INSTALLATION

Payment Surcharge for Magento 2

Compatibility of Payment Surcharge (M2)

Based on our experience with Magento 1 and surcharges, certain custom payment methods (those which send cart contents but do not honour Magento custom totals like surcharges) may not be compatible with Fooman Payment Surcharge out of the box. Our extensions include a 30 day money back guarantee, so feel free to test out the extension with any custom payment method if you suspect it may not not honour Magento custom totals. We are also more than happy to take a look at the implementation details of your payment method.

Will Fooman Payment Surcharge (M2) work with my one step checkout?

The extension is tested with default Magento payment methods and is compatible with Magento 2's default one page checkout.

Generally, third party one step checkout extensions which adequately replace Magento 2's default functionality are compatible with Fooman Payment Surcharge. If your one step checkout extension ticks the following boxes, the chances are good that Fooman Payment Surcharge will work out of the box:

- Upon switching of the payment method, the review section reloads •
- Custom Totals are displayed
- Custom Totals are displayed using their configured block
- Settings are honoured
- Totals sort order is honoured

The ones that don't work out of the box are most often the ones that look at the individual cart items instead of charging the grand total of the order while neglecting custom totals (due to the way the one step checkout extension is coded which is beyond the control of Fooman).

Rest assured that our extensions include a 30 day money back guarantee, so feel free to test out the extension on your own development site with other extensions.

Access Surcharge Information via Code

Our Surcharge extensions save their total information as an extension attribute. The below code shows how you would access the individual totals:

```
private function addFoomanTotals(\Magento\Sales\Api\Data\OrderInterface $order)
{
    $extAttr = $order->getExtensionAttributes();
    if (!$extAttr) {
        return;
    }

    $foomanGroup = $extAttr->getFoomanTotalGroup();
    if (empty($foomanGroup)) {
        return;
    }

    $totals = $foomanGroup->getItems();
    if (empty($totals)) {
        return;
    }
    foreach ($totals as $total) {
        /** @var \Fooman\Totals\Api\Data\OrderTotalInterface $total */
        $description = $total->getLabel();
        $baseAmount = $total->getBaseAmount();
        $baseTaxAmount = $total->getBaseTaxAmount();
        $amount = $total->getAmount();
        $taxAmount = $total->getTaxAmount();
        $identifier = $total->getTypeId();
        $qty = 1;
        /** @TODO do something with the total */
    }
}
```

Surcharges via REST Api

Fooman Surcharges will automatically appear in REST responses for individual orders/invoices/creditmemos under the extension_attributes key.

For example the below response for salesOrderRepositoryV1Get will list the surcharges under extension_attributes > fooman_total_group > items:

/ orders / {id} / **sales Order Repository V1Get Get**

GET {{baseUrl}}/V1/orders/:id

Params • Authorization Headers (8) Body Pre-request Script Tests

Query Params

	KEY	VALUE
	Key	Value

Body Cookies (1) Headers (11) Test Results

Pretty Raw Preview Visualize JSON


```
        ],
        "payment_additional_info": [
            {
                "key": "method_title",
                "value": "Check / Money order"
            }
        ],
        "applied_taxes": [],
        "item_applied_taxes": [],
        "fooman_total_group": {
            "items": [
                {
                    "type_id": "payment1",
                    "code": "fooman_surcharge",
                    "amount": 5,
                    "base_amount": 5,
                    "label": "test",
                    "tax_amount": null,
                    "base_tax_amount": null
                }
            ]
        }
    }
}
```

Creating Refunds for Orders with Surcharges

By default when creating a creditmemo the whole surcharge amount will be refunded. If you wish to adjust the amount to a lower amount you can append an amount override to the REST Api url like the below:
?creditmemo[fooman_surcharge][payment1]=3.50
where payment1 is the type id as seen in the previous API response.

So the full url would be something like this

/V1/order/:orderId/refund?creditmemo[fooman_surcharge][payment1]=3.50

for offline payment methods. And this for creating an online refund:

/V1/invoice/:invoiceId/refund?creditmemo[fooman_surcharge][payment1]=3.50

Which results in

Note: These instructions apply to installing an extension downloaded from the Fooman website only. If you downloaded the extension from the Magento Marketplace, follow these installation instructions instead.

