Documentation of Advanced Shipping extension for Magento 2 by Owebia

Introduction

Advanced Shipping is an extension for the e-commerce solution Magento (http://www.magentocommerce.com/).

The syntax using PHP 5.6 (http://php.net/manual/en/) allows a great flexibility in setting delivery charges.

Technical Informations

The usage of the PHP syntax (with an Abstract Syntax Tree (http://en.wikipedia.org/wiki/Abstract_syntax_tree)) has been preferred to the usage of a JSON syntax (with regular expressions and the eval function) as in previous versions of the extension. This is the result of considerations on security and performance.

The PHP code defined in the configuration is not evaluated with the eval function for security reasons. The library PHP-Parser (https://github.com/nikic/PHP-Parser) is used to obtain an Abstract Syntax Tree (AST). The AST is browsed and only a limited set of functions and variables is allowed.

As the AST is not version-dependent, you can use the PHP 5.6 syntax even if your server uses a lower version of PHP.

Add-ons

You want to do more with **Advanced Shipping**?

Discover our add-ons on Owebia Store (https://en.store.owebia.com/).

Unlimited Carriers Add-on (https://en.store.owebia.com/magento2-module-adv-ship-unlimited-carriers-addon.html)

Create as many carrier as you need with Advanced Shipping

Functions Add-on (https://en.store.owebia.com/magento2-module-adv-ship-functions-addon.html)

Use functions to simplify the writing of your configuration
Look at **"With Functions Add-on"** tabs in examples to see how easy it becomes with this add-on

Universal Setting Add-on (https://en.store.owebia.com/magento2-module-adv-ship-universal-setting-addon.html)

Hide or change price of methods regardless the shipping module

What's new?

- Get Functions Add-On (https://en.store.owebia.com/magento2-module-adv-ship-functions-addon.html) and make configuration creation easy with additional functions

Quick start

A call to `addMethod` function adds a shipping method.

```
object addMethod ( string $method_id , array $method_properties )
```

List of method **properties**:

- `title`
- `price`
- `description` (optional)
- `enabled` (optional): conditions for displaying the shipping method

EXAMPLE

```
// First shipping method
addMethod('method1', [
    'title' => "Standard Shipping",
    'price' => 10,
]);
// Second shipping method
addMethod('method2', [
    'title' => "Express Shipping",
    'price' => 20,
]);
```

More examples

> **Warning:** to avoid conflicts, use only the characters `a-z` , `0-9` and `_` for the method id.
> You should also avoid identifiers that already correspond to variable names (`quote` , `request` ...).

Use Variables and Functions

You can use variables, functions and operators (http://php.net/manual/en/language.operators.php) to define a dynamic value (all operators are not supported).

EXAMPLE

```
addMethod('id_001', [
    // Dynamic title using the variable $request->dest_country_id
    'title' => "Delivery to " . $request->dest_country_id,
    // Dynamic price using the variable $quote->subtotal
    'price' => 0.2 * $quote->subtotal,
]);
addMethod('id_002', [
    'title'   => "Free Shipping",
    // Dynamic enabled using the function substr() and the variable $quote->coupon_code
    'enabled' => substr($quote->coupon_code, 0, 4) == 'free',
    'price'   => 0,
]);
```

1. Variables

There are global variables and local variables that you can access with array functions and callbacks.

> If you activate the debug option, you can see all data available for each object that you use.
> Remember that you don't have access to all methods of the original objects because they are not accessed directly.

1.1. Global variables

These variables are global variables and can be accessed everywhere (use the **`global`** keyword when you are inside a function).

- The **request** object (`\Magento\Quote\Model\Quote\Address\RateRequest`) given by Magento:
  - `$request->all_items` : array of items in cart
  - The source address:
    - `$request->country_id` *
    - `$request->region_id`
    - `$request->city`
    - `$request->postcode`
  - The destination address:
    - `$request->dest_country_id` *
    - `$request->dest_region_id`
    - `$request->dest_region_code`
    - `$request->dest_street`
    - `$request->dest_city`
    - `$request->dest_postcode`
  - The package:
    - `$request->package_value` : value of the package
    - `$request->package_value_with_discount` : value of the package with discount
    - `$request->package_weight` : weight of the package
    - `$request->package_qty` : number of products in the package
  - Other attributes:
    - `$request->free_shipping` : free shipping calculated by Magento rules
    - `$request->*` : property of the request object
- The **quote** (`\Magento\Quote\Model\Quote`):
  - `$quote->subtotal` : subtotal (excluding tax)

