKETPLACE_API_URL/order/attachments		save			
message	MARKETPLACE_API_URL/message	read	save	count		
category	MARKETPLACE_API_URL/category	read		count		
vat	MARKETPLACE_API_URL/vat	read				
handling_time	MARKETPLACE_API_URL/handling_time	read				
locality	MARKETPLACE_API_URL/locality	read		count		
courier_accounts	MARKETPLACE_API_URL/courier_accounts	read				
awb	MARKETPLACE_API_URL/awb	read	save			
rma	MARKETPLACE_API_URL/rma	read	save			
invoice/categories	MARKETPLACE_API_URL/api-3/invoice/categories	read				
invoice	MARKETPLACE_API_URL/api-3/invoice	read				
customer-invoice	MARKETPLACE_API_URL/api-3/customer-invoice	read				

Below a code example using the resource "category" and the action "read"

Resource	Example	Context
category/read	reading_categories. txt	http method: POST

The API needs authorization and has an IP level filtering. Before testing, sellers should provide a list of whitelisted IP's. eMAG Marketplace will only allow API calls only from those IP's.

The POST data consist of 1 mandatory key:

REQUEST	
Key	Description
data	Data to be passed to the API. The following document will describe keys.

1.3. Pagination and filters

In order to limit the number of items returned, read actions accept pagination by passing to POST data following parameters:

PAGINATION			
Key	**Description**	**Default value**	**Example**
currentPage	Set current page displayed	1	currentPage =3
itemsPerPage	Set number of items to be displayed in one page. Maximum is set to 100.	100	itemsPerPage=50

Also, filters can be included in POST to refine result set. Filters vary depending on the resource called and are exampled on every resource section.

1.4. Response

When an API call is made, the server MUST reply with a response. The response will ALWAYS be JSON formatted and the header 'Content-type: application/json' will always be passed.

RESPONSE	
Key	**Description**
isError	Boolean value representing response status.
messages	Messages included in the response, like error messages, etc.
results	Results included in the response, mostly when reading resources.

IMPORTANT: Every API request must have a response and the response must contain the key "isError" and its value must be "false". For each call that does not have the key and "false" value, we recommend setting up alerts, as the call most likely was not interpreted. We also recommend logging all calls and the corresponding API response for a 30 days period.

Every request must have at most 4000 elements. If the call surpasses this limit the call will have a response with key "isError:true" and "message: Maximum input vars of 4000 exceeded".

In the event of an documentation error when saving a product, the API will return and "isError:true" message, but the new offer will be saved and processed.

1.5. Rate limiting

All resources described at the table from paragraph 1.2 have the following limits:

- Maximum 1 request every 3 seconds and maximum 20 requests every 1 minute. For optimal performance we recommend not scheduling requests at fixed hours. For example use 12:04:42 as a starting point instead of 12:00:00. The following responses are possible:

Time throttling limit is exceeded	Time throttling limit was not reached
HTTP/1.1 429 Date: Wed, 21 Mar 2018 08:22:44 GMT Content-Type: application/json; charset=utf-8 Transfer-Encoding: chunked X-RateLimit-Limit-3second: 1 X-RateLimit-Remaining-3second: 0 Server: kong/0.12.1 {"message":"API rate limit exceeded"}	HTTP/1.1 200 Date: Wed, 21 Mar 2018 08:23:52 GMT Content-Type: application/json Transfer-Encoding: chunked X-RateLimit-Limit-3second: 1 X-RateLimit-Remaining-3second: 0 Server: nginx {"isError":false,"messages":[],"results":[]}

Invalid requests will also be limited to a maximum number of 3 requests every 1 minute or 180 requests every 1 hour.

- For API resources that accept bulk save, the limit is 50 entities per request. For optimal performance we recommend using between 10 and 50 entities per request.

1.6. Callback URLs

The following callback URLs can be activated from the Marketplace interface:

Context	Description
New order	You will be notified for each new order
Order cancellation	You will be notified whenever an order in canceled
New return & status change	You will be notified for each new return request and select status changes (New, Acknowledged & Received)
AWB status change	You will be notified for each AWB status change

2. Publishing products and offers

We define a **draft** product as a minimum set of details required to publish a product. The elements are:
- Name
- Brand
- Part number
- Category (optional)
- EAN (optional)
- Source language (optional)

We define a **product** as a list of elements displayed for a product page. The elements are:
- Name
- Brand
- Part number
- Description
- Images
- Product characteristics (and product families)
- Category
- Barcodes (optional)
- Other attachments (optional)
- EAN (required depending on Category)
- Source language (optional)

We define an **offer** as a list of elements required for an offer to be available for a product. These elements are:
- Price
- VAT rate
- Warranty
- Numerical stock
- Handling_time

eMAG Marketplace API allows a seller to:
- Send new products and offers
- Send new offers for existing eMAG products (sold by eMAG or any other seller)
- Update existing own offers and/or products

2.1. Reading categories, characteristics and family_types

Every eMAG product has to be included in a certain category. Sellers cannot create new categories or change existing ones. Also, a seller can only post products and offers in its allowed categories list.

Reading categories without parameters will generate a response containing the first 100 categories. The read can be paginated thus obtaining a full list of categories. Only active categories will be returned.

When passing a category id, the API will return the category name and the list of available characteristics and their corresponding IDs, as well as the available product family_types and their corresponding IDs. Reading categories one by one is important as it is the only way to identify the restrictive characteristics and corresponding allowed values.

You can read the categories and their characteristics through the API.

The resource is **category** and the available actions are **read** and **count**.

CATEGORY – read					
Key – level 1	**Key – level 2**	**Key – level 3**	**Description**	**Type**	**Example**
id			Category eMAG id	Integer	id=604
name			Category name	String	name="Music"
is_allowed			Indicates if the seller can send products and offers in the category. In order to request access to a specific category, you can use the Marketplace interface. 0 = No 1 = Yes.	Integer	is_allowed=0
parent_id			Id of the parent category	Integer	parent = 12
is_ean_mandatory			Indicates if the sending an EAN is mandatory when saving products • 1 = mandatory • 0 = optional	Integer	is_ean_mandatory =1
is_warranty_mandatory			Indicates if adding warranty is mandatory when saving products • 1 = mandatory • 0 = optional	Integer	is_warranty_mandatory =1
characteristics			All characteristics available in category	List of arrays	
	id		Characteristic eMAG id	Integer	id=38
	name		Characteristic name	String	name="Audio"
	type_id		Characteristic type. Indicates the type of values that a characteristic can accept. **Single value characteristics**: • 1 = Numeric (ex: 20, 1, 30, 40, etc) • 60 = Size (ex. 36 EU, XL INTL, etc) • 20 = Boolean (Yes, No, N/A) **Multiple values characteristics**: • 2 = Numeric + unit of measure (ex. 30 cm, 45 m, 20 GB, 30 inch, etc) • 11 = Text Fixed (max length 255 chars) (ex. Blue, Green,	Integer	type_id=11

			Laptop, Notebook, Copil, Man, Woman) • 30 = Resolution (Width x Height) (ex. 100 x 200, 640 x 480) • 40 = Volume (Width x Height x Depth - Depth 2) (ex. 30 x 40 x 50 - 10)		
	display_order		Characteristic display order	Integer	display_order=6
	is_mandatory		Indicates if the characteristic is mandatory when sending a product. Possible values are 0 (the characteristic is not mandatory) and 1 (the characteristic mandatory).	Integer	is_mandatory=0
	is_filter		Indicates if the characteristic represents a filter in the website. Possible values are 0 (the characteristic is not filter) and 1 (the characteristic is a filter).	Integer	is_filter=1
	allow_new_value		Indicates if the current characteristic allows you to submit new values that are automatically validated.	Integer	allow_new_value=0
	values		List the first 256 existing values of the current characteristic. Important: This key is available only when reading a single category.	Array	0 => 'Value 1' 1 => 'Value 2'
family_types			List of all family types available in category	List of arrays	
	id		Family type id	Integer	Id=95
	name		Family name	String	name="Quantity"
	characteristics		All characteristics of current family type	List of arrays	
		characteristic_id	Characteristic Id	Integer	characteristic_id=44
		characteristic_family_type_id	Can only have 3 values, each corresponding to a different display method: "1" ="Thumbnails"; "2"="Combobox"; "3"="Graphic Selection"	Integer	characteristic_family_type_id=2
		is_foldable	A foldable characteristic wraps all family members (with different characteristic values) as one item in the eMAG category listing	Integer	Is_foldable=1

		display_order	Characteristic display order	Integer	

By default responses will contain category names in the platform language but you can also send the language as a parameter, in case you want to receive category names in a specifc language. Available languages are: EN, RO, HU, BG, PL, GR and DE

E.g. https://marketplace-api.emag.ro/api-3/category/read?language=en

2.2. Reading VAT rates

When sending an offer, you have to send the VAT rate id by sending us a valid VAT id.
The resource is *vat* and the action is *read*. The API will return the list of available VAT rates and their corresponding id's.

2.3. Reading Handling Time values

When sending an offer, you have to send the Handling Time by sending us a valid value.
The resource is **handling_time** and the action is **read**. The API will return the list of available handling_time values.

2.4. Sending a new product

2.4.1. Draft product

Sending a draft product requires you to send a smaller set of data for publishing a new product. The information is saved in eMAG platform and the necessary details for publishing a product can be added at any time.

Please note:
Draft products won't be sent to eMAG Catalogue team for validation unless you send the other details necessary for publishing a new product (described below).

If an EAN published on a draft product is found in the eMAG catalogue you can skip the product publishing process and attach the offer directly to existing product.

Details for sending a new draft can be found here.

2.4.2. Product

Sending a product for the first time requires you to send the entire product documentation and all the offer data. Please note that creating new products implies human validation, so a new product will not be displayed in eMAG platform immediately.
The products that are not compliant with eMAG Documentation Standard will not pass the human validation; in this case you will be notified by our support team. The eMAG Documentation Standard that is available upon request for each category, and it contains the best practices for documenting a product.

In order to send a new product, the resource is product_offer and the available action is save.

