Follow Up Email Manual

Getting Started

Welcome to the **Follow-Up Email** documentation.

Here, you can find everything you need to set up your **Follow-Up** fully-featured campaigns and create a vast promotional network, motivating your customers with a rich income.

Go ahead, dive in!

Firstly, please visit our extension in the My Downloadable Products section of our store. Learn how to install extension, and proceed with Quick Start, which will guide you through setting up your Follow-Up service.

How to install the extension

1. Back up your store database and web directory.
2. Log in to the SSH console on your server and navigate to the root directory of the Magento 2 store.
3. Copy installation instructions from the page My Downloadable Products to the SSH console and press ENTER.
4. Run the command `php -f bin/magento module:enable Mirasvit_Core Mirasvit_EmailDesigner Mirasvit_Email Mirasvit_EmailReport Mirasvit_Event` to enable the extension.
5. Run the command `php -f bin/magento setup:upgrade` to install the extension.
6. Run the command `php -f bin/magento cache:clean` for clean cache.
7. Deploy static view files

   ```
   rm -rf pub/static/*; rm -rf var/view_preprocessed/*; php -f bin/magento setup:static-content:deploy
   ```

Note

If you install the module manually to the `"app/code/"` directory, you additionally need to install the required libraries through the composer:

```
composer require "liquid/liquid":"~1.4"

composer require "geoip2/geoip2":"^2.9"
```

Quick Start

Our extension is simple yet powerful. Once you install it, you can proceed with the creation of your promotional campaigns.

Here are some tips to quickly tackle our key features:

1. Basic building blocks are **Templates**, which contain actual messages, sent in emails. Enrich them with **Liquid Variables** and give them a consistent look & feel with **Themes**.

2. Create **Campaigns** to start your promotional service. Each campaign will require at least one **Trigger** - an action that starts sending **emails chain**.

   We have a nice set of **examples of campaigns and triggers**, which will help you master them.

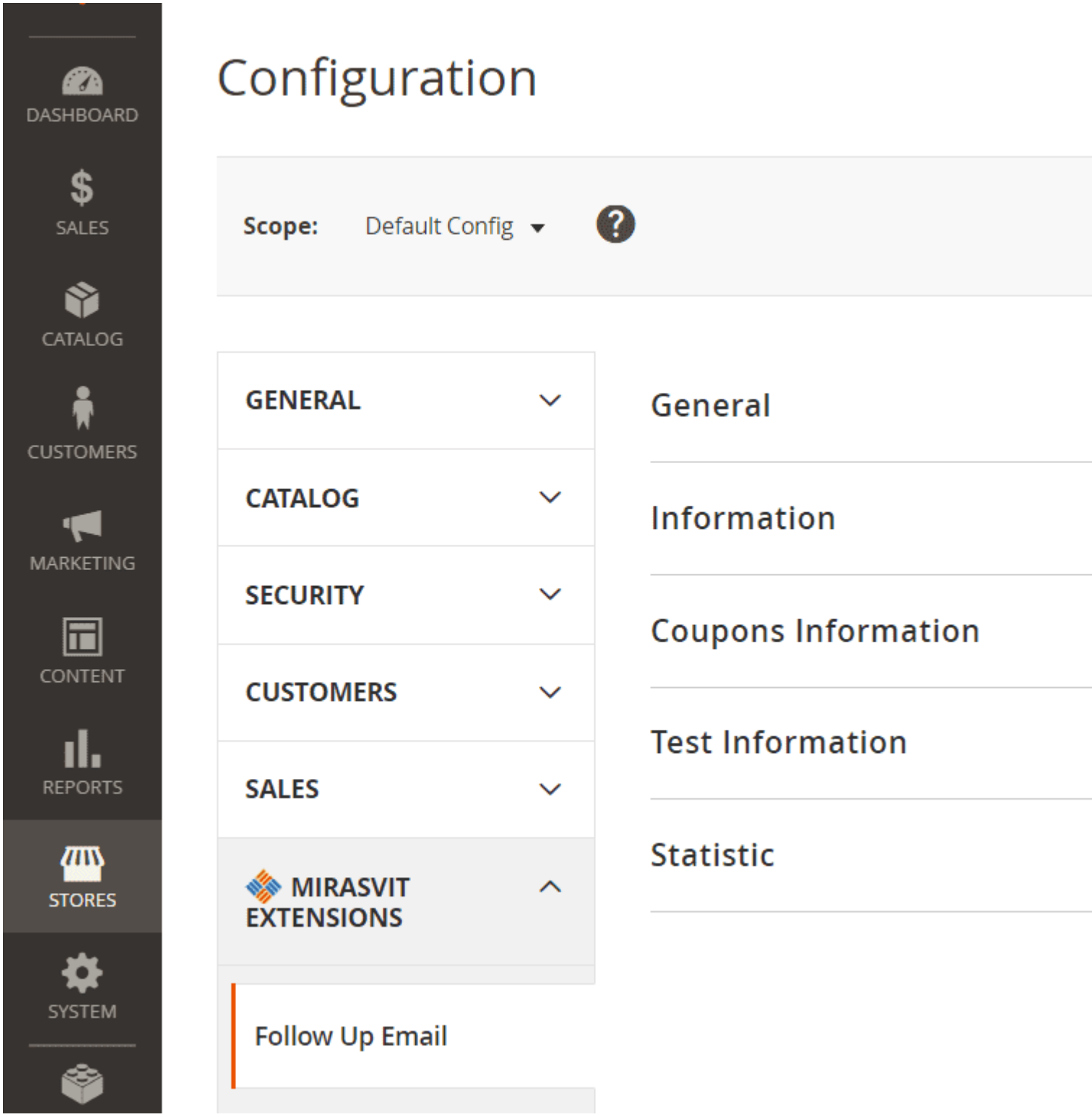
3. Test your campaign with our built-in simple **testing framework**.

4. Analyze **Mail Logs** and **Event Logs** and adjust your promotional campaigns for maximum efficiency.

5. If you need to adjust your service to GDPR, consult our **GDPR Tips** section.

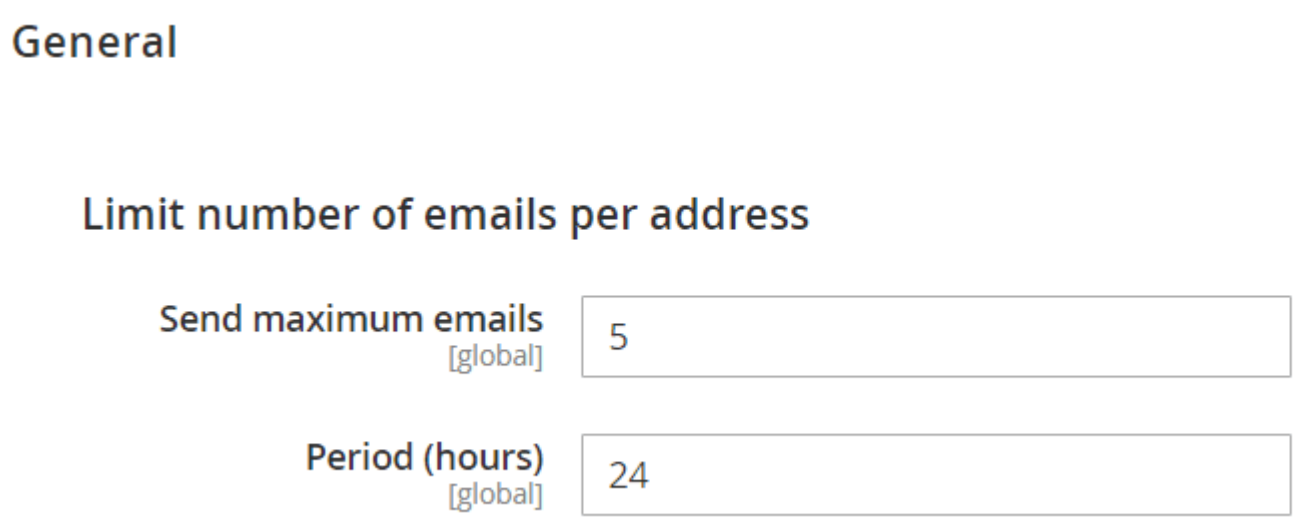
This should be a good start. Refer to the respective sections of this manual to learn more.