- `$quote->subtotal_with_discount` : subtotal (excluding tax) with discount
- `$quote->grand_total` : total (including tax) with discount
- `$quote->base_subtotal` : subtotal (excluding tax) in base currency
- `$quote->base_subtotal_with_discount` : subtotal (excluding tax) with discount in base currency
- `$quote->base_grand_total` : total (including tax) with discount in base currency
- `$quote->coupon_code` : the coupon code used in cart
- `$quote->*` : property of the quote (ex: `$quote->subtotal`)

- The **customer group** (`\Magento\Customer\Model\Group`):
  - `$customer_group->id` : alias of `$customer_group->customer_group_id`
  - `$customer_group->code` : alias of `$customer_group->customer_group_code`
  - `$customer_group->name` : alias of `$customer_group->customer_group_code`
  - `$customer_group->*` : property of the customer group (ex: `$customer_group->tax_class_id`)
- The **customer** (`\Magento\Customer\Model\Customer`):
  - `$customer->id` : alias of `$customer->entity_id`
  - `$customer->*` : attribute of the customer (ex: email, lastname, firstname, group_id…)
- Custom **variables** (`\Magento\Variable\Model\Variable`):
  - `$variable->*` : custom variable defined in Magento (ex: `$variable->my_var`)
- The **store** (`\Magento\Store\Model\Store`):
  - `$store->id` `$store->code` `$store->name` `$store->address` `$store->phone`

* Magento apparently uses the ISO 3166-1 alpha-2 codes (http://fr.wikipedia.org/wiki/ISO_3166-1).
Note that the country code for United Kingdom is GB and not UK.

1.2. Accessing items

You can use `$request->all_items` with array functions.

EXAMPLE **WITH FUNCTIONS ADD-ON**

```
addMethod('id_003', [
    'title'      => "If at least one product has the attribute 'name' equal to 'Cat'",
    'enabled'    => count(
                        array_filter($request->all_items, function ($item) {
                            return $item->product->name == 'Cat';
                        })
                    ) > 0,
    'price'      => 10
]);
```

- The **item** (`\Magento\Quote\Model\Quote\Item`):
  - `$item->qty` : quantity of the item
  - `$item->weight` : weight of the item
  - `$item->base_original_price` : price excluding tax without discount
  - `$item->price_incl_tax` : price including tax without discount
  - `$item->options->*` : option (the available options depend on the product)
- The **product** (`\Magento\Catalog\Model\Product`):
  - `$item->product->*`
    - `sku`
    - `name`
    - `weight`
    - `price` (as defined in Magento backoffice)
    - `special_price` : (as defined in Magento backoffice)
    - `tax_class_id` : the taxclass id
    - `getAttributeText('tax_class_id')` : the tax class name
      To get the text value of an attribute of type select, use `getAttributeText('attribute_code')`
    - …
  - Categories of the product: (`\Magento\Catalog\Model\Category`)
    - `$item->product->category->*` : attribute of the first category
      - `id`
      - `name`
      - `is_active`
      - …
  - All product categories (returns an array, examples):
    - `$item->product->category_ids` : array of id of the categories
  - The attribute set (`\Magento\Eav\Model\Entity\Attribute\Set`):
    - `$item->product->attribute_set` : the attribute set
    - `$item->product->attribute_set->*` : attribute of the attribute set
      - `id`
      - `attribute_set_name`

        - ...
    - The stock item (`\Magento\CatalogInventory\Model\Stock\Item`):
        - `$item->product->stock_item->*` : attribute of the product stock
            - `is_in_stock`
            - `qty`
            - ...

2. Functions

Look at the PHP documentation if you want to know how to use these functions.

Functions allowed:

- abs (http://php.net/manual/en/function.abs.php)
- array_filter (http://php.net/manual/en/function.array-filter.php)
- array_intersect (http://php.net/manual/en/function.array-intersect.php)
- array_map (http://php.net/manual/en/function.array-map.php)
- array_reduce (http://php.net/manual/en/function.array-reduce.php)
- array_sum (http://php.net/manual/en/function.array-sum.php)
- array_unique (http://php.net/manual/en/function.array-unique.php)
- ceil (http://php.net/manual/en/function.ceil.php)
- count (http://php.net/manual/en/function.count.php)
- date (http://php.net/manual/en/function.preg-match.php)
- explode (http://php.net/manual/en/function.explode.php)
- floor (http://php.net/manual/en/function.floor.php)
- implode (http://php.net/manual/en/function.implode.php)
- in_array (http://php.net/manual/en/function.in-array.php)
- max (http://php.net/manual/en/function.max.php)
- min (http://php.net/manual/en/function.min.php)
- preg_match (http://php.net/manual/en/function.preg-match.php)
- range (http://php.net/manual/en/function.range.php)
- round (http://php.net/manual/en/function.round.php)
- strtotime (http://php.net/manual/en/function.strtotime.php)
- substr (http://php.net/manual/en/function.substr.php)