PRODUCT OFFER – save and create/update product

Key – level 1	Key – level 2	Description	Constraints	Example
id		Seller internal product id. This is the primary key for identifying a product offer.	Required. Integer value between 1 and 16777215.	id=243409
category_id		Product category eMAG id.	Required. Integer between 1 and 65535.	category _id=506
vendor_category_id		Seller internal category id.	Integer. Optional.	vendor_category_id=506
part_number_key		eMAG part_number_key. Used for attaching a product offer to an existing product in eMAG platform. If you want to create new product, don't set this key.	Optional. String. Will be validated.	part_number_key=ES0NKBBBM
source_language		The language of the product content input. If it differs from the platform local language, then the product will enter a translation process. Available values for this key are: ro_RO, bg_BG, hu_HU, pl_PL, de_DE, it_IT, fr_FR, es_ES, nl_NL, zh_CN, cs_CZ, ru_RU.	Optional. String. Default value: on Marketplace RO: "ro_RO" on Marketplace BG: "bg_BG" on Marketplace HU: "hu_HU"	source_language="de_DE"
name		Product name. Should be consistent with eMAG Product Documentation Standard.	Required. String between 1 and 255 characters long.	name="Test product"
part_number		Manufacturer unique identifier of the product.	Required. String between 1 and 25 characters. Important: The following characters will be automatically stripped: • spaces [ ] • comma [,] • semicolon [;] Example: "part number;" will be saved "partnumber"	part_number="md788hc/a"
description		Product description. Should be consistent with eMAG Product Documentation Standard.	Optional. String between 1 and 16777215 characters. Can contain basic HTML tags.	description="test"
brand		Brand name. Should be consistent with eMAG Product Documentation Standard.	Required. String between 1 and 255 characters.	brand="Brand test"
weight		The weight of the product	Optional. Decimal value between 0 and 999999. Up to six decimals.	weight=12.123456

PRODUCT OFFER – save and create/update product				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
force_images_download		Image attachement redownload flag. Only to be used when updating product documentation 1-forces images redownload 0-images will not be redownloaded	Optional. Integer value, 1 or 0. Default = 0	force_images_download=1
images		Product images data array.	Optional. List of arrays.	
	display_type	Image display type. 1 – main image 2 – secondary image 0 – other images	Optional. Default value 0. Integer value between 0 and 2.	display_type=1
	url	Seller image URL. Should be consistent with eMAG Product Documentation Standard. Max 6000px x 6000px and 8 Mb in size.	Required. String between 1 and 1024 characters. Valid URL. JPG, JPEG or PNG file type.	url="http://valid-url.jpg"
characteristics		Characteristic data. Note that characteristics have to be category valid (be part of category template). Should be consistent with eMAG Product Documentation Standard.	Optional. List of arrays.	
	id	Characteristic eMAG id.	Required. Integer value between 1 and 65535	id=24
	value	Characteristic value. Should be consistent with eMAG Product Documentation Standard.	Required. String between 1 and 255 characters	value="test"
family		Family array. Used to create a new family, add a product to an existing family, or removing a product from a family.	Optional. Array.	
	id	The unique integer identifier of the family in your platform. If set to 0 (id=0), the product will be removed from its current family.	Required. Integer	Id=0
	name	Required. Seller Family name.	Required if family id is not equal to 0;	name="Test product"
	family_type_id	Required. eMAG Family type id that can be acquired by API (the resource is **category** and the action is **read**).	Required if family id is not equal to 0. Integer.	family_type_id=95

PRODUCT OFFER – save and create/update product

Key – level 1	Key – level 2	Description	Constraints	Example
url		Product URL on the seller website.	Optional. String between 1 and 1024 characters.	url="http://valid-url.html"
warranty		The warranty offered in months.	Required/Optional based on category. Default value: - 0 (no warranty) if optional - No default if required. Integer between 0 and 255.	warranty=24
ean		Product barcode identifier (EAN -8, EAN-13, UPC-A, UPC-E, JAN, ISBN-10, ISBN-13, ISSN, ISMN-10, ISMN-13, GTIN-14). Please use the supplier barcode, not your internal barcodes.	Required/Optional based on category. No default value. Array of strings between 6 and 14 characters long. Only numeric figures allowed.	ean=Array('ean1', 'ean2')
attachments		Product attachments data. Max 10 Mb in size.	Optional. List of arrays.	
	id	Seller attachment internal id.	Optional. Integer value between 1 and 4294967295.	id=123
	url	Seller attachment URL.	Required. String between 1 and 1024 characters. Valid URL to document.	url="http://valid-url"
status		Seller offer status. 1 – status active 0 – status inactive	Required. Integer value, 1 or 0.	status=1
sale_price		Seller offer sale price without VAT	Required. Decimal value greater than 0. Up to four decimals.	sale_price=51.6477
recommended_price		Seller offer recommended retail price before discount, without VAT. If set, the offer will be displayed as promo.	Optional. Decimal value greater than 0. Up to four decimals. Must be greater than sale_price.	recommended_price=51.6477
min_sale_price		Seller's min offer sale price without VAT	Required on first product save. Decimal value greater than 0. Up to four decimals.	min_sale_price=40.6477
max_sale_price		Seller's max offer sale price without VAT	Required on first product save. Decimal value greater than 0. Up to four decimals. Must be greater than min_sale_price.	max_sale_price=60.6477
currency_type		Offer currency. Only send the key if it is different from the	Optional. 3 characters string.	currency_type="EUR"

PRODUCT OFFER – save and create/update product				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
		local Marketplace currency. Available options: EUR or PLN		

stock		Offer available quantity array.	Required. List of arrays.	{
				0=>{
				warehouse_id=1,
				value=20}}

	warehouse_id	The id of the warehouse. Use warehouse_id=1 for only one warehouse.	Required inside stock array. Integer.	warehouse_id=1
	value	Offer available quantity.	Required inside stock array. Integer between 0 and 65535.	value=20

handling_time		Handling time array. If no array is sent, the products are shipped the same day they are received.	Optional. List of arrays.	{
				0=>{
				warehouse_id=1,
				value=1}}

	warehouse_id	The id of the warehouse. Use warehouse_id=1 for only one warehouse.	Required inside handling_time array. Integer.	warehouse_id=1
	value	Handling time, in number of days counted from the day the order was received. If handling_time = 0 the order will be shipped the same day it is received.	Required inside handling_time array. Integer value between 0 and 255. Default value = 0.	value=0
supply_lead_time		The number of days needed to restock the product, from order placement to the supplier or production order, until product reception in the warehouse. Available values for this key are: 2, 3, 5, 7, 14, 30, 60, 90, 120	Optional. Integer. Default value = 14	supply_lead_time=5
start_date		If it's a new offer, it represents the date your offer will be available from. For offer updates, it schedules value updates for the following data: • sale_price • recommended_price • stock • handling_time • vat_id • warranty • status All other data will be updated on the fly. Using start_date,	Optional. Text in YYYY-MM-DD format. Date can be as far as 60 days in the future (cannot be earlier than tomorrow). Cannot be null.	start_date="2014-12-31"

PRODUCT OFFER – save and create/update product				
Key – level 1	Key – level 2	Description	Constraints	Example
		for example, you can schedule the inactivation of an offer, a price update, etc.		
vat_id		Seller offer VAT rate id. Use /vat/read to display possible values.	Required. Integer.	Ex: vat _id=1

IMPORTANT:

- During campaings with stock in site we will not allow the following:
  - regular offer updates;
  - updates sent during campaign time that have a start_date in the future;
  - previously scheduled updates with start_date during campaign time.
- You can save an offer update with currency_type different from the local Marketplace currency and start_date before the end of the current month. Any update attempt with a start_date greater than the last day of the current month will be rejected.
- Be aware that prices published directly in the local Marketplace currencies will be overwritten by prices published in other currencies when these are recalculated and published automatically on FX change at the beginning of every month. The automated recalculation can be disabled on request.
- Min / Max sale price keys are used for price check purposes and are mandatory for all calls used to create product/offers for the first time. As a best practice we recommend sending these keys only when you want to change their values.
- Sale price will be validated against min_sale_price and max_sale_price. Any offer that is not within the specified range will be rejected.
- In order to change a previously sent product image or attachment the url should be different from the one already sent. We reload the images only if the URL differs.
- We recommend sending the product data only upon product create/update, as there is no need to resend product unless it changed. Also we recommend sending the offer data upon changing (no matter the frequency) and at least weekly (even if the offer is the same) rather using periodical sending (crons, agents). You should program marketing campaigns using "start_date" campaign. Also please offer the possibility for an offer to be attached to an existing eMAG product (using part_number_key).
- The product part number should be assigned to a single product.
  If a product part number is re-used (set on a second product) an error will be generated and the product will NOT be saved.
- Multiple EAN codes can be set on an offer, but one EAN product code can NOT be used on two or more products. If one EAN code is added on a second new product, an error will be generated and the product will NOT be saved. If one EAN code used on a new product is already linked to a product in the eMAG catalogue, the offer will be automatically associated to that existing product.
- In the event of an documentation error when saving a product, the API will return and "isError:true" message, but the new offer will be saved and processed.
- When adding a product to a family
  - The category id of a product and the category id of its family type (family_type_id) must be the same.

- o All characteristics that define a family must be present and must have a valid value
- o All characteristics that define a family must have a single value
- o If a family is not valid, you will receive a warning response, but the product will be saved/updated
- o When moving a product from one family to another you only have to send the product with its new family type, id and name and make sure you follow the same rules as above

2.5. Example for a new product

Resource	Example	Context
product_offer/save	new_product (4).txt	http method: POST

2.6. Updating existing offer

When updating an existing offer for a product, you should send only the offer, without the documentation. Mandatory when updating a product offer are the following keys:

- id
- status
- sale_price
- vat_id
- handling_time
- stock

Please note that although the API permits sending the entire documentation on each offer update (price change, out-of-stock change, etc.) we do not recommend or encourage such a practice.

If you need to deactivate a valid offer on the website, you should send the offer with the "status = 0".

2.7. Saving volume measurements on products

In order to save volume measurements on existing products, the resource is *measurements* and the available action is *save*. The measurement units for volume are millimeters and grams.