General Settings



General Follow Up Email settings are located at **Marketing -> Follow Up Email -> Settings**, and consist of the following sections:

General



Option	Description
Limits the number of emails per address	Allows you to limit the maximum number of emails sent per address within the specified period.
Send maximum emails	Sets a maximum number of emails allowed for a specified **Period**.
Period (hours)	Sets the period for the maximum amount of emails which can be set at **Send maximum emails**.

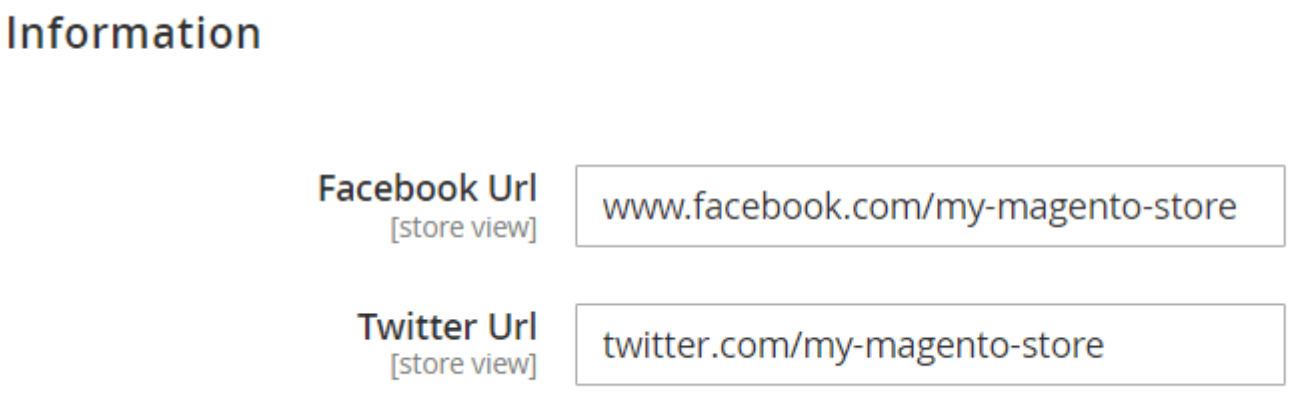
Example

Limit the number of emails:

```
Send maximum emails  3
Period (hours)       24
```

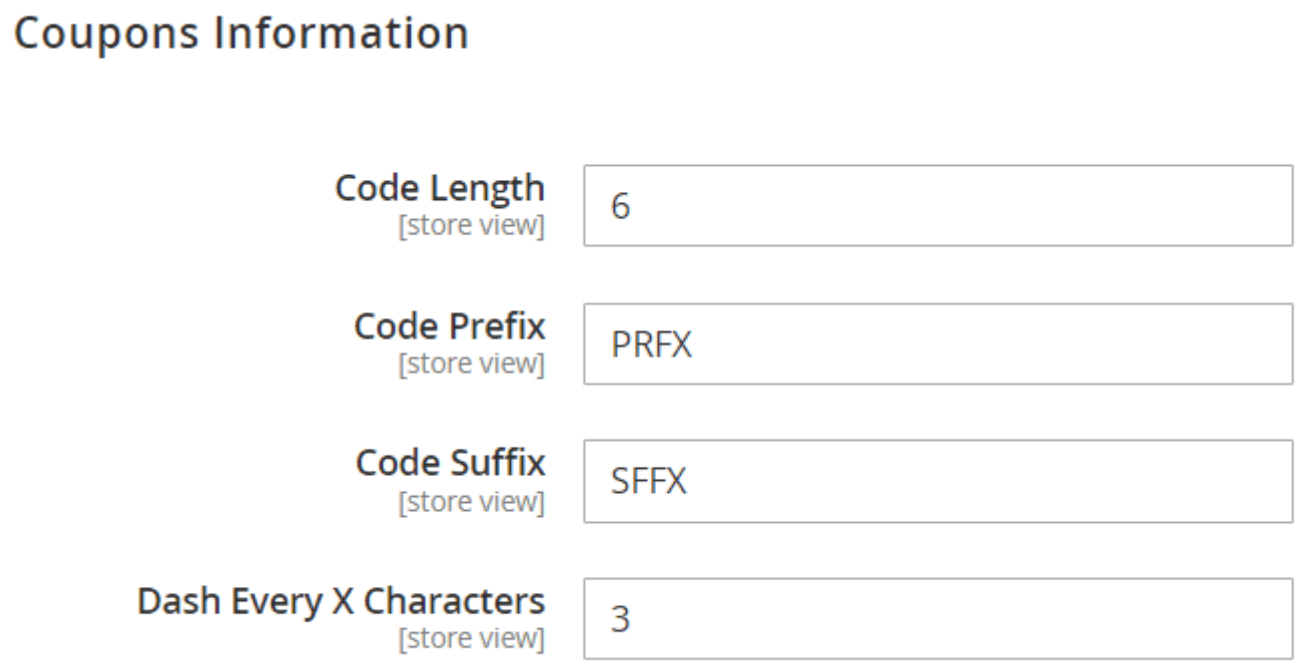
If a customer has already received 3 emails from you within 24 hours, all other emails during the same 24 hours will be canceled

Information



Option	Description
Twitter Url	Allows you to add a Twitter URL to the trigger emails.
Facebook Url	Allows you to add a Facebook URL to the trigger emails.

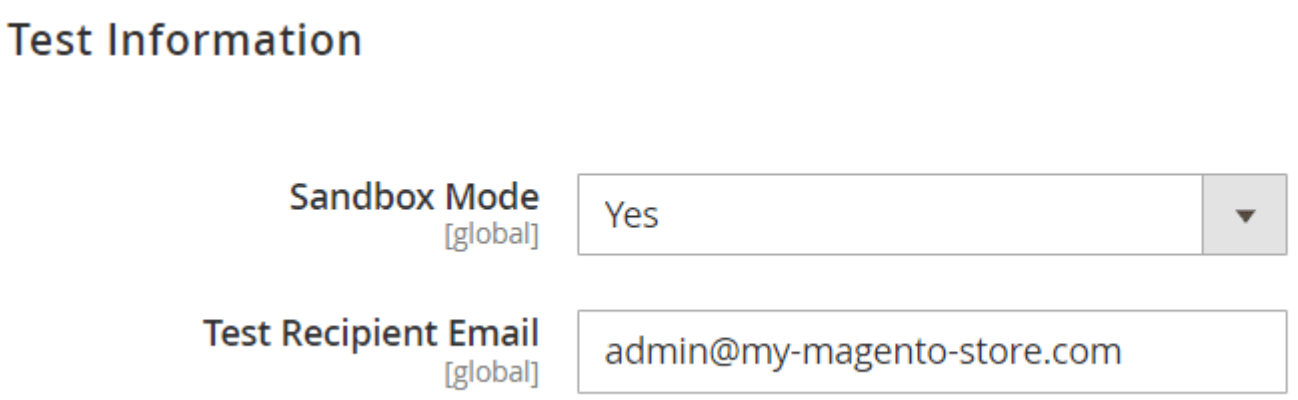
Coupons Information



The Follow Up Email extension generates the coupon codes based on the selected shopping cart price rule. Using the settings below, you can configure an appearance of the generated coupon code.