2.1. Functions available with Functions Add-On

- **_country**: see examples
- **_postcode**: see examples
- **_customerGroup**: see examples
- **_sum**: see examples
- **_min**: see examples
- **_max**: see examples
- **_count**: see examples
- **_anyCat**: see examples
- **_allCat**: see examples

Examples

Free shipping

EXAMPLE | WITH FUNCTIONS ADD-ON

```
addMethod('id_004', [
    'title'      => "Free shipping",
    'price'      => 0,
]);
```

Filter on destination

EXAMPLE | WITH FUNCTIONS ADD-ON

```
addMethod('id_005', [
    'title'      => "France, Germany, Switzerland, Spain, Italy",
    'enabled'    => in_array($request->dest_country_id, ['FR', 'DE', 'CH', 'ES', 'IT']),
    'price'      => 10,
]);
addMethod('id_006', [
    'title'      => "Postcode starting with 25",
    'enabled'    => $request->dest_country_id == 'FR' && substr($request->dest_postcode, 0, 2) == '25',
    'price'      => 10,
]);
addMethod('id_007', [
    'title'      => "Regular expressions allowing postal codes beginning with 'PO' (case insensitive)",
    'enabled'    => $request->dest_country_id == 'GB' && preg_match('/^PO.*$/i', $request->dest_postcode),
    'price'      => 10,
]);
```

France excluding DOM/TOM

EXAMPLE | WITH FUNCTIONS ADD-ON

```
// Regular Expression refusing delivery to France with postal codes
// beginning with 97 and 98 (with or without interspersed zeros and spaces)
addMethod('id_008', [
    'title'      => "France excluding DOM/TOM",
    'enabled'    => $request->dest_country_id == 'FR'
                        && !preg_match('/^[0\\s]*9\\s*[78]/', $request->dest_postcode),
    'price'      => 10,
]);
```

Filter on customer group

You can use the name or ID of the customer groups.

EXAMPLE | WITH FUNCTIONS ADD-ON

```
addMethod('id_009', [
    'title'      => "Retailer group",
    'enabled'    => $customer_group->code == "Retailer",
    'price'      => 10,
]);
addMethod('id_010', [
    'title'      => "NOT LOGGED IN or General groups",
    'enabled'    => in_array($customer_group->code, ['NOT LOGGED IN', 'General']),
    'price'      => 10,
]);
addMethod('id_011', [
    'title'      => "NOT LOGGED IN or General groups by their ID",
    'enabled'    => in_array($customer_group->id, [0, 1]),
    'price'      => 10,
]);
```

Using formulas

Formulas can be used to calculate the price of the delivery.

EXAMPLE

```
addMethod('id_012', [
    'title'      => "Shipping",
    'price'      => 0.1 * $quote->subtotal + 10.00,
]);
```

Using the configuration of another element

EXAMPLE

```
$standard = addMethod('standard', [
    'title'      => "Standard Shipping",
    'enabled'    => $quote->subtotal < 1000.00,
    'price'      => 10,
]);
addMethod('express', [
    'title'      => "Express Shipping",
    'enabled'    => $standard->enabled && $request->package_weight < 10,
    'price'      => 12,
]);
```

Using special functions

Defining a table

EXAMPLE | WITH FUNCTIONS ADD-ON

```
// If the package weight is lower or equal to 0.5, the price will be 5.30
// If the package weight is lower or equal to 1.0, the price will be 6.50
// In others cases, the price will be 7.50
addMethod('id_013', [
    'title'      => "Shipping",
    'price'      => array_reduce([ [0.7, 5.30], [1.0, 6.50], ['*', 7.50] ], function ($carry, $item) {
                        global $request;
                        if (isset($carry)) return $carry;
                        if (isset($item[0]) && ($request->package_weight <= $item[0] || $item[0] == '*')) {
                            $carry = $item[1];
                        }
                        return $carry;
                    }),
]);
// You can exclude the upper limit value by adding an argument false
addMethod('id_014', [
    'title'      => "Shipping",
    'price'      => array_reduce([ [0.7, 5.30], [1.0, 6.50], ['*', 7.50] ], function ($carry, $item) {
                        global $request;
                        if (isset($carry)) return $carry;
                        if (isset($item[0]) && ($request->package_weight < $item[0] || $item[0] == '*')) {
                            $carry = $item[1];
                        }
                        return $carry;
                    }),
]);
```