MEASUREMENTS – save			
Key – level 1	**Description**	**Constraints**	**Example**
id	Seller internal product id. This is the primary key for identifying a product offer.	Required. Integer value between 1 and 16777215.	id=243409
length	The length of the product in millimeters (mm)	Required. Decimal value between 0 and 999999. Up to two decimals.	length=100
width	The width of the product in millimeters (mm)	Required. Decimal value between 0 and 999999. Up to two decimals.	width=122.50

MEASUREMENTS – save			
Key – level 1	**Description**	**Constraints**	**Example**
height	The height of the product in millimeters (mm)	Required. Decimal value between 0 and 999999. Up to two decimals.	height=250
weight	The weight of the product in grams (g)	Required. Decimal value between 0 and 999999. Up to two decimals.	weight=1254.50

Resource	**Example**	**Context**
measurements/**save**	measurements.txt	http method: POST

2.8. Reading and counting products and offers

In order to check the existing products (offers) and their status, the resource is **product_offer** and the available action are **read** and **count**.

		PRODUCT_OFFER – read		
Key – level 1	**Key – level 2**	**Description**	**Type**	**Example**
part_number_key		eMAG part_number_key.	String	part_number_key=ES0 NKBBBM
number_of_offers		How many sellers have active offers on this product	Integer	number_of_offers=3
buy_button_rank		The rank of the offer in its race to win the <Add to cart> button	Integer	buy_button_rank=1
best_offer_sale_price		Best selling price available in eMAG for the same Product	Decimal	best_offer_sale_price =51.6477
best_offer_recommend ed_price		The corresponding recommended price for the offer holding the best selling price	Decimal	best_offer_recommen ded_price=54.6477
ownership		Indicates who has ownership on the product's documentation. Posible values: 1 – Eligible for content updates 2 – Not eligible for content updates* *For products with ownership = 2, any content updates will be rejected.	Integer	ownership=1
category_id		Product category eMAG id.	Integer	category _id=506
vendor_category_id		Seller internal category id.	Integer	vendor_category_id=5 06

		PRODUCT_OFFER – read		
Key – level 1	**Key – level 2**	**Description**	**Type**	**Example**
id		Seller internal product id. This is the primary key for identifying a product offer.	Integer	id=243409
brand		Product brand name.	String	brand="Brand test"
name		Product name.	String	name="Test product"
part_number		Product part number.	String	part_number="md788 hc/a"
sale_price		Seller offer sale price without VAT	Decimal	sale_price=51.6477
recommended_price		Seller offer recommended retail price before discount, without VAT.	Decimal	recommended_price= 54.6477
currency		Product price currency.	String	currency='RON'
description		Product description.	String	description="test"
url		Product URL on the seller website.	String	url="http://valid-url.html"
warranty		The warranty offered in months.	Integer	warranty=24
ean		Product barcode identifier (EAN -8, EAN-13, UPC-A, UPC-E, JAN, ISBN-10, ISBN-13, ISSN, ISMN-10, ISMN-13, GTIN-14).	Array	ean=Array('ean1', 'ean2')
general_stock		The sum of the stock on all seller warehouses. Is decremented and incremented when orders are processed.	Integer	general_stock=20
estimated_stock		This key takes into account the reserved stock on unacknowledged orders.	Integer	estimated_stock=20
weight		The weight of the product	Decimal	weight=12.123456
status		Seller offer status. 1 – status active 0 – status inactive	Integer	status=1
images			List of arrays	
	url	Seller image URL.	String	url="http://valid-url.jpg"
	display_type	Image display type. 1 – main image 2 – secondary image 0 – other images	Integer	display_type=1
characteristics		All characteristics available in category	List of arrays	
	id	Characteristic eMAG id	Integer	id=38

		PRODUCT_OFFER – read		
Key – level 1	**Key – level 2**	**Description**	**Type**	**Example**
	value	Characteristic value.	String	value="test"
vat_id		Seller offer VAT rate id.	Integer	vat _id=1
family		Product family.	Array	
	id	Family id.	Integer	id=295
	name	Family name.	String	name="Test family"
handling_time			List of arrays	
	warehouse_id	The id of the warehouse.	Integer	warehouse_id=1
	value	Handling time, in number of days counted from the day the order was received.	Integer	value=0
validation_status		Product validation status	List of arrays	
	value	Product validation status value	Integer	value=4
	Description	Product validation status description	String	Description="Rejected documentation"
offer_validation_status		Offer validation status	List of arrays	
	value	Offer validation status value	Integer	value=2
	Description	Offer validation status description	String	Description="Invalid price"

The following filters are available when counting and reading products and offers:

Key	Description	Constraints
id	Displays the details for the corresponding ext_id.	Optional. Integer value between 1 and 4294967295.
currentPage	Set current page displayed	Optional, integer. Ex: currentPage =3
itemsPerPage	Set number of items to be displayed in one page. Maximum is set to 100.	Optional, integer. itemsPerPage=50
status	Returns only the offers with this status.	Optional. Seller offer status. 1 – status active 0 – status inactive

Key	Description	Constraints
general_stock	Returns only offers with numerical general_stock that have a value between 0 and the input.	Optional general_stock = 3
estimated_stock	Returns only offers with numerical estimated_stock that have a value between 0 and the input.	Optional reserved_stock = 3
validation_status	Returns only the results with this validation status.	Optional. 1 = Awaiting MKTP validation 2 = Awaiting Brand validation 3 = Awaiting EAN validation 4 = Awaiting Documentation Validation 5 = Rejected Brand 6 = Rejected EAN 7 = Rejected Association 8 = Rejected Documentation 9 = Approved Documentation 10 = Blocked
offer_validation_status	Returns only the results with this validation status.	Optional. 1 = Saleable 2 = Invalid price

Reading products for which sellers do not have ownership over the documentation will not return the values sent by sellers but the values that are displayed in the website.

2.9. Product validation responses

After reading a product, all the elements previously sent are returned, along with the key *doc_errors*. The key is not null for products that were rejected due to improper documentation. Below the list of possible errors, when they occur and the possible actions you need to take.

Product
documetation error
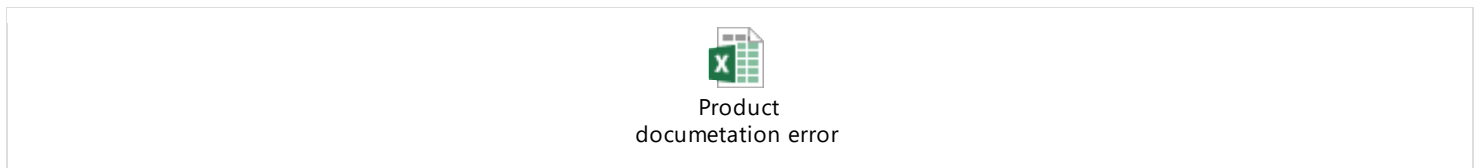
2.10. Matching products

In order to check if a product you sell already exists in eMAG catalog the resource is **product_offer** and the available action is **match**. The matching action is similar to the saving action so you should simply send the entire product documentation and all the offer data to the product matching resource. Please note that the matching action is available only for one product at a time therefore the request must be encapsulated in a single array.

As a response to the matching request, the following information will be returned:

Key – level 1	Description	Type	Example
part_number_key	eMAG part_number_key.	String	part_number_key=ES0NKBBBM
name	eMAG product name	String	name="Test product"
emag_product_url	eMAG product name	String	emag_product_url="http://www.emag.ro/product/pd/ES0NKBBBM/"

2.11. Attaching offers on existing products

You can chose between using PNK (part_number_key) or EAN for **attaching offers on existing eMAG products**

If the product already exists in eMAG catalog, just add the key "part_number_key" with product's part_number_key or the "ean" key with a single EAN.

IMPORTANT:

Only one offer can be attached to an existing product (identified by a "part_number_key") in eMAG catalogue.
In case you try to attach a second offer to a "part_number_key" that already has one of your offers attached, an error will be generated and the offer will NOT be saved.
If you already have an offer attached to a "part_number_key" please update it instead of trying to attach a new one.

> The part_number_key is the last key found in the URL of an eMAG product. It will ALWAYS have both numbers and characters. Ex: for product http://www.emag.ro/telefon-mobil-nokia-105-black-105-black/pd/D5DD9BBBM/ the part_number_key is D5DD9BBBM.

PRODUCT OFFER – save and create/update product				
Key – level 1	Key – level 2	Description	Constraints	Example
id		Seller internal product id. This is the primary key for identifying a product offer.	Required. Integer value between 1 and 16777215.	id=243409
name		Product name. Should be consistent with eMAG Product Documentation Standard.	Required. String between 1 and 255 characters long.	name="Test product"
ean		Product barcode identifier (EAN -8, EAN-13, UPC-A, UPC-E, JAN, ISBN-10, ISBN-13, ISSN, ISMN-10, ISMN-13, GTIN-14). Please use the supplier barcode, not your internal barcodes.	Required if part_number_key is not present. No default value. Array of strings between 6 and 14 characters long. Only numeric figures allowed. **OBS**: "part_number_key" and "ean" keys are mutually exclusive, you should use one or the other.	ean=Array('ean1')

PRODUCT OFFER – save and create/update product

Key – level 1	Key – level 2	Description	Constraints	Example
part_number_key		eMAG part_number_key. Used for attaching a product offer to an existing product in eMAG platform. If you want to create new product, don't set this key.	Required if ean is not present. String. Will be validated OBS: "part_number_key" and "ean" keys are mutually exclusive, you should use one or the other.	part_number_key=ES0NK BBBM
status		Seller offer status. 1 – status active 0 – status inactive	Required. Integer value, 1 or 0.	status=1
sale_price		Seller offer sale price without VAT	Required. Decimal value greater than 0. Up to four decimals.	sale_price=51.6477
recommended_price		Seller offer recommended retail price before discount, without VAT. If set, the offer will be displayed as promo.	Optional. Decimal value greater than 0. Up to four decimals. Must be greater than sale_price.	recommended_price=51.6 477
min_sale_price		Seller's min offer sale price without VAT	Required on first product save. Decimal value greater than 0. Up to four decimals.	min_sale_price=40.6477
max_sale_price		Seller's max offer sale price without VAT	Required on first product save. Decimal value greater than 0. Up to four decimals. Must be greater than min_sale_price.	max_sale_price=60.6477
currency_type		Offer currency. Only send the key if it is different from the local Marketplace currency. Available options: EUR or PLN	Optional. 3 characters string.	currency_type="EUR"
vat_id		Seller offer VAT rate id. Use /vat/read to display possible values.	Required. Integer.	Ex: vat _id=1
stock		Offer available quantity array.	Required. List of arrays.	{ 0=>{ warehouse_id=1, value=20}}
stock	warehouse_id	The id of the warehouse. Use warehouse_id=1 for only one warehouse.	Required inside stock array. Integer.	warehouse_id=1
stock	value	Offer available quantity.	Required inside stock array. Integer between 0 and 65535.	value=20
handling_time		Handling time array. If no array is sent, the products are shipped the same day they are received.	Optional. List of arrays.	{ 0=>{ warehouse_id=1, value=1}}