Option	Description
Code Length	Length of the coupon code, excluding prefix, suffix, and separators.
Code Prefix	Specifies common prefix used for coupon codes generated by Follow Up Email extension.
Code Suffix	Specifies common suffix used for coupon codes generated by Follow Up Email extension.
Dash Every X Characters	Adds dash character every X symbol to the coupon code.

Test Information

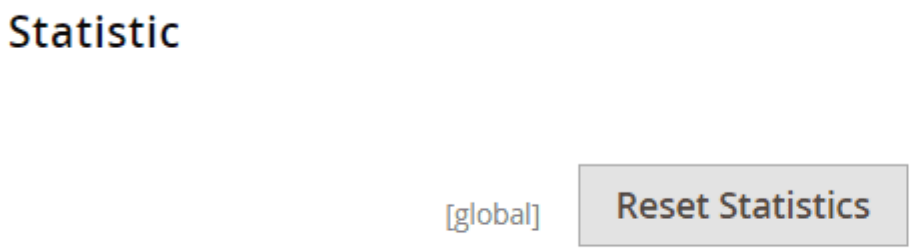


This section governs Sandbox Mode, which can be used for **Campaign Testing**.

Option	Description

Sandbox Mode if this option is enabled, all emails will be sent only to the **Test Recipient Email**.
Test Recipient Email Sets receiving email for trigger emails if **Sandbox Mode** is enabled.

Statistic

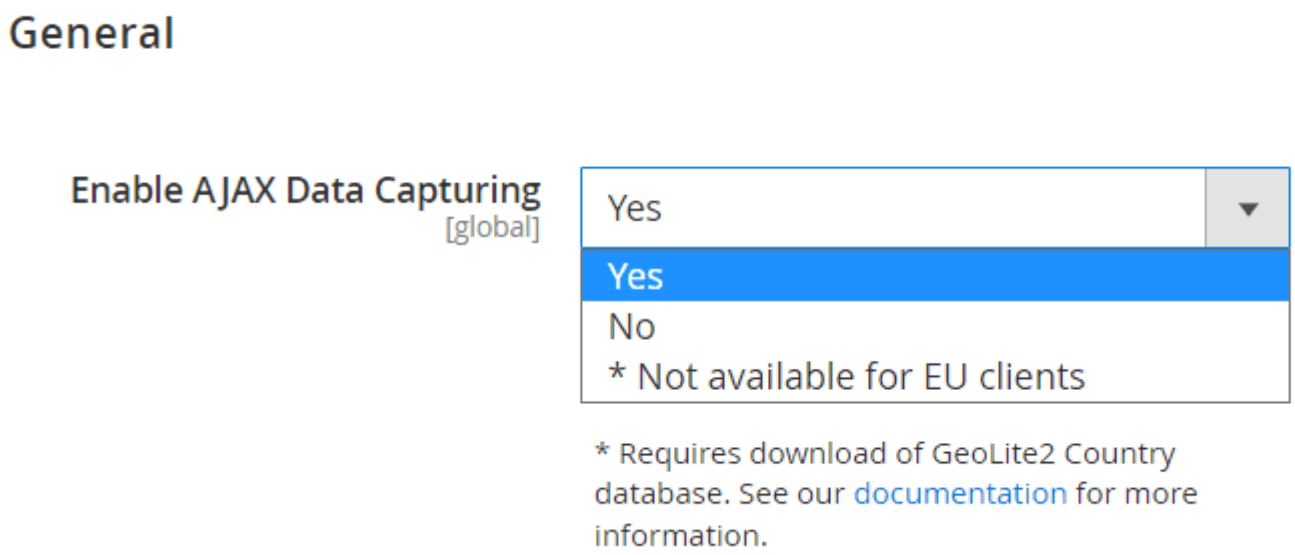


There is only one option-- **Reset statistic**, which will clear your campaigns' data.

Event Settings

Go to **Stores -> Settings -> Configuration**. In the panel on the left under **Mirasvit Extensions**, choose **Event**. You will see the following settings:

General



- **Enable AJAX Data Capturing** - allows you to control guest users' data capturing.

Useful Info
When enabled, our module automatically captures the guest customer's (not logged in) first name, last

name, and email when a client enters this information in your store's fields for later use while sending emails.

Available options:

- **Yes** - data capturing enabled for all users
- **No** - data capturing disabled for all users
- **No for EU clients only** - data capturing disabled for EU users only

Note

To use this option, make sure that you download the GeoLite2 Country database, otherwise the data capturing is disabled for all users.

- **GeoLite2 Country database path** - the absolute path to the GeoLite2 Country database on your server. Make sure to copy the database file **GeoLite2-Country.mmdb** to the folder available for the web-server user, e.g. `/magento_root_folder/var/GeoLite2-Country.mmdb`