Using an associative table

EXAMPLE

```
// If the coupon code is equal to "coupon1", the price will be 5.30
// If the coupon code is equal to "coupon2", the price will be 6.50
// In others cases, the price will be 7.50
addMethod('id_015', [
    'title'      => "Shipping",
    'price'      => [ 'coupon1' => 5.30, 'coupon2' => 6.50 ][$quote->coupon_code] ?? 7.50,
]);
```

Using the attributes and options of products

Count items

EXAMPLE WITH FUNCTIONS ADD-ON

```
addMethod('id_016', [
    'title'      => "If at least one product has the attribute 'name' equal to 'Cat'",
    'enabled'    => count(
                        array_filter($request->all_items, function ($item) {
                            return $item->product->name == 'Cat';
                        })
                    ) > 0,
    'price'      => 10
]);
addMethod('id_017', [
    'title'      => "If all items have the option 'Size' greater or equal to 1",
    'enabled'    => count(
                        array_filter($request->all_items, function ($item) {
                            return isset($item->options['Size']) && $item->options['Size']['value'] >= 1;
                        })
                    ) == $request->package_qty,
    'price'      => 10
]);
addMethod('id_018', [
    'title'      => "More than 3 sku differents",
    'enabled'    => count(
                        array_unique(
                            array_map(function ($item) {
                                return $item->product->sku;
                            }, $request->all_items)
                        )
                    ) > 3,
    'price'      => 10
]);
```

Get minimum value, get maximum value

EXAMPLE WITH FUNCTIONS ADD-ON

```
addMethod('id_019', [
    'title'      => "Minimum price excluding tax without discount is greater to 10",
    'enabled'    => min(
                        array_map(
                            function ($item) {
                                return $item->base_original_price;
                            },
                            $request->all_items
                        )
                    ) > 10,
    'price'      => 10
]);
addMethod('id_020', [
    'title'      => "Maximum value of the option 'Size' is lower than 50",
    'enabled'    => max(
                        array_map(
                            function ($item) {
                                return $item->options['Size']['value'] ?? 0;
                            },
                            $request->all_items
                        )
                    ) < 50,
    'price'      => 10
]);
```

Sum

EXAMPLE **WITH FUNCTIONS ADD-ON**

```
addMethod('id_021', [
    'title'      => "Sum of all options 'Size' is greater than 30",
    'enabled'    => array_sum(
                        array_map(
                            function ($item) {
                                return $item->options['Size']['value'] ?? 0;
                            },
                            $request->all_items
                        )
                    ) > 30,
    'price'      => 10
]);
addMethod('id_022', [
    'title'      => "Calculate shipping fees by product",
    'price'      => array_sum(
                        array_map(
                            function ($item) {
                                // shipping is a custom product attribute
                                return $item->product->shipping * $item->qty;
                            },
                            $request->all_items
                        )
                    ),
]);
addMethod('id_023', [
    'title'      => "5 x weight of products that are in categories 2 or 3",
    'price'      => array_sum(
                        array_map(
                            function ($item) {
                                return count(array_intersect($item->product->category_ids, [2, 3]))>= 1
                                    ? $item->product->weight * $item->qty
                                    : 0;
                            },
                            $request->all_items
                        )
                    ) * 5.0,
]);
```

Using categories

Warning: you must notice that in Magento, **a product can be in multiple categories**.
So be particularly careful how you use this property.

EXAMPLE **WITH FUNCTIONS ADD-ON**

```
addMethod('id_024', [
    'title'      => "Sum of weight attributes of products in category 12",
    'price'      => array_sum(
                        array_map(
                            function ($item) {
                                return in_array(12, $item->product->category_ids)
                                    ? $item->product->weight : 0;
                            },
                            $request->all_items
                        )
                    ),
]);
addMethod('id_025', [
    'title'      => "Sum of weight attributes of products in categories whose id is 11 and 12",
    'price'      => array_sum(
                        array_map(
                            function ($item) {
                                return count(array_intersect($item->product->category_ids, [11, 12])) >= 1
                                    ? $item->product->weight : 0;
                            },
                            $request->all_items
                        )
                    ),
]);
addMethod('id_026', [
    'title'      => "Sum of weights of products having for first category id 12",
    'price'      => array_sum(
                        array_map(
                            function ($item) {
                                return $item->product->category_id == 12
                                    ? $item->product->weight : 0;
                            },
                            $request->all_items
                        )
                    ),
]);
```