PRODUCT OFFER – save and create/update product				
Key – level 1	Key – level 2	Description	Constraints	Example
handling_time	warehouse_id	The id of the warehouse. Use warehouse_id=1 for only one warehouse.	Required inside handling_time array. Integer.	warehouse_id=1
handling_time	value	Handling time, in number of days counted from the day the order was received. If handling_time = 0 the order will be shipped the same day it is received.	Required inside handling_time array. Integer value between 0 and 255. Default value = 0.	value=0
start_date		If it's a new offer, it represents the date your offer will be available from. For offer updates, it schedules value updates for the following data: • sale_price • recommended_price • stock • handling_time • vat_id • warranty • status All other data will be updated on the fly. Using start_date, for example, you can schedule the inactivation of an offer, a price update, etc.	Optional. Text in YYYY-MM-DD format. Date can be as far as 60 days in the future (cannot be earlier than tomorrow). Cannot be null.	start_date="2014-12-31"
warranty		The warranty offered in months.	Required/Optional based on category. Default value: - 0 (no warranty) if optional - No default if required. Integer between 0 and 255.	warranty=24

Resource	Example	Context
product_offer/save	new_offer.txt	http method: POST

2.12. Reading commission for an offer

In order to read the estimated commission for an offer, a new REST API is available. Please consult the API Swagger.

3. Updating stock

In order to update only the stock of an offer a REST resource is available

Resource	Resource URL	Method	Authorization	Headers	Parameters
offer_stock	MARKETPLACE_API_URL/offer_stock/{resourceId}	PATCH	Basic (Base64)	Content-Type application/json	resourceId

The resourceId parameter represents the Seller internal product id. This is the primary key for identifying a product offer.

Resource	Example	Context
offer_stock	offer_stock.txt	http method: PATCH

4. Proposing offers in campaigns

To propose a valid offer in a campaign the resource is **campaign_proposals** and the available action is **save**.

Proposing offers in campaigns				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
id		Seller internal product id. This is the primary key for identifying a product offer.	Required. Integer value between 1 and 16777215.	id=243409
sale_price		Seller offer sale price without VAT available in the campaign	Required. Decimal value greater than 0. Up to four decimals.	sale_price=51.6477
original_sale_price		Seller offer recommended retail price before discount, without VAT available in the campaign. If set, the offer will be displayed as promo.	Optional. Decimal value greater than 0. Up to four decimals. Must be greater than sale_price.	original_sale_price =51.6477
stock		Available stock for the campaign. Once the stock has finished, the product can no longer be ordered.	Required. Integer value between 0 and 255.	stock=1
max_qty_per_order		The maximum quantity that a customer can order for a product from the campaign. This column is mandatory for stock in site campaigns.	Required/Optional depending on campaign type. List of arrays.	max_qty_per_order=4
post_campaign_sale_price		Product price after campaign end. The automatically filled price is the sale price of the product from the moment when offers are downloaded.	Optional. Decimal. Up to four decimals.	post_campaign_sale_price=55.6477

Proposing offers in campaigns				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
post_campaign_original_sale_price		Recommended retail price before discount of the product, available after campaign end.	Optional. Decimal. Up to four decimals.	post_campaign_original_sale_price=60.6477
campaign_id		eMAG internal campaign ID in which the proposal will be uploaded	Required. Integer value between 1 and 16777215.	campaign_id=344

Resource	Example	Context
***campaign_proposals*/save**	campaign_proposals.txt	http method: POST

5. Processing orders

An order consists of customer details, products and discounts from vouchers. It also has information about payment method, shipping tax. Also, each order always has a status attached. The available statuses are:

- 0 – canceled
- 1 – new
- 2 – in progress
- 3 – prepared
- 4 – finalized
- 5 – returned

The resource is ***order*** and the available actions are ***read***, ***save***, ***count*** and ***acknowledge***.

5.1. Order fields

An order has the following properties:

Key – level 1	Key – level 2	Description	Constraints	Example
id		The number that uniquely identifies an order.	Required. Integer value between 1 and 4294967295.	id=939393
status		The order processing status. Possible values: 0 - cancelled 1 - new 2 - in progress 3 – prepared 4 - finalized 5 - returned	Required. Integer value between 0 and 5.	status=1

is_complete		A flag indicating if the order is complete (has all details necessary for processing) or not. Possible values: 0 - incomplete; 1 - complete.	Optional. Integer value.	is_complete=1
type		A flag indicating if the order contains products fulfilled by eMAG or by seller. Possible values: 2 – fulfilled by eMAG 3 – fulfilled by seller	Optional. Integer value.	type=3
payment_mode_id		The order payment method. Possible values: 1 - COD (cash on delivery) 2 - bank transfer 3 - online card payment	Required. Integer.	payment_mode_id=1
detailed_payment_method		The detailed order payment method.	Optional. String.	detailed_payment_method = "eCREDIT"
delivery_mode		The order delivery method. Possible values: "courier" – home delivery "pickup" – locker delivery	Optional. String.	delivery_mode="courier"
details		Order extra details.	Optional. Array.	
details	locker_id	The pickup point id, if the locker delivery option was selected by a customer.	Optional. String.	locker_id="dce0b7cf-dc38-11e8-a7d8-001a4a160153"
	locker_name	The pickup point name, if the locker delivery option was selected by a customer.	Optional. String.	locker_name="easybox eMAG Showroom"
date		The cart submission timestamp.	Optional. Text in YYYY-mm-dd HH:ii:ss format.	date="1970-01-01 23:59:59"
payment_status		The online payment status. Only used for online payment methods. Possible values: 0 - not paid 1 - paid	Required only for online payment methods. Integer. It is highly recommended to also interpret the payment status when reading orders with Card Online payment method.	payment_status=0
cashed_co		The cashed amount from Card online payment	Optional. Integer.	
cashed_cod		The cashed amount from cash on delivery payment	Optional. Integer.	
shipping_tax		The shipment tax value.	Optional. Decimal.	shipping_tax="19.99"

shipping_tax_voucher_split		A list of arrays describing the voucher discounts distribution on shipping tax level.	List.	
	voucher_id	The ID of the voucher discount	Optional. Integer value between 1 and 9999999.	voucher_id=123
	value	The value of the discount, whitout VAT	Optional. Negative value.	value=-200
	vat_value	The value of the VAT	Optional. Negative value.	Vat_value=-38
customer		A list with the details about the customer, the shipping and the billing addresses.	Optional. List.	The field list is detailed below.
products		A list of arrays describing the products in the order.	List.	The field list is detailed below.
attachments		A list of arrays describing the attachments in the order.	List.	The field list is detailed below (chapter 3.1.3).
vouchers		A list describing the voucher discounts.	List.	
vouchers	voucher_id	The ID of the voucher discount	Optional. Integer value between 1 and 9999999.	voucher_id=123
	modified	The modified date of the voucher discount	Optional. Text in YYYY-mm-dd HH:ii:ss format.	modified="2015-04-23 11:30:09"
	created	The date of the voucher discount	Optional. Text in YYYY-mm-dd HH:ii:ss format.	created="2015-04-23 11:30:09"
	status	The status of the voucher discount	Optional. Integer.	status=1
	sale_price_vat	The value of the VAT	Optional. Negative value.	sale_price_vat="-1.9355"
	sale_price	The value of the discount, without VAT	Optional. Negative value.	sale_price="-8.0645"
	voucher_name	The name of the voucher	Optional. String.	voucher_name="eMAG giftcard"
	vat	The VAT rate	Optional. Decimal.	vat="0.24"
	issue_date	The date when the voucher was issued	Optional. Text in YYYY-mm-dd format.	issue_date="2020-06-09"
is_storno		Mandatory key when products are returned for a finalized order. Further details here.	Optional. Boolean. True indicates partial storno.	is_storno=true

| cancellation_reason | | The order cancellation reason. Possible values:

1 - Out of stock
2 - Cancelled by the client
3 - The client can not be contacted
15 - Courier delivery term is too large
16 - Transport tax, is too large
17 - Large delivery term, until the product will arrive in our warehouse
18 - Better offer in another store
19 - Payment order has not been paid
20 - Undelivered order, courier reasons
21 - Others
22 - Order Incomplete - automatic cancellation
23 - The customer changed his mind
24 - By customer request
25 - Failed delivery
26 - Late shipment
27 - Irrelevant Order
28 - Canceled by SuperAdmin on seller request
29 - Blacklisted customer
30 - No VAT invoice
31 - The eMAG Marketplace partner requested the order cancellation
32 - The delivery estimate is too long
33 - The product is no longer available in the stock of the eMAG Marketplace partner
34 - Other reasons
35 - The delivery is too expensive
36 - The customer found a better price elsewhere
37 - The customer registered another eMAG order with a similar product
38 - The customer changed his mind, does not need the product
39 - The customer can purchase the product only by installments | Optional. Integer value between 1 and 5. | cancellation_reason=1 |

5.1.1. Product field in order details

Key – level 1	Key – level 2	Description	Constraints	Example
id		eMAG internal order product line id. Any update on order product lines must use this id.	Required. Integer value between 1 and 9999999. id=243409	id=123
product_id		Seller internal product id. This is the primary key for identifying a product offer.	Optional. Integer.	product_id=3331

Key – level 1	Key – level 2	Description	Constraints	Example
product_voucher_split		A list of arrays describing the voucher discounts distribution on product level.	List.	
	voucher_id	The ID of the voucher discount	Optional. Integer value between 1 and 9999999.	voucher_id=123
	value	The value of the discount, whitout VAT	Optional. Negative value.	value=-200
	vat_value	The value of the VAT	Optional. Negative value.	Vat_value=-38
status		The status of product of the order. Possible values: 0 - cancelled 1 - active	Required. Integer.	status=1
part_number		Manufacturer unique identifier for the product.	Optional. String between 1 and 25 characters. Important: The following characters will be automatically stripped: • spaces [ ] • comma [,] • semicolon [;] Example: "part number;" will be saved "partnumber"	part_number='682133frs'
created		The date when the order product line was created.	Optional. Text in YYYY-mm-dd HH:ii:ss format.	created='2014-07-24 12:16:50'
modified		The date when the order product line was last modified.	Optional. Text in YYYY-mm-dd HH:ii:ss format.	modified='2014-07-24 12:18:53'
currency		Product price currency.	Optional. String.	currency='RON'
quantity		Product quantity.	Required. Integer. Positive, different than 0.	quantity=2
sale_price		The sale price without VAT.	Required. Integer.	sale_price=12.1234
details		Additional product notes.	Optional. Text.	details="text"
status		The status of product of the order. Possible values: 0 - cancelled 1 - active	Required. Integer.	status=1
sale_price		The sale price without VAT.	Required. Integer.	sale_price=12.1234
details		Additional product notes.	Optional. Text.	details="text"

IMPORTANT

Please note that multiple vouchers, possibly having different VAT rates, can be applied on a product. You should always read all voucher parameters.