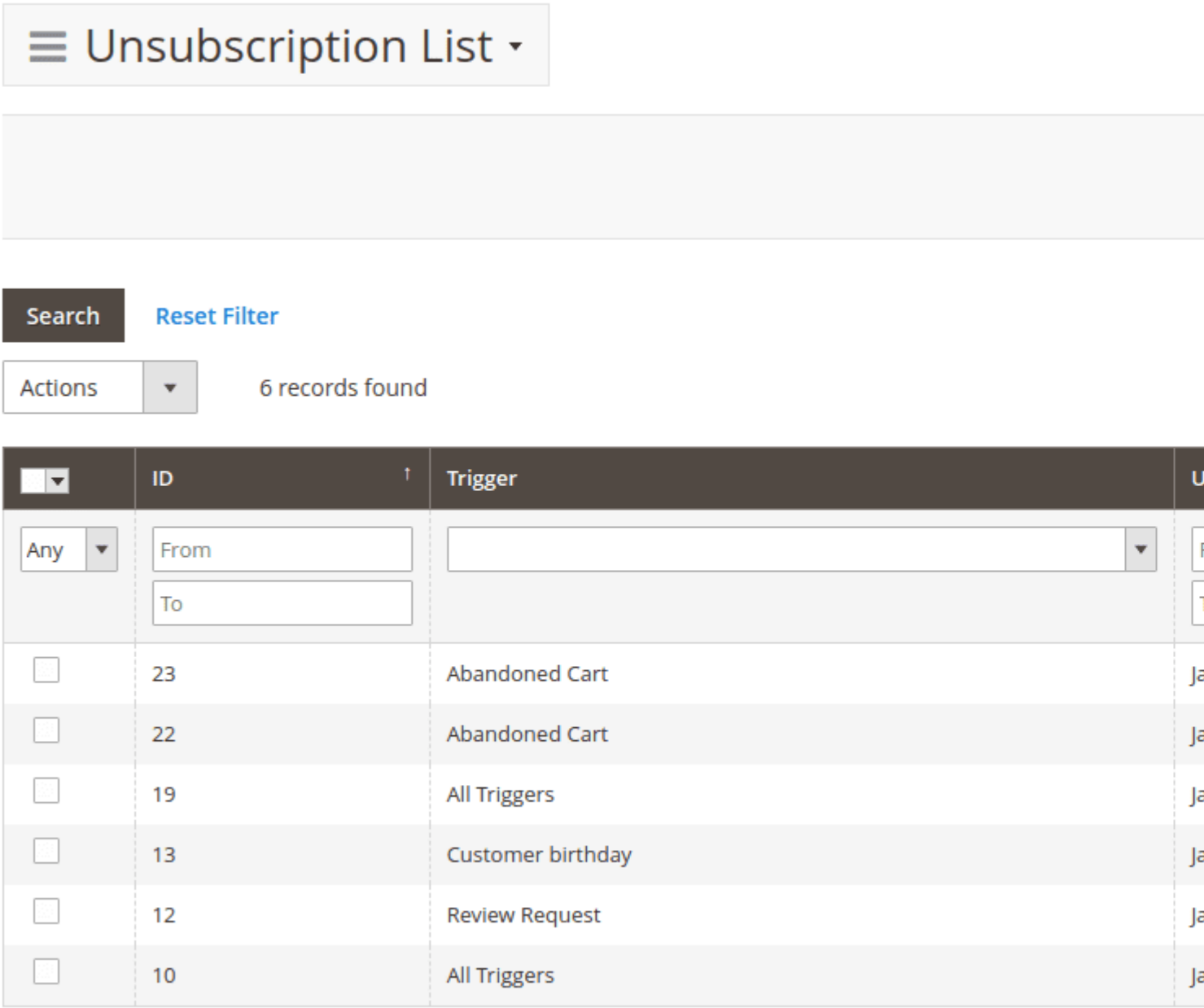
Useful Info

If the email extension was installed manually, GeoLite2 package should be installed on the server via composer command: **composer require geoip2/geoip2:~2.0**

Unsubscription List

The extension allows you to manage customer emails' subscription/unsubscription.

Go to **Marketing -> Follow Up Email -> Unsubscription List**.



At the main grid, you can see a list of all customers who have unsubscribed from the trigger emails, where:

- **ID** - internal unsubscription id number
- **Trigger** - trigger the customer unsubscribed from.
  If the customer unsubscribed from all triggers emails, the value will be "*All Triggers*"
- **Updated at** - last unsubscription status update
- **Email** - customer email

Note

Click the button **Subscribe** to remove customer unsubscription.
It is possible to make emails mass subscription by clicking the button **Subscribe** at tab menu **Actions**

Unsubscribe Emails Manually

At the main grid, click the button **Unsubscribe Email**.

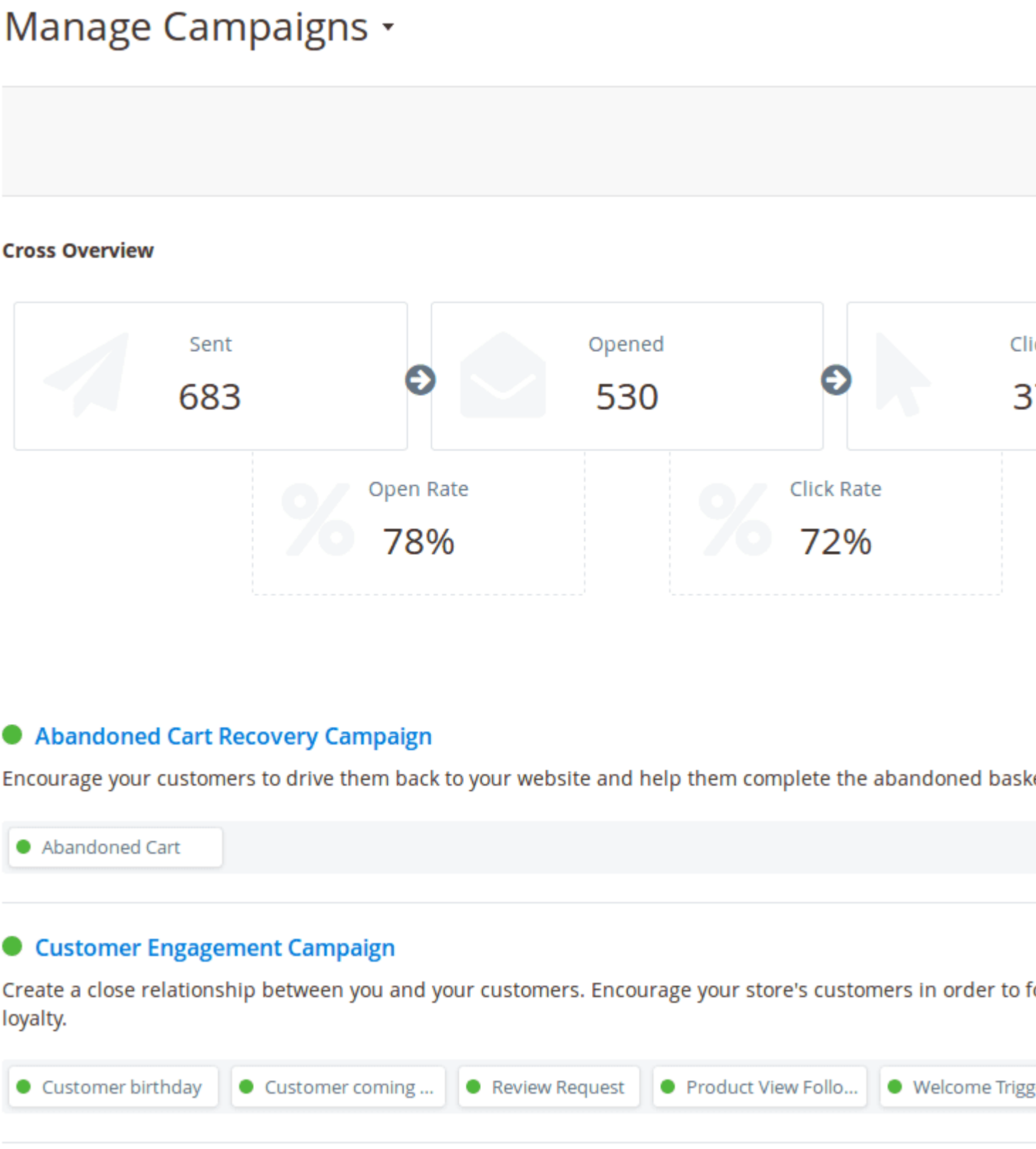
On the new page will be the following fields:

- **Set emails via comma to unsubscribe** - set customer email addresses
- **Triggers** - select customer triggers for unsubscription. To unsubscribe a customer from all trigger emails, select the option "All Triggers".

Campaigns Dashboard

A campaign is a central point in our Follow-Up Emails extension. Each campaign is a project which consists of one or more Triggers, each with its own Audience, triggering Event and Emails Chain.