Option 1: Preferred Installation Method (Fooman-Hosted Repository)

Fooman will host your extension and will automatically push new extension updates to your repository, ready for you to pull/update in the future.

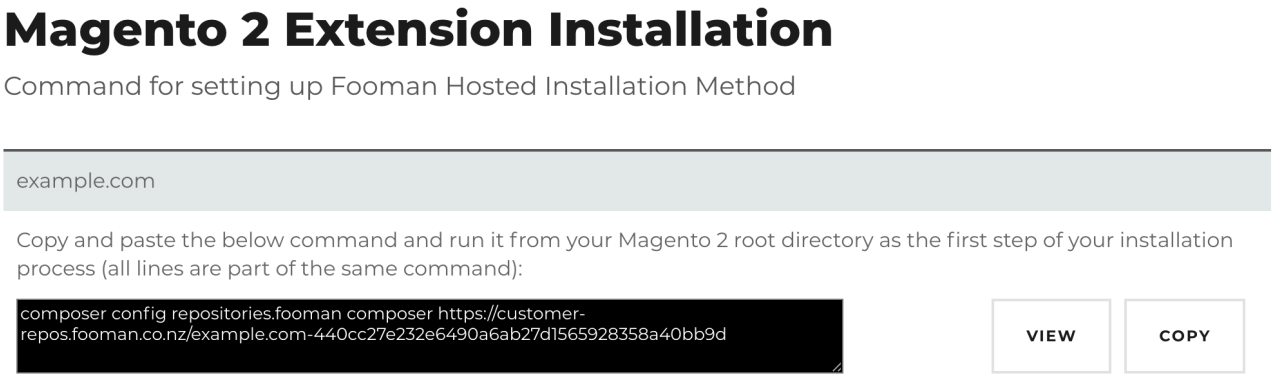
The advantages of this approach are:

- Saves time installing
- Saves time on installing future extension updates (you don't need to manually download these from the Fooman website and repeat the installation process each time)
- Helps to avoid common errors some people make when using the manual installation

method
- It integrates better with a multi-stage deployment process

New extension updates will be pushed automatically to this repository for as long as your support/update period is valid.

1. Log into your account on the Fooman website, using your login details you signed up with. On your My Extensions page you will see a custom command line. Copy the command line:



2. From a command line in your Magento root folder run

```
1. [The custom command line you copied from your Fooman Account dashboard]
2. composer require fooman/surcharge-payment-m2:^3.0
3. bin/magento module:enable --clear-static-
content Fooman_Totals Fooman_Surcharge Fooman_SurchargePayment
4. bin/magento setup:upgrade
```

If you are using Production Mode please also run

```
5. bin/magento setup:static-content:deploy
6. bin/magento setup:di:compile
```