5.1.2. Customer fields in order details

The customer field has the following properties:

Key	Description	Constraints	Example
id	The number that uniquely identifies a customer.	Optional. Integer value between 1 and2147483647.	id=1
name	The customer's name.	Optional. Text.	name="Surname Name"
email	This is a hash that uniquely identifies the customer's email.	Optional. Text.	email="1243536@emag.ro"
company	The name of the company. For physical person it has the same value as name.	Optional. Text.	company="Company name ltd."
gender	The customer gender. Possible values: M - male F – female	Optional. Text.	gender="M"
code	The company registration code.	Optional. Text.	code="14399840"
registration_number	The company registration number.	Optional. Text	registration_number=" 40/372/2002"
bank	The bank name.	Optional. Text.	bank="Bank name"
iban	The bank account.	Optional. Text.	iban=" RO24BACX0000000031430310"
fax	The customer's fax number.	Optional. Text.	fax="4021123123"
legal_entity	A flag indicating if the customer is physical or juridical entity. Possible values: 0 - private entity; 1 - legal entity.	Optional. Integer value.	legal_entity=1
is_vat_payer	A flag indicating it the customer is vat payer. Possible values: 0 - the customer is not vat payer; 1 - the customer is vat payer.	Optional. Integer value.	is_vat_payer=0
phone_1	The customer's first phone number.	Optional. Text.	phone_1="4021123123"
phone_2	The customer's second phone number.	Optional. Text.	
phone_3	The customer's third phone number.	Optional. Text.	
billing_name	The customer's invoice name.	Optional. Text.	billing_name="Surname Name"
billing_phone	The customer's invoice phone.	Optional. Text.	billing_phone="4021123123"
billing_country	The customer's invoice country.	Optional. Text.	billing_country="Romania"
billing_suburb	The customer's invoice county.	Optional. Text.	billing_suburb="Suburb"
billing_city	The customer's invoice city.	Optional. Text.	billing_city="City"

Key	Description	Constraints	Example
billing_street	The customer's invoice address.	Optional. Text.	billing_street="Street Name"
billing_postal_code	The customer's invoice postal code.	Optional. Text.	billing_postal_code="23125"
shipping_contact	The name of the contact person that will pick up the parcel. Should be printed on the AWB.	Optional. Text.	shipping_contact="Name Surname"
shipping_phone	The phone used by the courier to contact the shipping person. Should be printed on the AWB.	Optional. Text.	shipping_phone="23125"
shipping_country	The customer's shipping country.	Optional. Text.	shipping_country="Romania"
shipping_suburb	The customer's shipping county.	Optional. Text.	shipping_suburb="Suburb"
shipping_city	The customer's shipping city.	Optional. Text.	shipping_city="City name"
shipping_street	The customer's shipping suburb.	Optional. Text.	shipping_street="Street name"
shipping_postal_code	The customer's shipping postal code.	Optional. Text.	shipping_postal_code="23125"
billing_locality_id	This field uniquely identifies a locality in the eMAG database. It represents the billing locality.	Integer value between 1 and 4294967295.	billing_locality_id="23"
shipping_locality_id	This field uniquely identifies a locality in the eMAG database. It represents the shipping locality.	Integer value between 1 and 4294967295.	shipping_locality_id="23"

5.1.3. Order invoices

When pushing orders into finalized status, you should also send the invoice PDF file location for the specific order.

The resource is **order/attachments** and the available action is **save**.

The following keys should be sent in attachments array in order to display an invoice in the customer's order details: name, url, type.

An attachment has the following properties:

Key	Description	Constraints	Example
order_id	The number that uniquely identifies an order.	Required. Integer value between 1 and 4294967295.	id=939393
name	The name of the attachment displayed to the customer (in order history or in email)	Optional. String between 1 and 60 characters	name='Invoice title'
url	Attachment URL.	Required. String between 1 and 1024 characters. Valid URL to document.	url="http://valid-url/invoice.pdf"
type	The type of document attached to the order. Possible values are: • 1 - invoice • 3 - warranty • 4 - user manual	Integer. Optional. Default 1="Invoice". Only .pdf files are accepted	type=1

Key	Description	Constraints	Example
	• 8 - user guide • 10 - AWB • 11 - proforma		
force_download	Flag used in order to force attachment download restrictions. If value is 0 and the attachment URL has not changed, the attachment will not be downloaded again.	Integer. Optional. Default value 0. Possible values: 0,1.	force_download=0

5.2. Order notification, acknowledgment and order filters

When a new order is placed in eMAG Marketplace for the first time, it's status is 1 (new) and a GET request with the order id is automatically made to an URL you provide (call-back URL). Ex: http://valid_url/path?order_id=123

In the next step, you should read the order passing the id previously mentioned and after successfully saving the order in your database you should notify us by calling back the API using the route MARKETPLACE _URL/api-3/order/acknowledge/[orderId]. This stops the order notification system for the mentioned order. Unless acknowledged, we will notify the new orders for up to 48 hours.

IMPORTANT:
- Order acknowledge is the only method of marking the order status as "in progress".
- Clients may ask for an order to be canceled, this will be done by eMAG <u>only</u> if the order was not acknowledged by the seller, thus some of the orders may be read directly with status 0 (canceled).

5.3. Order status matrix

The following matrix defines the order flow in eMAG Marketplace:

	New status					
Actual status	1 - New	2 - In progress	3 – Prepared	4 - Finalized	0 - Canceled	5 - Returned
1 - New	No	Yes by ACK only	No	No	No	No
2 - In progress	No	Yes	Yes	Yes	Yes	No
3 - Prepared	No	No	Yes	Yes	Yes	No
4 - Finalized	No	No	Yes in 48h max	Yes	Yes in 48h max	Yes in RT* + 5 days max
0 - Canceled	No	Yes in 48h max	Yes in 48h max	Yes in 48h max	Yes	No
5 - Returned	No	No	No	No	No	No

RT = return time allowed to customers

IMPORTANT:
- We recommend setting up a periodical /order/read (cron, agent) that should identify orders that were not acknowledged. By default on /order/read we expose the last 100 orders, but you can request up to 1000 or use pagination. Do not forget to test the order status matrix against your internal order workflow. As a best practice you should either acknowledge the order prior the read or re-read the order after acknowledging it; an order can be modified by eMAG employees upon the client's request as long as it is not acknowledged;
- You can only edit the order (add/remove products modify quantity or price) when in status 2 (in progress) or 3 (prepared);

- Once an order is finalized, you can change its status back to status 3 (prepared) or 0 (canceled) only in the first 48 hours since finalization;
- Order status "finalized" will be set automatically when issuing the first AWB for that order. See chapter Saving AWBs;
- The order status "returned" is set automatically when all the products from the initial invoice are marked as returned. The change is permitted only within the maximum return timeframe allowed to the customer.

5.4. Order filters

You can read all your orders though the API, using filters. The following are available when counting orders:

Key	Description	Constraints
id	Only the order with this value.	Optional. Integer value between 1 and4294967295.
createdBefore	Only the orders created before the specified date. Can only be set if "createdAfter" is present. Maximum 1 month difference.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
createdAfter	Only the orders created after the specified date. Can only be set if "createdBefore" is present. Maximum 1 month difference.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
modifiedBefore	Only the orders modified before the specified date. Can only be set if "modifiedAfter" is present. Maximum 1 month difference.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
modifiedAfter	Only the orders after before the specified date. Can only be set if "modifiedBefore" is present. Maximum 1 month difference.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
status	Only the orders with the specified status. It is a single value or a list of values.	Optional. Integer or list.
payment_mode_ id	Only the orders with the specified payment method id. It is a single value or a list of values.	Optional. Integer or list.
is_complete	Only the orders with the specified completion status.	Optional. Order completion status. 1 – complete orders 0 – incomplete orders
type	Only the orders with the specified type	Optional. Default value = 3. 2 – fulfilled by eMAG 3 – fulfilled by seller

The following filters are available when reading orders:

Key	Description	Constraints
itemsPerPage	The maximum number of orders to return.	Optional. Integer value between 1 and 100.
currentPage	The page offset.	Optional. Integer value between 1 and 65535.

Key	Description	Constraints
id	Only the order with this value.	Optional. Integer value between 1 and 4294967295.
createdBefore	Only the orders created before the specified date. Can only be set if "createdAfter" is present. Maximum 1 month difference.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
createdAfter	Only the orders created after the specified date.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
modifiedBefore	Only the orders modified before the specified date. Can only be set if "modifiedAfter" is present. Maximum 1 month difference.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
modifiedAfter	Only the orders after the specified date.	Optional. Text in YYYY-mm-dd HH:ii:ss format.
status	Only the orders with the specified status. It is a single value or a list of values.	Optional. Integer or list.
payment_mode_id	Only the orders with the specified payment method id. It is a single value or a list of values.	Optional. Integer or list.
is_complete	Only the orders with the specified completion status.	Optional. Order completion status. 1 – complete orders 0 – incomplete orders
type	Only the orders with the specified type	Optional. Default value = 3. 2 – fulfilled by eMAG 3 – fulfilled by seller

5.5. Updating orders

You cannot create new orders through the API, you can only read and update them. When updating an order, the seller should send ALL the fields initially read.