All campaigns are located at **Marketing -> Follow-Up Email -> Manage Campaigns** and organized in the Dashboard with statistics, quick info, and basic actions. It is shown on the screenshot below:



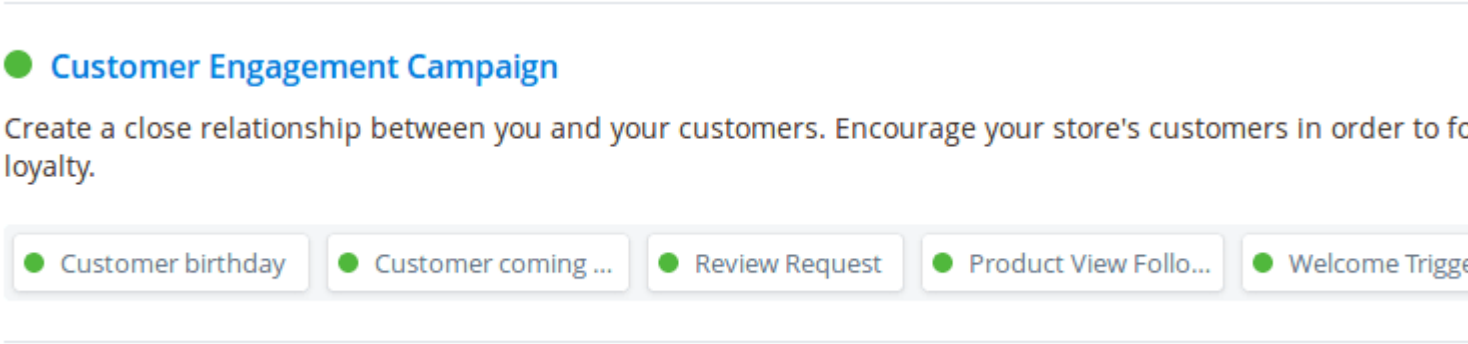
As you can see, the Dashboard consists of two subpanels - top and bottom.

The top is a **Cross Overview** - an overall statistics hub, where displayed, how all of your campaigns succeeded in general. It breaks into two rows:

- **Campaigns Metrics** - is the first row, which displays raw metrics, such as the quantity of **Sent** emails, **Opened** messages, **Clicked** links, created **Orders** and **Review** written and approved.
- **Rate Metrics** - is the second row, where displayed metrics relative to overall success, such as:
    - **Open Rate** - is calculated from **Sent** emails;

- **Click Rate** - is calculated from **Open Rate**;
- **Order Rate** - is the rate of placed **Orders** from **Click Rate**.
- **Review Rate** - is the rate of approved **Reviews** from **Click Rate**.

The bottom contains the list of all Campaigns that are defined in your store(s). Each row on this list contains the following information:

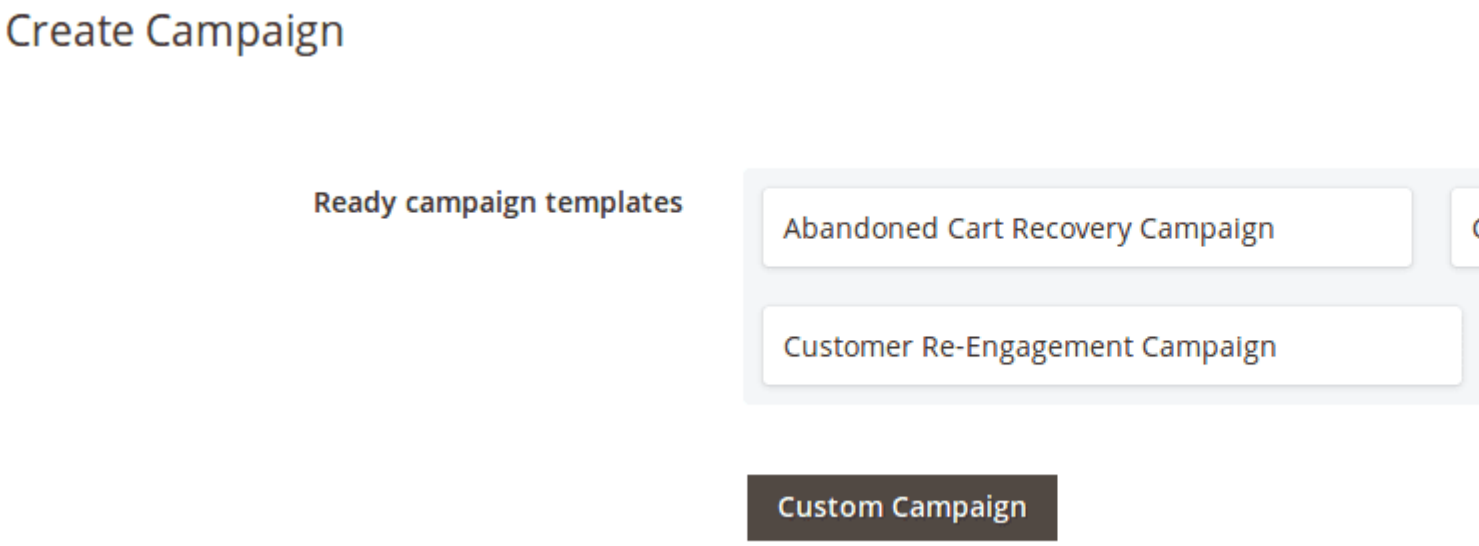


- **Title** - name of the Campaign. The green point shows whether current campaigns are active;
- **Short Description** - a few words about the current campaign;
- **Assigned Triggers Pane** - a list of triggers assigned to this campaign. Each element is a button, which instantly brings you to **Trigger** edit pane. The green point shows whether it is active;
- **Actions Pane** - which contains basic actions:
  - **View** - allows you to edit a campaign, or view it;
  - **Delete** - allows you to remove a campaign instantly;
  - **Duplicate** - creates a precise copy of the current campaign, saving time for creating similar campaigns;
- **Raw Metrics Pane** contains all raw performance metrics of the current campaign (see above).

Creating a Campaign

The simplest way to create a Campaign is to use the **Duplicate** action, and then adjust it to your needs.

Still, in most cases, you will need a custom campaign. You can create one by pressing **Create Campaign** button at **Marketing -> Follow-Up Email -> Manage Campaigns**. It will bring you to the first stage of Campaign creation:



At this stage, you can select one of the suggested (and defined before) campaigns as a template. Pressing on one of them is identical to employing the **Duplicate** function - our extension will create a campaign using one of the pre-defined templates, automatically creating all necessary events and triggers.

There is also a **Custom Campaign** button. It starts creating a Campaign from scratch, and brings you to the empty Campaign workspace.

To operate successfully, a Campaign needs to have one or more Triggers assigned to it. You can add a Trigger by pressing the **Add Trigger** button. You will then see a Trigger definition dialog which contains the basic data, broken into three categories:

Edit Trigger

General Information

Title *

Is Active No

Active From

Active To

Store View *

All Store Views
Main Website
Main Website Store
Default Store View

Sender Details

Sender Email

Sender Name

Send copy to email

These addresses will be added to the BCC. Separate e-m

Google Analytics Campaign

- **General Information**
  - **Title** - a sensible name for a Trigger
  - **Is Active** - defines whether a Trigger is active, and emails should flow
  - **Active From**, **Active To** - defines the date period during which a Trigger should be active
  - **Store View** - defines which store Trigger should work on.
- **Sender Details**
  - **Sender Email** - email address which will be used for sending emails
  - **Sender Name** - name (or title), which will be used for sending emails
  - **Send a copy to email** - here you can add one or more emails, where blind copies of all emails within this Trigger will be sent. It is used for email flow analysis.
- **Google Analytics Campaign** - a unique feature that allows you to analyze the number of visits, conversion rate, time of visits, etc. for those who arrive after reading a specific email. Read more about **this service**.
  - **Campaign Source** - defines the search engine, newsletter name, or other data sources. (available: `email`, `follow-up-email`, `newsletter`)
  - **Campaign Medium** - defines which medium analytics shall be used, e.g. `cpc`, `banner`, `email`.
  - **Campaign Name** - the name of your Google Analytics service.
  - **Campaign Term** - paid keywords for your campaign.