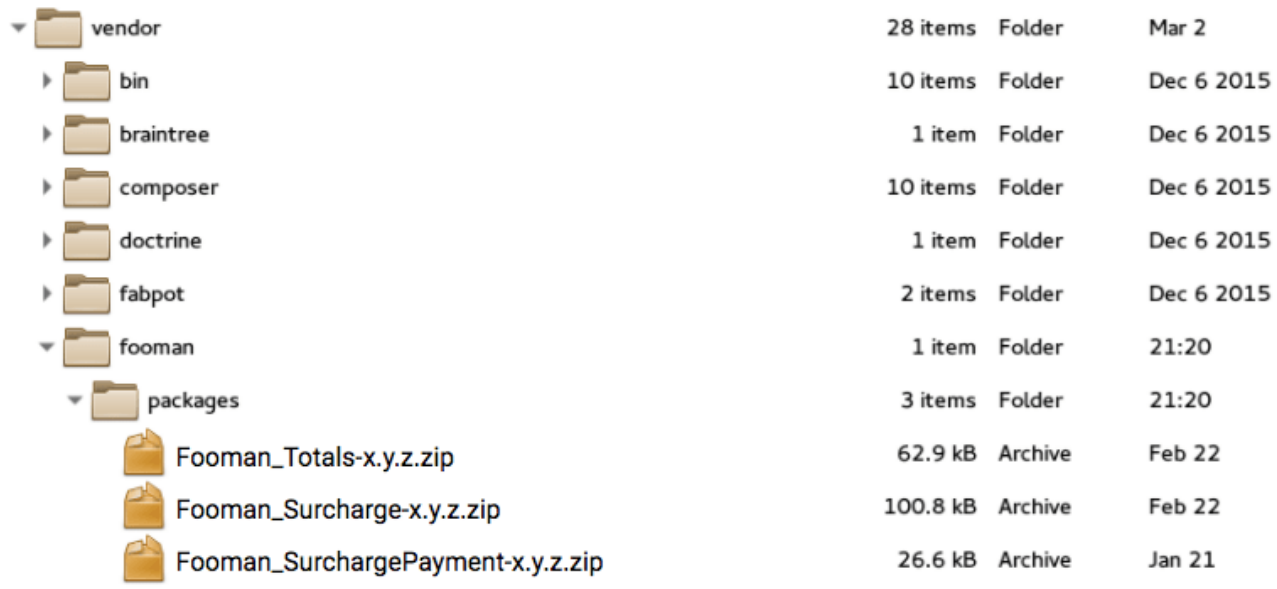
In the above example, line 1 would look like this:

```
1. composer config repositories.fooman composer  https://customer-
repos.fooman.co.nz/www.example.com-1a5a588b88a32a68260806391601f153214628055
```

Option 2: Manual Installation Method (Self-Hosted Repository)

1. Unzip the zip file you downloaded from our store into the following folder (please create this folder if it's not already present):

MAGENTO_ROOT/vendor/fooman/packages



2. From a command line in your Magento root folder run

```
1. composer config repositories.foomanartifacts artifact $(pwd)/vendor/fooman/packages
2. composer require fooman/surcharge-payment-m2:^3.0
3. bin/magento module:enable --clear-static-
content Fooman_Totals Fooman_Surcharge Fooman_SurchargePayment
4. bin/magento setup:upgrade
```

If you are using Production Mode please also run

```
5. bin/magento setup:static-content:deploy
6. bin/magento setup:di:compile
```

Uninstall Payment Surcharge (M2)

From a command line in the Magento root folder run:

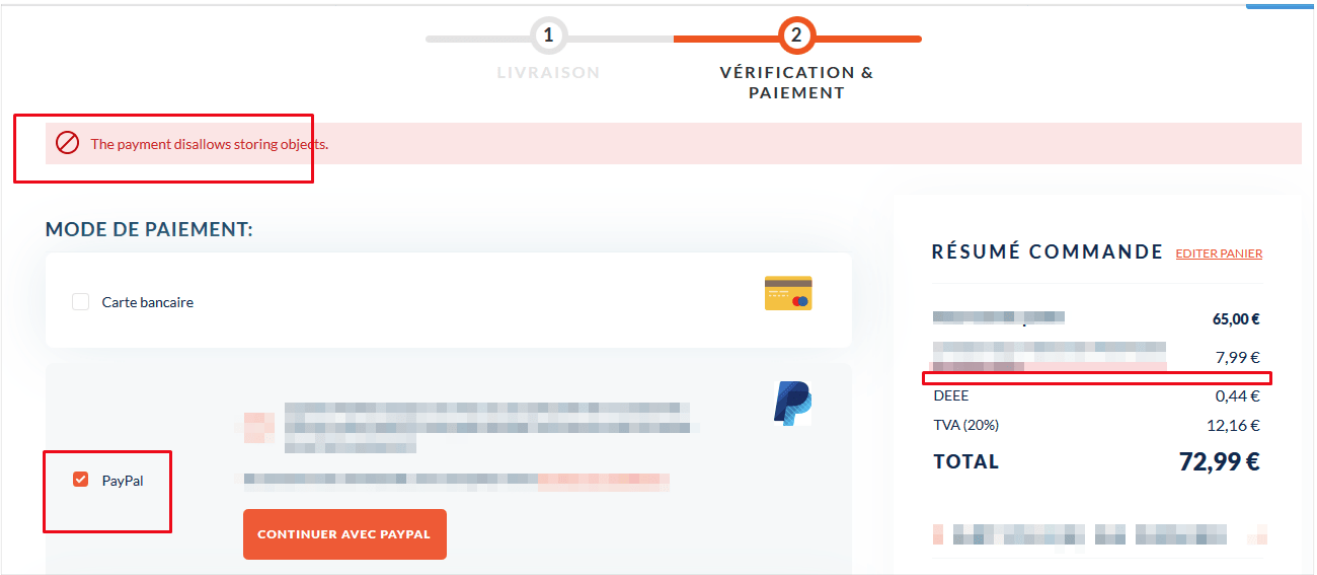
```
bin/magento module:disable Fooman_Totals Fooman_Surcharge Fooman_Surc
hargePayment
bin/magento module:uninstall Fooman_Totals Fooman_Surcharge Fooman_Su
rchargePayment --remove-data
bin/magento setup:upgrade
composer remove --update-with-dependencies fooman/surcharge-payment-
m2
```

