**eMAG Marketplace**

**API Implementation Best Practices v1.0**

**17.03.2015**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date modified** | **Changes** |
| 1.0 | 12.02.2015 | First draft |

Table of Contents

[1. General guidelines for authentication 2](#_Toc414361059)

[2. Maximum size for a request 3](#_Toc414361060)

[3. Publishing a product documentation 3](#_Toc414361061)

[4. Updating product offers 4](#_Toc414361062)

[5. Publishing a product offer attached to a pre-existing product 5](#_Toc414361063)

[6. Order read 6](#_Toc414361064)

[7. Critical errors 6](#_Toc414361065)

# General guidelines for authentication

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.

Table 1 - Change username, usercode, password and Marketplace URL

|  |
| --- |
| <html>  Running...<br>  <?  $data =  array (  'currentPage' => 1,  'itemsPerPage' => 10  );  $hash = sha1(http\_build\_query($data) . sha1(**'password'**));  $requestData = array(  'code' => ‘**usercode'**,  'username' => **'username'**,  'data' => $data,  'hash' => $hash);  $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL, **'https://mktp-stage.emag.ro/api-3/vat/read'**);  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($ch, CURLOPT\_CUSTOMREQUEST, 'POST');  curl\_setopt($ch, CURLOPT\_POSTFIELDS, http\_build\_query($requestData));  $result = curl\_exec($ch);  print($result);  ?>  </html> |

# Maximum size for a request

Requests are capped at a maximum size of 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”.

# Publishing a product documentation

We recommend as a best practice sending one product in each product\_offer call and multi-threading requests rather than sending multiple at once.

Also, we recommend sending the product data only upon product create/update, as there is no need to resend product unless it changed.

Table 2 - Array for publishing a new product

|  |
| --- |
| Array(  Array(  "id" => "6050",  "family" => Array(  "id" => 111,  "family\_type\_id" => 97,  "name" => "test\_family "  ),  "category\_id" => "1315",  "part\_number" => "test-part-number",  "name" => "Test name",  "description" => "Test description",  "brand" => "Test brand name",  "images" => Array(  Array(  "display\_type" => "1",  "url" => "http://www.image-url.test"  )  ),  "url" => "http://www.product-url.test",  "status" => "1",  "sale\_price" => "406.45",  "availability" => Array(  Array(  "warehouse\_id" => "1",  "id" => "3"  )  ),  "handling\_time" => Array(  Array(  "warehouse\_id" => "1",  "value" => "2"  )  ),  "stock" => Array(  Array(  "warehouse\_id" => "1",  "value" => "2"  )  ),  "commission" => Array(  "type" => "percentage",  "value" => "8"  ),  "vat\_id" => "1",  "characteristics" => Array(  Array(  "id" => "5213",  "value" => "Characteristic 5213 value"  ),  Array(  "id" => "1339",  "value" => "Characteristic 1339 1st value"  ),  Array(  "id" => "1339",  "value" => " Characteristic 1339 2nd value"  )  )  )  ) |

# Updating product offers

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.

|  |
| --- |
| Array(  Array(  "id" => "6050",  "category\_id" => "1315",  "status" => "1",  "sale\_price" => "406.4575",  "availability" => Array(  Array(  "warehouse\_id" => "1",  "id" => "3"  )  ),  "handling\_time" => Array(  Array(  "warehouse\_id" => "1",  "value" => "2"  )  ),  "stock" => Array(  Array(  "warehouse\_id" => "1",  "value" => "2"  )  ),  "commission" => Array(  "type" => "percentage",  "value" => "8"  ),  "vat\_id" => "1",  )  ) |

# Publishing a product offer attached to a pre-existing product

Please offer the possibility for an offer to be attached to an existing eMAG product (using part\_number\_key).

|  |
| --- |
| Array(  Array(  "id" => "6050",  "family" => Array(  "id" => 111,  "family\_type\_id" => 97,  "name" => "test\_family "  ),  "part\_number" => "test-part-number",  "name" => "Test name",  "description" => "Test description",  "brand" => "Test brand name",  "images" => Array(  Array(  "display\_type" => "1",  "url" => "http://www.image-url.test"  )  ),  "url" => "http://www.product-url.test",  "pnk" => "DM51RBBBM",  "status" => "1",  "sale\_price" => "406.4515",  "availability" => Array(  Array(  "warehouse\_id" => "1",  "id" => "3"  )  ),  "handling\_time" => Array(  Array(  "warehouse\_id" => "1",  "value" => "2"  )  ),  "stock" => Array(  Array(  "warehouse\_id" => "1",  "value" => "2"  )  ),  "commission" => Array(  "type" => "percentage",  "value" => "8"  ),  "vat\_id" => "1",  )  ) |

# Order read

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.

# Critical errors

When an API call fails to update an **offer** or **order**, besides logging the response, you should also send an email alert to the store owner as these errors are critical and may have financial consequences.