Note

After configuring the **Google Analytics** section, the extension will automatically add special `GET` params to all links in the emails. This way, following them will be recorded by Google automatically without any additional adjustments.

Example: `http://example.com/about-us/ will be converted to http://example.com/about-us/?utm-source=email&utm-medium=trigger-email&utm-name=review-request`.

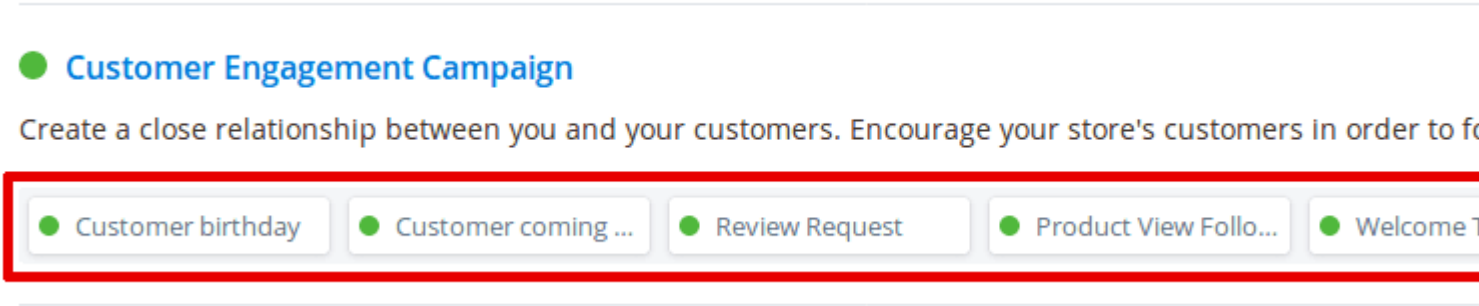
To track the campaign performance at **Google Analytics**, log in to your Google account and go to **Traffic Sources -> Campaigns**. Select the campaign source from the list, and you will receive a report.

After the Trigger is attached to the Campaign, you need to **set it up** and **assign audience**.

Setting Up an Email Trigger

A **Trigger** is an event or sequence of events that, as a result, generates a chain of emails.

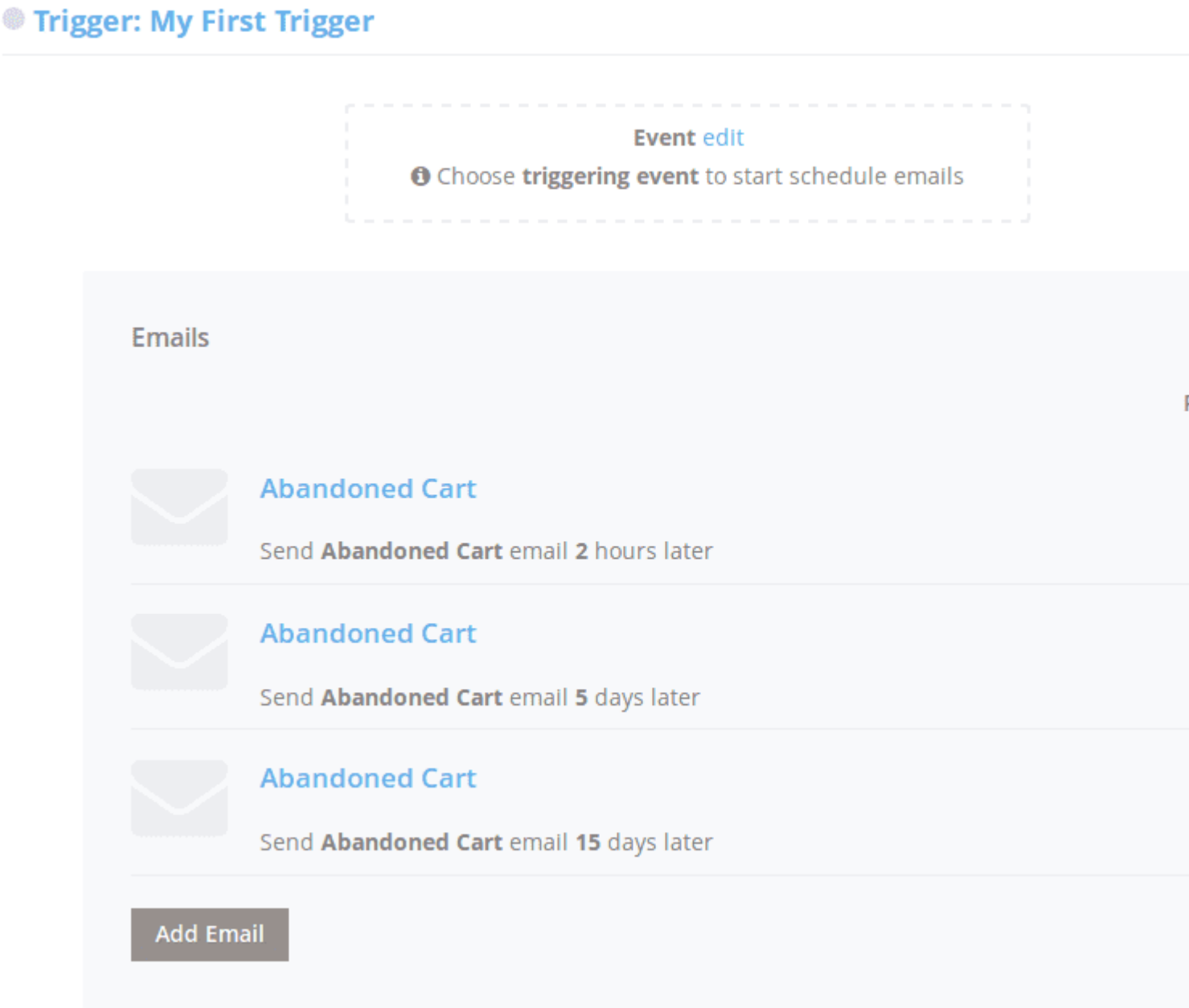
Each campaign has at least one trigger attached to it. To edit the trigger, you need to proceed to **Marketing -> Follow-Up Email -> Manage Campaigns** and select a Campaign, or directly click on the corresponding button on the Trigger Pane.



Each trigger has its own sub pane on the Campaign edit page.

It has two basic parameters that are required for proper work:

- **Event** - the action that will trigger email sending;
- **Audience** - the customers' group who are eligible to receive emails.



Assigning an Event

Event - a certain action of a visitor (e.g., login, registration, placing an order) or action of a system (e.g., change order status, change of price).

To assign an event to Trigger, press the **edit** link on the **Event** block. You will be asked for two parameters:

- **Triggering Event** - the event that launches a campaign;
- **Cancellation Event** - the event that will stop sending emails.

Both parameters use the same Event List which can be seen here.