If you have used the Manual Installation Method (Option 2), then please delete the following files if they exist.

```
rm vendor/fooman/packages/Fooman_Totals-*.zip
rm vendor/fooman/packages/Fooman_Surcharge-*.zip
rm vendor/fooman/packages/Fooman_SurchargePayment-*.zip
```

Paypal error message : The payment disallows storing object

On some Magento 2.2 versions (including 2.2.3 - 2.2.5) when PayPal is selected as payment method in combination with a configured Surcharge for PayPal it shows the warning "The payment disallows storing objects":



This is due to the core Magento bug discussed at https://github.com/magento/magento2/issues/11885 which affects a few versions.

To fix this you can install this module https://github.com/CopeX/PaypalFix with:

```
composer config repositories.copexpaypal vcs https://github.com/CopeX/PaypalFix
composer require copex/module-paypalfix:dev-master
bin/magento module:enable CopeX_PaypalFix
bin/magento setup:upgrade
```

Paazl Shipping - Call to a member function getFoomanTotalGroup() on array

The Paazl Shipping extension unfortunately sets extension attributes on the address object to an array which should not be possible. The developers themselves have acknowledged this with a note in their code:

```
@todo Temporary file until I find a cleaner way to alter the payload / set the extension att
ributes server side.
```

This in turn triggers the following error message in our code:

```
{"messages":{"error":[{"code":500,"message":"Fatal Error: 'Uncaught Error: Call to a member
function getFoomanTotalGroup() on array in /public_html\/vendor\/fooman\/surcharge-core-
m2\/src\/Model\/Total\/Quote\/Surcharge.php:163
Stack trace:
#0 /public_html\/vendor\/fooman\/surcharge-core-m2\/src\/Model\/Total\/Quote\/Surcharge.php(
120): Fooman\\Surcharge\\Model\\Total\\Quote\\Surcharge->removeInactiveSurcharges(Object(Mag
ento\\Quote\\Model\\Quote\\Interceptor), Array)
#1 /public_html\/vendor\/magento\/module-quote\/Model\/Quote\/TotalsCollector.php(274): Foom
an\\Surcharge\\Model\\Total\\Quote\\Surcharge->collect(Object(Magento\\Quote\\Model\\Quote\\
Interceptor), Object(Magento\\Quote\\Model\\ShippingAssignment), Object(Magento\\Quote\\Mode
l\\Quote\\Address\\Total))
#2 /public_html\/vendor\/magento\/module-quote\/Model\/ShippingMethodManagement.php(303): Ma
gento\\Quote\\Model\\Quote\\TotalsCollector->collectAddressTotals(Object(Magento\\Quote\\Mod
el\\Quote\\Interceptor), Object(Paazl\\Shipping\\Model\\Quote\\Address\\Interceptor))
#3 /public_html\/ve' in '/public_html\/vendor\/fooman\/surcharge-core-m2\/src\/Model\/Total\
/Quote\/Surcharge.php' on line 163","trace":"Trace is not available."}]}}
```