IMPORTANT:
- Updating products by reducing their quantities for orders with Online Card payment method is no longer possible.
- Updating product prices is no longer available
- A canceled order can no longer be reactivated if more than 48 hours have passed since cancelation

Resource	Example	Context
order/save	order_update.txt	http method: POST

5.5.1. Removing products from an order

To remove a product from the order send the status=0 for the product or do not send it at all. Products can be removed from an order only while in status 2 or 3 (in progress or prepared) for orders with payment methods different than Online card. For returned products (the order is in status 4, finalized), please use the storno route.

IMPORTANT: Removing products for orders with Online Card payment method is no longer possible.

5.5.2. Adding products to an existing order

To add a new product to an existing order, add it to the order by sending the product id (mandatory), name, status and sale price.

IMPORTANT: virtual products such as internal discounts can be inserted in an order, even if they were not previously sent to eMAG. Adding these products to an order will not make them available for purchase in the eMAG Marketplace platform.

5.5.3. Returned products and storno route

A finalized order cannot be modified, it can be fully reversed by changing the order status from finalized (4) to returned (5) or have only some of the products reversed using a call with is_storno key true.
The following conditions must be met in order for a partial storno to occur:

- Order must be in status 4
- At least one product quantity was reduced

The following scenarios can be used as a guideline for returning products (partial storno) from a finalized order:

Current order status	Request	isError	Order read

status' => 4,
'products' =>
  array (
    0 =>
    array (
     'id' => 1,
     'product_id' => '1',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
  ),
	status' => 4,
	'products' =>
	  array (
	    0 =>
	    array (
	     'id' => 1,
	     'product_id' => '1',
	     'quantity' => 1,
	     'sale_price' => '123.4567',
	     'status' => 1,
	    ),
	    1 =>
	    array (
	     'id' => 2,
	     'product_id' => '2',
	     'quantity' => 2,
	     'sale_price' => '123.4567',
	     'status' => 1,
	    ),
	  ),
	'is_storno'=true
		FALSE	status' => 4,
			'products' =>
			  array (
			    0 =>
			    array (
			     'id' => 1,
			     'product_id' => '1',
			     'quantity' => 1,
			     'sale_price' => '123.4567',
			     'status' => 1,
			    ),
			    1 =>
			    array (
			     'id' => 2,
			     'product_id' => '2',
			     'quantity' => 2,
			     'sale_price' => '123.4567',
			     'status' => 1,
			    ),
			  ),

status' => 4,
'products' =>
  array (
    0 =>
    array (
     'id' => 1,
     'product_id' => '1',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
	status' => 4,
	'products' =>
	  array (
	    0 =>
	    array (
	     'id' => 1,
	     'product_id' => '1',
	     'quantity' => 2,
	     'sale_price' => '123.4567',
	     'status' => 1,
	    ),
	    1 =>
	    array (
	     'id' => 2,
		FALSE	status' => 4,
			'products' =>
			  array (
			    0 =>
			    array (
			     'id' => 1,
			     'product_id' => '1',
			     'quantity' => 0,
			     'sale_price' => '123.4567',
			     'status' => 1,
			    ),
			  ),

Current order status	Request	isError	Order read
'product_id' => '2', 'quantity' => 2, 'sale_price' => '123.4567', 'status' => 1,),),	'product_id' => '2', 'quantity' => 0, 'sale_price' => '123.4567', 'status' => 1,),), 'is_storno'=true		

status' => 4,
'products' =>
  array (
  0 =>
  array (
  'id' => 1,
  'product_id' => '1',
  'quantity' => 2,
  'sale_price' => '123.4567',
  'status' => 1,
  ),
  1 =>
  array (
  'id' => 2,
  'product_id' => '2',
  'quantity' => 2,
  'sale_price' => '123.4567',
  'status' => 1,
  ),
  ),
	status' => 4,
	'products' =>
	  array (
	  0 =>
	  array (
	  'id' => 1,
	  'product_id' => '1',
	  'quantity' => 2,
	  'sale_price' => '123.4567',
	  'status' => 1,
	  ),
	  1 =>
	  array (
	  'id' => 2,
	  'product_id' => '2',
	  'quantity' => 2,
	  'sale_price' => '123.4567',
	  'status' => 0,
	  ),
	  ),
	'is_storno'=true
		FALSE	status' => 4,
			'products' =>
			  array (
			  0 =>
			  array (
			  'id' => 1,
			  'product_id' => '1',
			  'quantity' => 0,
			  'sale_price' => '123.4567',
			  'status' => 1,
			  ),
			  ),

status' => 4,
'products' =>
  array (
  0 =>
  array (
  'id' => 1,
  'product_id' => '1',
  'quantity' => 2,
  'sale_price' => '123.4567',
  'status' => 1,
  ),
  1 =>
  array (
  'id' => 2,
  'product_id' => '2',
  'quantity' => 2,
  'sale_price' => '123.4567',
  'status' => 1,
  ),
  ),
	status' => 4,
	'products' =>
	  array (
	  0 =>
	  array (
	  'id' => 1,
	  'product_id' => '1',
	  'quantity' => 2,
	  'sale_price' => '123.4567',
	  'status' => 1,
	  ),
	  1 =>
	  array (
	  'id' => 2,
	  'product_id' => '2',
	  'quantity' => 1,
	  'sale_price' => '123.4567',
	  'status' => 1,
	  ),
	  ),
		TRUE	The request will be discarded, as you are trying to modify a finalized order without is_storno key.

status' => 4,
'products' =>
  array (
  0 =>
  array (
  'id' => 1,
  'product_id' => '1',
	status' => 4,
	'products' =>
	  array (
	  0 =>
	  array (
	  'id' => 1,
	  'product_id' => '1',
		TRUE	The request will be discarded, as you are sending is_storno key without any change to an order line

Current order status	Request	isError	Order read
| ```
    'quantity' => 2,
    'sale_price' => '123.4567',
    'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
   ),
``` | ```
    'quantity' => 2,
    'sale_price' => '123.4567',
    'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
   ),
  'is_storno'=true
``` | | |
| ```
status' => 3,
'products' =>
   array (
    0 =>
    array (
     'id' => 1,
     'product_id' => '1',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
   ),
``` | ```
status' => 3,
'products' =>
   array (
    0 =>
    array (
     'id' => 1,
     'product_id' => '1',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
   ),
  'is_storno'=true
``` | TRUE | The request will be discarded, as you are sending is_storno key for an order with a status different than 4 |
| ```
status' => 4,
'products' =>
   array (
    0 =>
    array (
     'id' => 1,
     'product_id' => '1',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
   ),
``` | ```
status' => 4,
'products' =>
   array (
    0 =>
    array (
     'id' => 1,
     'product_id' => '1',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    1 =>
    array (
     'id' => 2,
     'product_id' => '2',
     'quantity' => 2,
     'sale_price' => '123.4567',
     'status' => 1,
    ),
    2 =>
    array (
``` | TRUE | The request will be discarded, as you are trying to send a negative quantity for a product |

Current order status	Request	isError	Order read
	'id' => 2, 'product_id' => '2', 'quantity' => -1, 'sale_price' => '123.4567', 'status' => 1,),), 'is_storno'=true		

6. Shipping orders

For electronic deliveries and downloadable goods, please skip this section. Shipping an eMAG Marketplace order requires the seller to issue an AWB using eMAG Marketplace API.

The resource is **AWB** and the available actions are **read** and **save**

6.1. Saving AWB

To save an AWB just call the API with the following parameters:

Key	Description	Constraints
order_id	Identifies the order	Required. Integer value between 1 and 4294967295. Must be a valid order in the eMAG database, and must be owned by the seller.
rma_id	Identifies the return request	Optional. Integer value between 1 and 4294967295. Must be a valid return request in the eMAG database, and must be owned by the seller.
sender	*Array explained below	
receiver	*Array explained below	
locker_id	The pickup point id. Should be filled in with the pickup point id received on the order. If filled in, the courier will deliver the parcel in the designated locker.	Optional. String value between 3 and 255 characters
is_oversize	If set to 1, marks the delivery as containing oversized products	Required. Value can only be 0 or 1.
insured_value	The insured value	Optional. Double value between 0 and 999999999
weight	The weight of delivery	Optional. Double value between 0 and 99999
envelope_number	Number of envelopes to be delivered	Required. Integer value between 0 and 9999. If parcel_number is 0, this parameter cannot be 0
parcel_number	Number of parcels to be delivered	Required. Integer value between 0 and 999. If envelope_number is 0, this parameter cannot be 0

Key	Description	Constraints
observation	Observation text	Optional. String value between 0 and 255
cod	Cash on delivery	Required. Double value between 0 and 999999999
courier_account_id	Unique identifier for seller's courier account. If not provided, a default account will be used.	Optional. Integer.
pickup_and_return	If set to 1, sender expects something in return to this expedition (documents, buy-back products, etc).	Optional. Value can only be 0 or 1.
saturday_delivery	If set to 1, sender requests the package to be delivered on Saturday.	Optional. Value can only be 0 or 1.
sameday_delivery	If set to 1, sender requests the package to be delivered the same day.	Optional. Value can only be 0 or 1.

An AWB S/R (sender/receiver) has the following properties:

Key	Description	Constraints
name	S/R's name	Required. String value between 3 and 255
contact	Receiver's contact person name	Required. String value between 1 and 255
phone1	S/R first phone number	Required. String value between 8 and 11 digits (only '+' character is allowed at the beginning of the string)
phone2	S/R second phone number	Optional. String value between 8 and 11 digits (only '+' character is allowed at the beginning of the string)
legal_entity	If Receiver is legal entity (applicable only to receiver)	LEGAL_ENTITY_NO = 0 LEGAL_ENTITY_YES = 1
locality_id	S/R's locality_id	Required. Integer value between 1 and 4294967295. Must be a valid locality in the eMAG database.
street	S/R's street	Required. String value between 3 and 255
zipcode	S/R's zipcode	Optional. String value between 1 and 255

IMPORTANT:

- For orders with "pickup" as a delivery method if you do not change the locker id that is already included in the "shipping_street" field the AWB will be issued as a locker delivery using the proper courier account regardless of the actual courier account you specified when issuing the AWB.