Example

If you need to send follow-ups for tracking order status, then:

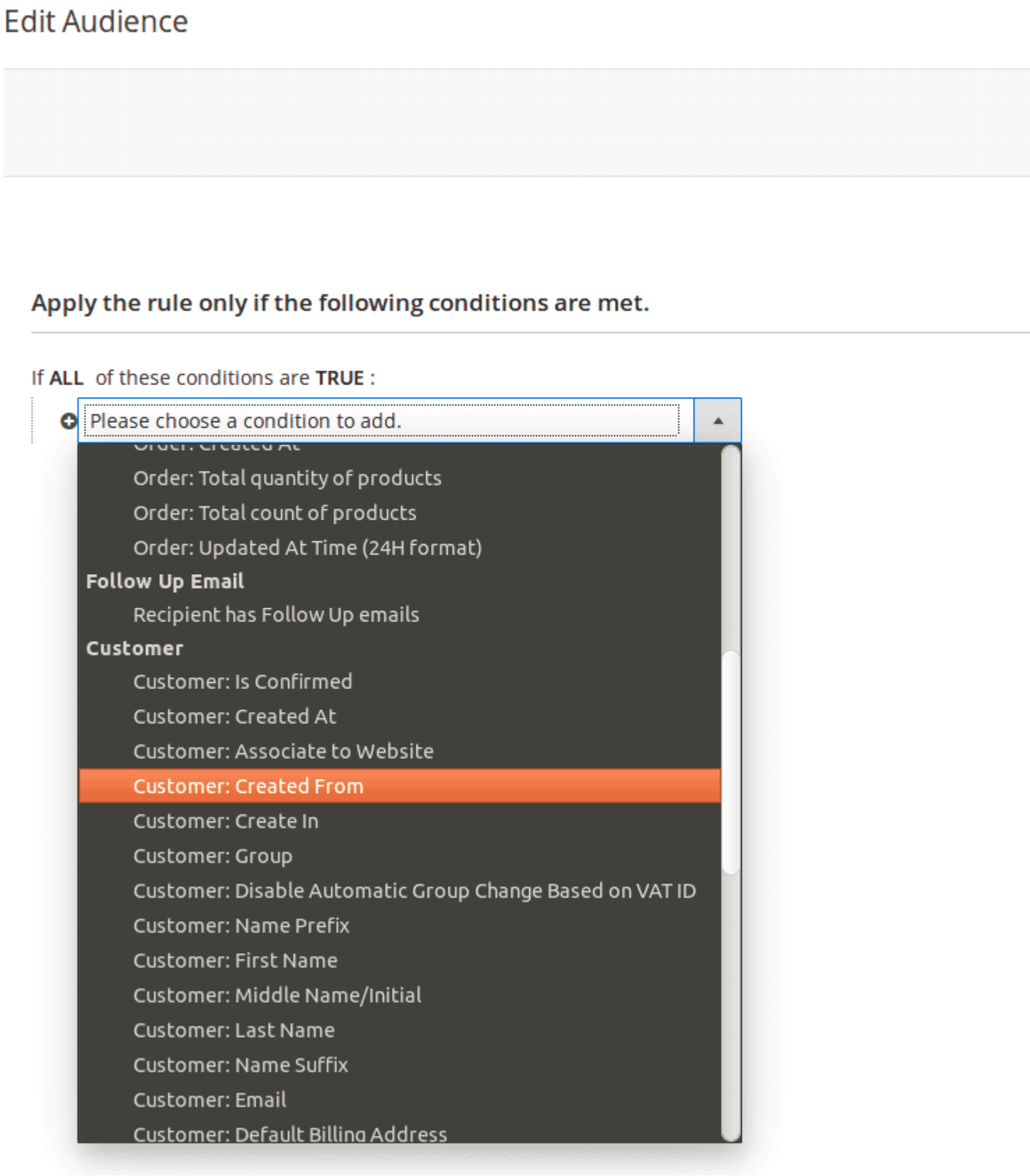
- **Triggering Event**: `Sales / Order obtained Pending status`
- **Cancellation Event**: `Sales / Order obtained Completed status`

Marking an Audience

The audience is defined as the group of customers who are eligible to receive emails during the current campaign.

This block can be empty: in this case, all of your customers will be eligible for the campaign and email sending. However, if you wish to have different email flows for different groups of customers, you need to mark **Audience**.

You need to press the **edit** link on the Audience block to view the audience's conditions.



This rule allows you to use the following conditions to limit the campaign audience, which are grouped to categories:

- **Store**
  - **Lifetime Sales**
  - **Number of Orders**
- **Products**

    - **Products Subselection** - allows you to analyze the current cart or order (this condition is used on **Order**-connected events). It spawns condition sub-block `If ALL/ANY products in cart/order matching these conditions`, which can contain one or more product's properties, which should trigger an email.
    - **Products Attribute Value Comparison** - allows for comparing elements of a cart or order, also spawning a sub-block with one or more conditions.
- **Order**
    - **Grand Total**
    - **Shipping Method**
    - **Shipping Created**
    - **Payment Method**
    - **Invoice Created**
    - **Status**
    - **Created At** - should use the format YYYY-MM-DD
    - **Updated At** - should use the format YYYY-MM-DD
    - **Total Quantity of Products** - the total quantity of ordered products.
    - **Total Count of Products** - a count of unique product titles in the ordered quote.
    - **Updated At Time**
- **Follow-Up Email**
    - **Recipiend has Follow-Up Emails
- **Customer** contains the customer's properties, such as **Name**, **Email**, **Gender** and so on.
- **Shipping Address** contains the address properties, such as **Country**, **City**, **Street**, and so on.

After configuring the Event and Audience, you can **add email to the chain** and **test the new trigger**.