Until this issue is addressed in the Paazl Shipping extension you can apply the following edit to /app/code/Paazl/Shipping/Model/PaazlManagement.php and change the below

```
 } else if (is_array($extensionAttributes)) {
                    if (!empty($extensionAttributes['checkout_fields'])) {
                        $streetName = $extensionAttributes['street_name'];
                        $houseNumber = $extensionAttributes['house_number'];
                        $addition = null;
                        if (isset($extensionAttributes['house_number_addition'])) {
                            $addition = $extensionAttributes['house_number_addition'];
                        }
              }
                }
```

to read

```
 } else if (is_array($extensionAttributes)) {
                    $addressExtension = $this->addressExtensionFactory->create();
                    if (!empty($extensionAttributes['checkout_fields'])) {
                        $streetName = $extensionAttributes['street_name'];
                        $houseNumber = $extensionAttributes['house_number'];
                        $addition = null;
                        if (isset($extensionAttributes['house_number_addition'])) {
                            $addition = $extensionAttributes['house_number_addition'];
                        }
                        $addressExtension->setStreetName($streetName);
                        $addressExtension->setHouseNumber($houseNumber);
                        $addressExtension->setHouseNumberAddition($addition);
                    }
                    $address->setExtensionAttributes($addressExtension);
                }
```

Amasty Promo compatibility issues with Fooman Surcharge

As per customer

It's getting more and more likely that the fault lies within Amasty, and your extension just happens to trigger their flaws.

They seem to have an aroundCollectAddressTotals plugin where they look through every quote item on the QuoteAddress. Inside this they call their own function named "updateQuoteItems"
In this function they loop over the quote items again. When the code goes through this function, it removes the quoteItem because of a flaw in their code.