6.2. Reading AWB

The following filters are available when reading AWBs:

Key	Description	Constraints
emag_id	The eMAG internal barcode id	Integer value between 1 and 4294967295.

Key	Description	Constraints
		Must be a valid AWB in eMAG database.
reservation_id	The eMAG internal AWB reservation id	Integer value between 1 and 4294967295. Must be a valid AWB in eMAG database.

An AWB has the following properties:

AWB – read				
Key – level 1	Key – level 2	Description	Constraints	Example
emag_id		The eMAG internal AWB id	Integer.	emag_id=243409
order_id		The id of the order on which the AWB was issued	Integer.	order_id=243409
rma_id		The id of the return request on which the AWB was issued	Integer.	rma_id=243409
weight		The weight of delivery	Integer.	weight=1
awb_type		The type of delivery. Possible values: 1 – delivery to customer 2 – pickup from customer	Integer.	awb_type=1
awb		The AWB	List of arrays.	
	emag_id	The eMAG internal AWB barcode id	Integer	emag_id=243409
	awb_number	The AWB number	String	awb_number = "2EMG00011012"
	awb_barcode	The AWB barcode	String	awb_barcode = "2EMG00011012001"
status		The status of the delivery	List of arrays.	
	code	The code status of the delivery	String.	code="DLV"
	name	The name of the status of the delivery	String.	name="Delivered"
	description	The description of the status of the delivery	String.	description="AWB delivered"
courier		The courier used for issuing the AWB	List of arrays.	
	courier_account_id	The eMAG internal courier account id used for issuing the AWB	Integer.	courier_account_id=5186
	courier_name	The eMAG internal courier name used for issuing the AWB	String.	courier_name="SAMEDAY"

6.3. Reading AWB PDF files

To download an AWB PDF file call the MARKETPLACE_API_URL/awb URL as in the example below

```
<html>
Running...<br>
<?
$username = 'user';
$password = 'pass';
$hash = base64_encode($username . ':' . $password);
$headers = array(
'Authorization: Basic ' . $hash
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,'https://marketplace-api.emag.ro/awb/read_pdf?emag_id=9755945&awb_format=A4';
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
$result = curl_exec($ch);
echo $result . "\n";
?>
</html>
```

Optional: You can set the paper format on PDF download link by using the parameter "awb_format=A4" in the link. The possible values are A4, A5, A6.

The following method will be released in the near future (H1 2021)

To download an AWB just call the API with the following parameters:

Key	Description	Constraints
emag_id	The AWB's eMAG id.	Integer value between 1 and 4294967295. Must be a valid AWB in eMAG database.
awb_format	The paper format on PDF download	The possible values are A4, A5, A6 and ZPL

6.4. Reading AWB ZPL type

To read an AWB in ZPL type call the MARKETPLACE_API_URL/awb URL as in the example below

```
<html>
Running...<br>
<?
$username = 'user';
$password = 'pass';
$hash = base64_encode($username . ':' . $password);
$headers = array(
'Authorization: Basic ' . $hash
);
$ch = curl_init();
```

```
curl_setopt($ch, CURLOPT_URL,'https://marketplace-api.emag.ro/awb/read_zpl?emag_id=9755945';
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
$result = curl_exec($ch);
echo $result . "\n";
?>
</html>
```

Using this request will return a base64 encoded content of the ZPL format as in the example below

Resource	Example	Context
awb/read_zpl	read_zpl response example.txt	http method: POST

6.5. Counting Localities

In order to issue an AWB you need to submit the correct locality id. You can also use the id from the order.

The resource is *locality* and the available actions are *read* and *count*.

The following filters are available when counting localities:

Key	Description	Constraints
emag_id	The locality with this id	Integer
name	All localities with this name	String of length between 0 and 60
modified	All localities modified after this date	Date with the 'Y-m-d H:i:s' format

6.6. Reading Localities

The following filters are available when reading localities:

Key	Description	Constraints
emag_id	The locality with this id	Integer
name	All localities with this name	String of length between 0 and 60
modified	All localities modified after this date	Date with the 'Y-m-d H:i:s' format
itemsPerPage	The maximum number of localities to return.	Optional. Integer value between 1 and 100.

Key	Description	Constraints
currentPage	The page offset.	Optional. Integer value between 1 and 65535.

A locality has the following properties:

Key	Description	Constraints
emag_id	The id of the locality	Integer
name	The name of the locality	String of length between 0 and 60
region([1-4]+)	Region name	String of length between 0 and 60
region([1-4]+)_latin	Region name latin version	String of length between 0 and 60
geoid	Geographic id of the location	Integer
modified	Last modification date	Date with the 'Y-m-d H:i:s' format

6.7. Reading courier accounts

In order to issue an AWB you need to submit the correct courier account id.

Key	Description	Constraints
account_id	The id of the account	Integer
account_display_name	The name of the account	String of length between 0 and 60
courier_account_type	The type of the account	Integer. Possible values: 1 - RMA; 2 - Order; 3 - RMA & Order; 4 - Non Marketplace
courier_name	The name of the courier	String of length between 0 and 60
courier_account_properties	The courier account properties	Array. Possible values: 1 – Regular; 4 – Locker delivery
created	The account creation date	Date with the 'Y-m-d H:i:s' format
status	Account status	Integer. Possible values: 1 - Active; 0 - Inactive

7. Processing return requests

A return request consists of customer details, products and reason for returning products. Each return request always has a status attached. The available statuses are:

      1 - incomplete

      2 - new

      3 - acknowledged

      4 - refused

5 - canceled

6 - received

7 - finalized

The resource is **RMA** and the available actions are ***read***, ***save***.

Return requests fields

The message structure for both read and save actions is detailed below:

RMA – read and save/update				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
emag_id		Return request eMAG system ID	Required. Type: bigint	
id		Seller internal return request ID	Type: bigint	
order_id		The id of the order in which the product to be returned was included	Required. Type: bigint	
type		A flag indicating if the return request contains products fulfilled by eMAG or by seller. Possible values: 2 – fulfilled by eMAG 3 – fulfilled by seller	Optional. Type: Integer.	type=3
invoice_number		Invoice of the order in which the product to be returned was included	Optional. Type: varchar Default value: NULL	
customer_name		Customer name	Required. Type: varchar	
customer_company		Customer company info	Optional. Type: varchar Default value NULL	
customer_phone		Customer phone no.	Required. Type: varchar	
products		Product to be returned info	Array	
	id	eMAG internal return product line id. Any update on return product lines must use this id.	Required. Integer value between 1 and 9999999.	id=123
	product_emag_id	Product to be returned eMAG ID		
	product_id	Product to be returned seller internal ID	Required. Type: int	
	quantity	Product quantity	Required Type: int	
	product_name	Product name	Required. Type: varchar	

RMA – read and save/update

Key – level 1	Key – level 2	Description	Constraints	Example
	return_reason	It holds the return reason selected by the customer. In the attached files you will find the possible IDs for the return reasons and their hierarchy. return_reasons_RO.xlsx return_reasons_BG.xlsx return_reasons_HU.xlsx <u>Note</u>: If you opt for a return reason that has other higher level reason(s), you will only need to fill in the last level ID from the hierarchy.	Required Type: int	
	observations	Free text notes field. <u>Note</u>: According to the attached files above, the observations key could be optional (value 0 or value 1) or mandatory (value 2).	Optional/ Required Type: text Default value: NULL	
	diagnostic	Diagnostic after product analysis	Optional. Type: int Default value NULL	
	reject_reason	Reason of return rejection	Optional. Type: int Default value NULL	
	refund_value	Refund value	Optional. Type: varchar	
awbs		Issued AWBs	List of arrays.	
	reservation_id	eMAG internal AWB reservation id. Use this id to read an AWB	Optional. Type: int	reservation_id= 4528511
pickup_country			Required. Type: Varchar	
pickup_suburb			Required. Type: Varchar	
pickup_city			Required. Type: Varchar	
pickup_address			Required. Type: Varchar	
pickup_address_id		Id of address already saved on the customers' account	Optional. Type: Int	

RMA – read and save/update				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
			Default value: NULL	
pickup_zipcode			Optional. Type: Varchar Default value: NULL	
pickup_date		Returned product pickup date (in case of vendor pickup from customer)	Optional. Type: datetime Default value NULL	
pickup_locality_id		The internal eMAG ID of the pickup address city/locality	Required. Type: int	
pickup_method		The product pickup method selected by the customer when inserting the return request. Possible values: 1 - eMAG courier 2 - Seller's own courier 3 - Sent by client	Required. Type: int	
return_reason		It holds the return reason selected by the customer. In the attached files you will find the possible IDs for the return reasons and their hierarchy. return_reasons_RO.xlsx return_reasons_BG.xlsx return_reasons_HU.xlsx Note: If you opt for a return reason that has other higher level reason(s), you will only need to fill in the last level ID from the hierarchy.	Required Type: int	
observations		Free text notes field. Note: According to the attached files above, the observations key could be optional (value 0 or value 1) or mandatory (value 2).	Optional/ Required Type: text Default value: NULL	
return_type		It holds the return type selected by the customer. Possible values: 1 - Replacement with same product 2 - Replacement with a different product 3 - Refund 4 - Cancel payment contract for this product 5 - Voucher	Required. Type: Int	

RMA – read and save/update				
Key – level 1	**Key – level 2**	**Description**	**Constraints**	**Example**
return_address_id		It will hold the id of the return address selected by the vendor in RMA UI	Optional. Type: Int Default value: id of the address set as default in vendor profile address page.	
return_tax_value		Shipping tax for refused returned products (VAT included) The currency used will be the platforms' default	Optional. Type: Float Default value: NULL	
customer_account_iban			Type: Varchar Default value: NULL	
customer_account_bank			Type: Varchar Default value: NULL	
customer_account_beneficiary			Type: Varchar Default value: NULL	
replacement_product_emag_id		The eMAG ID of the replacement product	Type: Int Default value: NULL	
replacement_product_id		The seller internal ID of the replacement product	Type: Int Default value: NULL	
replacement_product_name			Type: Varchar Default value: NULL	
replacement_product_quantity			Type: Int Default value: NULL	
date		RMA request insertion date	Required. Type: datetime	
request_status		RMA request status. Possible values: 1 - Incomplete 2 - New 3 - Approved 4 - Refused 5 - Canceled 6 - Received 7 - Finalized	Optional. Type: Int	

7.1. Return requests filters

When reading return requests the following filters are available:

Key	Description	Constraints
id	Seller internal return request ID	

Key	Description	Constraints
emag_id	eMAG return request ID	
order_id	Order on which the product to be returned was included	
product_id	Seller internal returned product ID	
product_emag_id	eMAG returned product ID	
requests_status	Return request status ID	
date	Return request insertion date	

7.2. Status change permissions

The following matrix defines the return request processing flow in eMAG Marketplace:

Actual status	New status					
	2 - New	3 - Acknowledged	4 - Rejected	5 - Canceled	6 - Received	7 - Finalized
2 - New	Yes	Yes	No	Yes	No	No
3 - Acknowledged	No	Yes	No	Yes	Yes	No
4 - Rejected	No	No	Yes	No	No	No
5 - Canceled	No	No	No	Yes	No	No
6 - Received	No	No	Yes	No	Yes	Yes
7 - Finalized	No	No	No	No	No	Yes

Some of the statuses were left out by design; these should not be used in any seller implementation

7.3. Return request deliveries

There are two types of possible deliveries for the return requests:

- pick-up requests - courier picks up the returned product(s) from the customer and delivers them to the seller
- regular deliveries - courier delivers the returned/replaced product back to the customer

The delivery requests will be generated using the *AWB save* resource.