Patch attached

amasty_gift-(2).patch

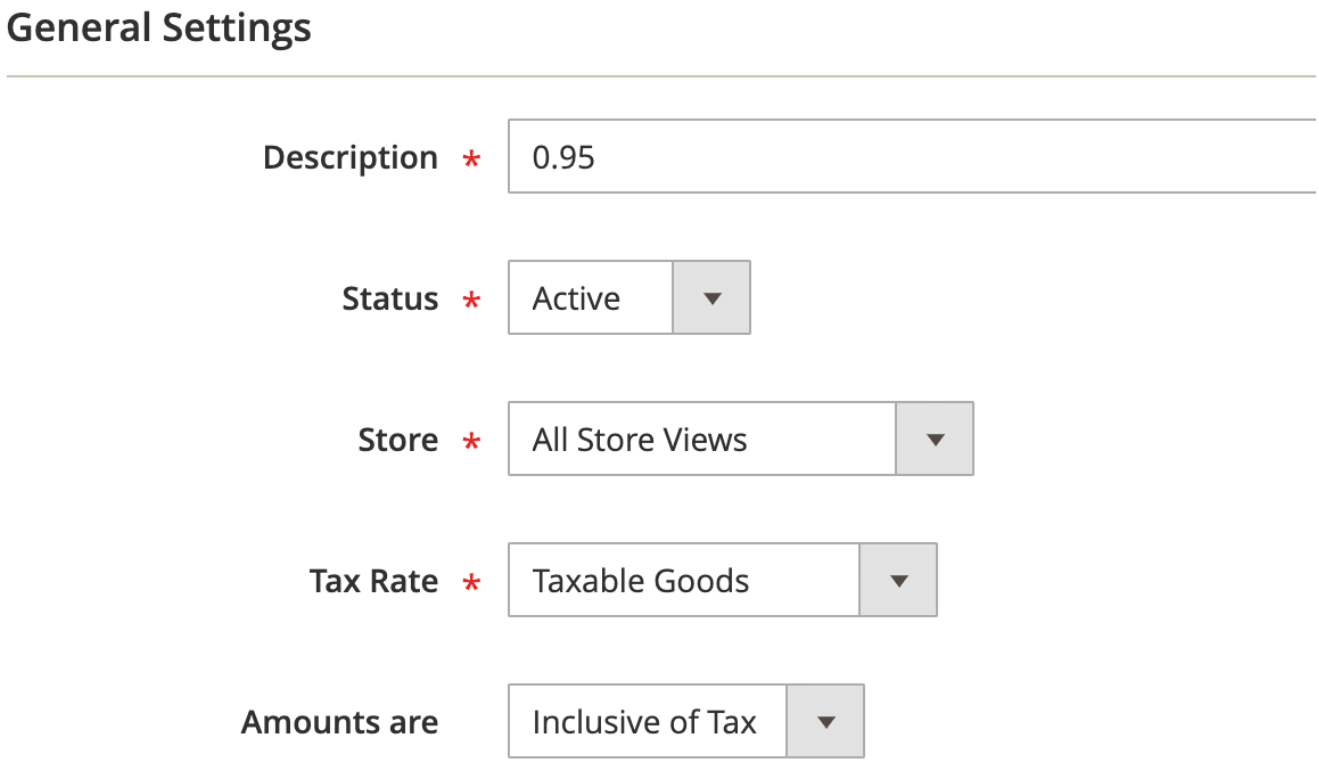
Tax changes fixed surcharge by 0.01

To enter a fixed amount surcharge that includes tax at a given fixed amount please use the following set up.

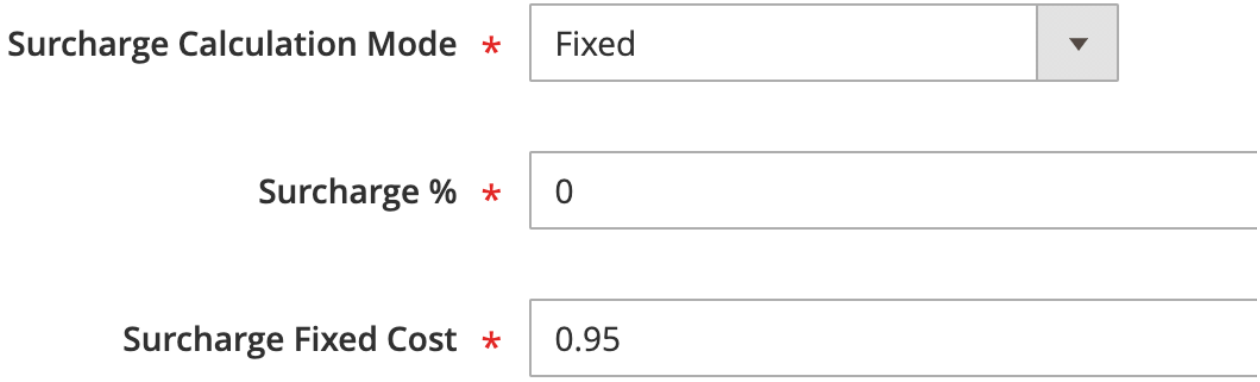
Surcharge Settings

For the surcharge itself please use

Tax Rate = Taxable Goods (or your equivalent)
Amounts are = inclusive of Tax



and entering a fixed amount

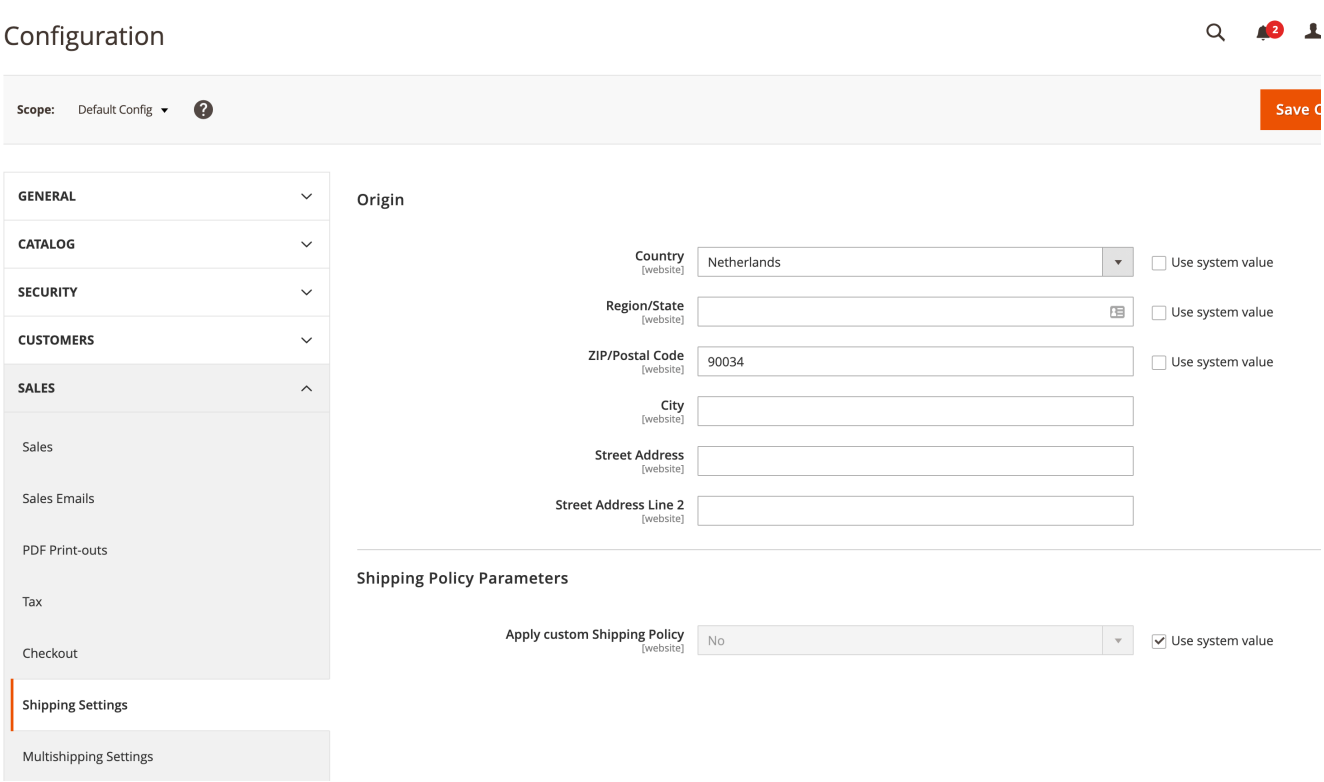


If you are still seeing a different amount to the entered 0.95 your tax configuration needs to be adjusted.

Tax Settings

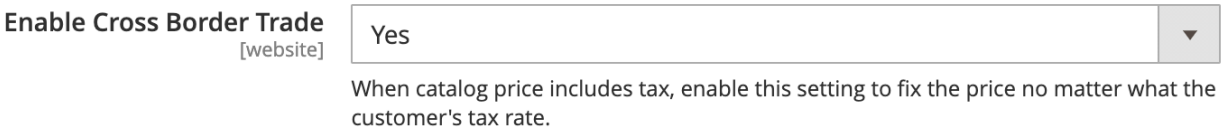
Option 1

For the fixed amount to end up as entered Magento will run the following process, factoring out the tax with your default customer group and your default tax configuration. Then apply the current tax rate. For this to end up with exact same amount the two tax rates need to be the same. One setting that needs to be checked is Sales > Shipping Settings > Origin. This Origin needs to produce a tax rate with the same percentage as the Tax Rate selected for the surcharge.



Option 2

use Enable Cross Border Trade as per the comment for the setting (please note this will apply for all catalog prices)



Reporting Any Issues/Bugs

We are proud of our quality extension code - it’s been widely tested and we stand by it 100%. If something does happen and you think you might be experiencing an issue or bug, please contact us via support@fooman.co.nz and we will help you out.