7.4. Examples requests and responses

Resource	Example	Context
RMA/read	reading_rma.txt	http method: POST Seller has return requests
RMA/read	reading_rma_noreq uests.txt	http method: POST Seller does not have return requests

RMA/save	saving_rma.txt	http method: POST

8. Invoice API

The following documentation covers the available APIs used to:
- Read invoice categories
- Read invoice data
- Read customer invoice data

8.1. Reading invoice categories

Every MKTP invoice is included in a category. In order to read all invoice data of a specific MKTP invoice type, first call the following API without any parameter **api-3/invoice/categories**.

Resource	Example	Context
invoice/categories	invoice_categories.txt	http method: POST

As a result, we will return a collection of categories and the invoice name as follows:

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
category			Invoice type	Type: string	category='FC'
name			Invoice type name	Type: string	name='Commission'

Resource	Example	Context
invoice/categories	reading_invoice_categories.txt	http method: POST

8.2. Reading invoice data

URL: **api-3/invoice/read**

The resource is *invoice* and for the moment the only available action is *read*.
Reading invoice data without parameters, will generate a response containing the last 100 invoices.

You can read the invoice details using the following available filters:

Key	Description	Constraints
category	The invoice category from the results displayed when calling the api-3/invoice/categories API.	Optional category='FC'
number	The invoice series+number.	Optional number='C-MKTP-100001'
date_start	Only invoices created after date_start.	Optional Text in YYYY-mm-dd format
date_end	Only invoices created before date_end.	Optional Text in YYYY-mm-dd format
itemsPerPage	The maximum number of invoice data to return.	Optional. Integer value between 1 and 1000. Default value=100.
currentPage	The page offset.	Default value=1.

Reponse:

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
total_results			Number of invoices identified	Type:interger	total_results=1
invoices	category		Invoice type	Type: string	category='FC'
	name		Invoice name	Type: string	name='Commision'
	number		Invoice series+number	Type: string	number='C-MKTP-100001'
	date		The date when the invoice was created.	Text in YYYY-mm-dd	date= '2020-07-24'
	is_storno		The invoice represents a reversal of another invoice.	Type: integer	is_storno=1/is_storno=0
	supplier	name	Supplier name (legal name)	Type: string between 1 and 100 characters.	name='Dante International SA'
		register_number	Registration number	Type: string between 1 and 50 characters.	register_number='J40/372/2002'
		cif	Unique Identification Code	Type: string between 1 and 50 characters.	cif='14399840'
		tax_code	VAT Number	Type: string between 1 and 50 characters.	tax_code='RO14399840'
		social_capital	Subscribed and paid capital	Type: string between 1 and 50 characters.	social_capital='1.210.822 RON'

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
		iban	Bank account	Type: string between 1 and 100 characters.	iban='RO73INGB0001008199 078940'
		bank	Bank name	Type: string between 1 and 100 characters.	bank='ING BANK'
		address	Headquarters	Type: string between 1 and 255 characters.	address=' 148 Virtutii, E47, 060787, Sector 6, Bucuresti'
		phone_number	Phone number	Type: string between 1 and 50 characters.	phone=number='402120052 00'
	customer	name	Buyer name (legal name)	Type: string between 1 and 100 characters.	name='Test SRL'
		register_number	Registration number	Type: string between 1 and 50 characters.	register_number=' JXX/XXX/2002''
		cif	Unique Identification Code	Type: string between 1 and 50 characters.	cif='123456'
		tax_code	VAT Number	Type: string between 1 and 50 characters.	tax_code='RO123456'
		iban	Bank account	Type: string between 1 and 100 characters.	iban=' RO00INGB000000000007000 0'
		bank	Bank name	Type: string between 1 and 100 characters.	bank='ING'
		country	Country	Type: string between 1 and 100 characters.	country='Romania'
		address	Headquarters	Type: string between 1 and 255 characters.	address='Strada ABC, Bucuresti'
	lines	product_name	Products/Services Description	Type: text	product_name='Comision aferent desfasurator _dc_082015_1443024267_v 1, conform contract'
		unit_of_measure	Unit of measure	Type: string between 1 and 20 characters	unit_of_measure='Buc'
		quantity	Quantity	Type: double	quatity=1

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
		unit_price	Unit value	Type: double	unit_price=100
		vat_rate	VAT Rate	Type: smallint	vat_rate=19
		value	Product value (without VAT)	Type: integer	value=119
		vat_value	VAT value	Type: integer	vat_value=19
	payment_term		Invoice due date	Type: integer	payment_term=0
	total_without_vat		Total invoice without VAT	Type: integer	total_without_vat=100
	total_vat_value		Total invoice VAT value	Type: integer	total_vat_value=19
	total_with_vat		Total invoice with VAT	Type: integer	total_with_vat=119
	currency		Invoice currency	Type: string	currency='RON'

8.3. Reading customer invoice data

URL: **api-3/customer-invoice/read**

The resource is **customer-invoice** and for the moment the only available action is **read**.
Reading customer invoice data without parameters, will generate a response containing the last 100 invoices.

You can read the customer invoice details using the following available filters:

Key	Description	Constraints
category	The invoice category: normal or storno invoice.	Optional category='normal' / category='storno'
order_id	The order which was invoiced.	Optional order_id='148717039'
number	The invoice series+number.	Optional number='PRAF100010'
date_start	Only invoices created after date_start.	Optional Text in YYYY-mm-dd format
date_end	Only invoices created before date_end.	Optional Text in YYYY-mm-dd format
itemsPerPage	The maximum number of invoice data to return.	Optional. Integer value between 1 and 1000. Default value=100.
currentPage	The page offset.	Default value=1.

Reponse:

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
total_results			Number of invoices identified	Type:interger	total_results=1
invoices	category		Invoice type	Type: string	category='storno'
	order_id		The order which was invoiced.	Type: integer	order_id='148717039'
	number		Invoice series+number	Type: string	number='PRAF101092'
	date		The date when the invoice was created.	Text in YYYY-mm-dd	date= '2020-11-11'
	is_storno		The invoice represents a reversal of another invoice.	Type: integer	is_storno=1/is_storno=0
	reversal_for		The invoice which was canceled through this invoice.	Type:string	reversal_for='PRAF101030'
	supplier	name	Supplier name (legal name)	Type: string between 1 and 100 characters.	name=Dobre BA Shop SRL'
		register_number	Registration number	Type: string between 1 and 50 characters.	register_number='J40/900/2000'
		cif	Unique Identification Code	Type: string between 1 and 50 characters.	cif='10874881'
		tax_code	VAT Number	Type: string between 1 and 50 characters.	tax_code='RO10874881'
		social_capital	Subscribed and paid capital	Type: string between 1 and 50 characters.	social_capital='200 RON'
		iban	Bank account	Type: string between 1 and 100 characters.	iban='RO02BTRLRONCRT00W6717503'
		bank	Bank name	Type: string between 1 and 100 characters.	bank='BANCA TRANSILVANIA S.A.'
		country	Country	Type: string between 1 and 100 characters.	country='Romania'

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
		address	Headquarters	Type: string between 1 and 255 characters.	address='Colentina 32, bl 2 sc 4 et 33 Bucuresti'
	customer	name	Buyer name (legal name)	Type: string between 1 and 100 characters.	name='Test Test'
		register_number	Registration number	Type: string between 1 and 50 characters.	register_number=' '
		cif	Unique Identification Code	Type: string between 1 and 50 characters.	cif=''
		tax_code	VAT Number	Type: string between 1 and 50 characters.	tax_code=''
		iban	Bank account	Type: string between 1 and 100 characters.	iban=''
		bank	Bank name	Type: string between 1 and 100 characters.	bank=''
		country	Country	Type: string between 1 and 100 characters.	country='RO'
		address	Headquarters	Type: string between 1 and 255 characters.	address='Strada ABC, Bucuresti'
	lines	product_name	Products/Services Description	Type: text	product_name='Usa de intrare, termopan , model Roxy white 110x210[61-110x210] '
		unit_of_measure	Unit of measure	Type: string between 1 and 20 characters	unit_of_measure='Buc'
		quantity	Quantity	Type: double	quatity=-1
		unit_price	Unit value	Type: double	unit_price=1008.4
		vat_rate	VAT Rate	Type: smallint	vat_rate=19
		value	Product value (without VAT)	Type: integer	value=-1008.4
		vat_value	VAT value	Type: integer	vat_value=-191.6
	total_without_vat		Total invoice without VAT	Type: integer	total_without_vat=-1008.4

Key – level 1	Key – level 2	Key – level 3	Description	Constraints	Example
	total_vat_value		Total invoice VAT value	Type: integer	total_vat_value=-191.6
	total_with_vat		Total invoice with VAT	Type: integer	total_with_vat=-1200
	currency		Invoice currency	Type: string	currency='RON'