- **(DEPRECATED) Administrator Trigger**

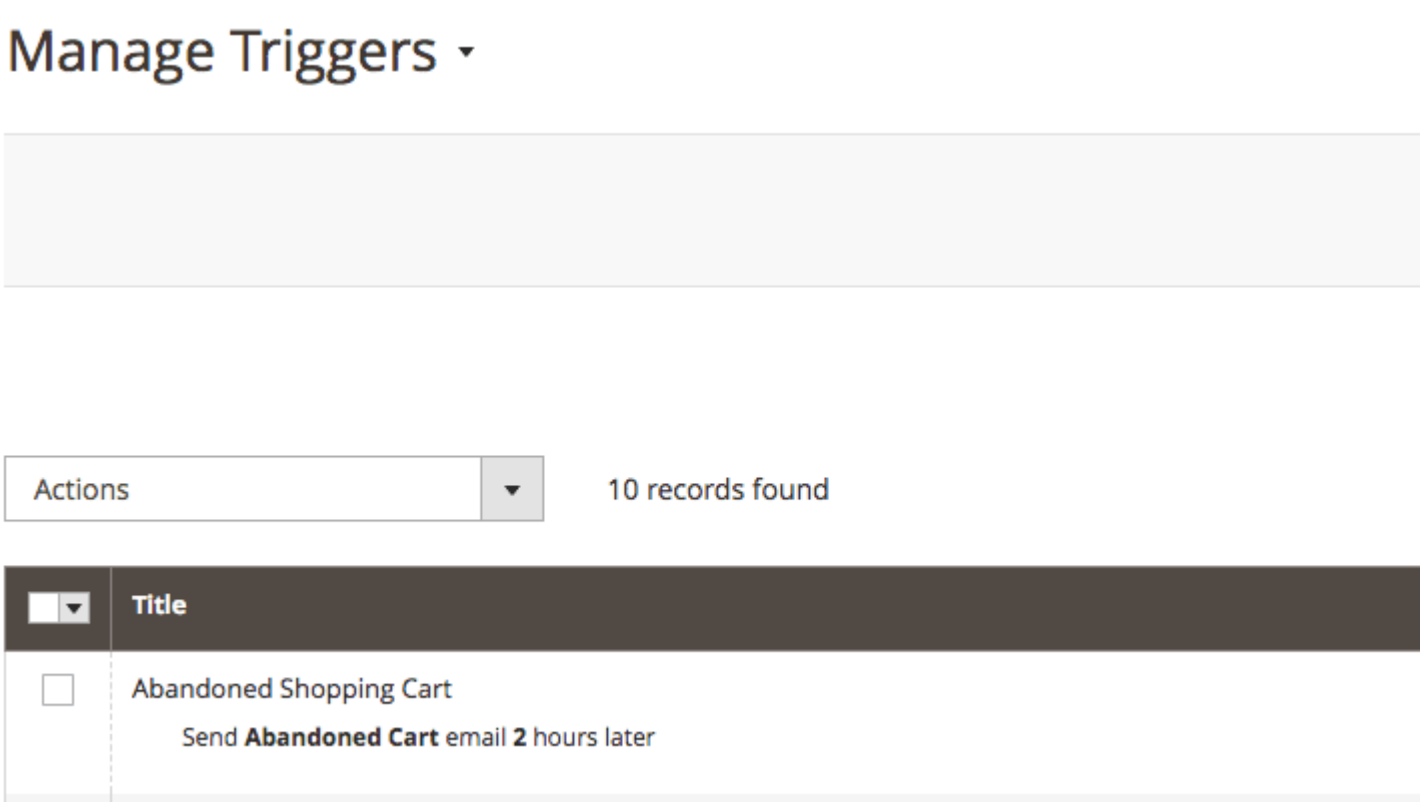
  In **older versions** in addition to simple triggers, there were also *Administrator Triggers*.

  The benefit of the Administrator Trigger is that you can use it to send an email to yourself when a particular event is triggered in your store, meaning that it serves just like a notifier or reminder about certain events which occurred in your store.

  Here are some examples of situations when you may want to send a message to yourself:

    - someone places an order for a large quantity
    - a customer leaves a review for your products
    - a new customer from a specific country is registered
    - a customer adds some specific product to the wishlist
    - customer's lifetime sales exceed some designated level, ensuring that you remind yourself to contact them individually

  If you still use the old version, you can create an Administrator Trigger with **Add New Administrator Trigger** at the Trigger edit page.



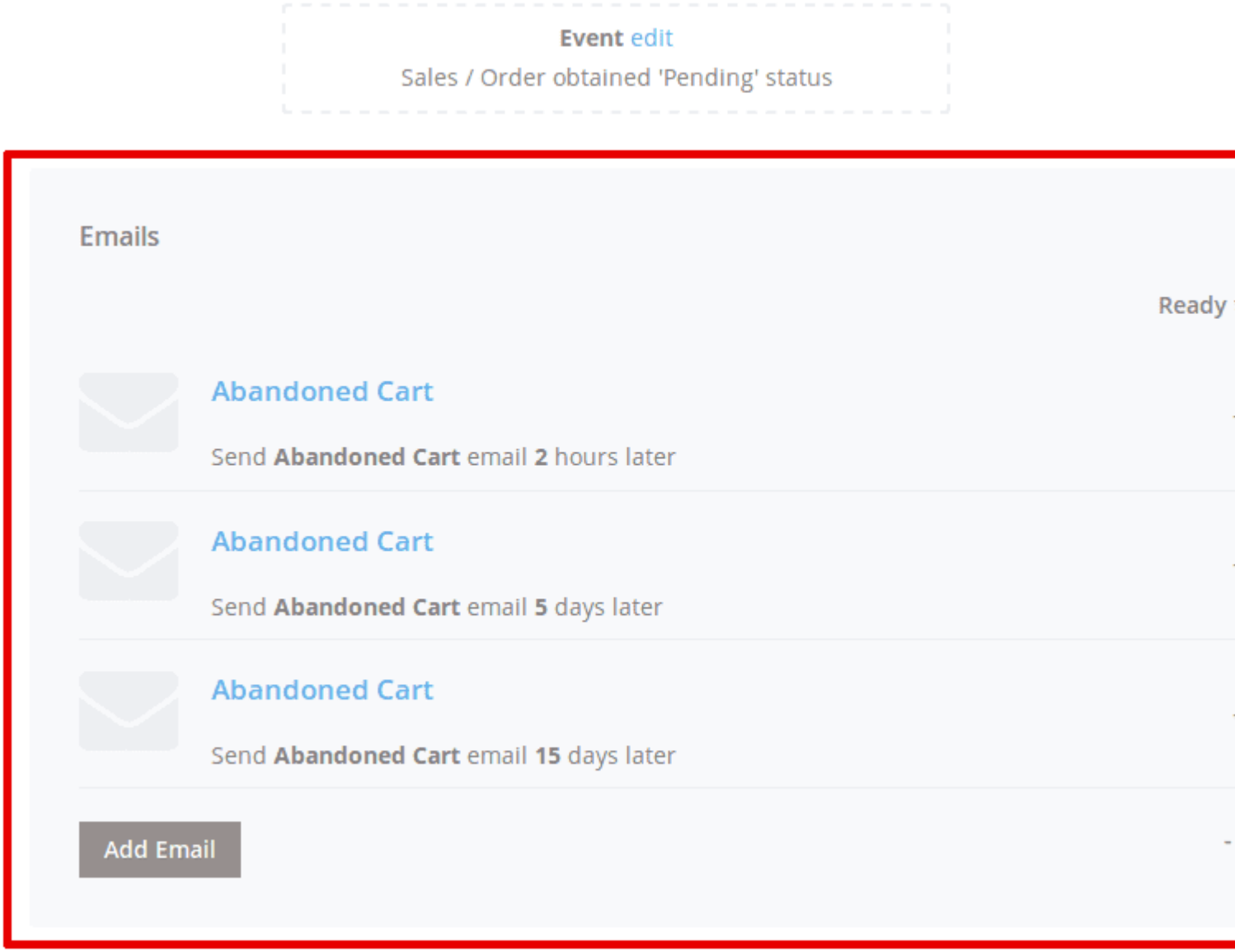
After that, you should complete one additional field in the trigger's settings:

- **Recipient Email** - specify target email addresses, separate e-mails by commas.

Manage Email Chain

You can create an unlimited number of emails that will be sent after event triggering.

All of them will be displayed on the **Emails** sub pane at the Trigger pane of the Campaign edit page:



Press **Add New Email** to add a new email to your Trigger, and you will see an Email adding dialog with the following properties:

- **General**
  - **Email Template** - selects a template, which will be used for emails. Templates for the emails can be defined using **Email Designer**.
  - **Delivery Time Delay** - the delay after which a triggered message will be sent. It can be set in **days** , **hours** and **minutes**.

    Note
    By default, it will send immediately after a triggering event (0 days 0 hours 0 minutes). You can choose times from as short as one minute to as long as a few years after the trigger's criteria is met.

  - **Excluded Weekdays** - allows you to select the days of the week when emails won't be sent. Typically, it is on Saturday and Sunday.
- **Coupons** - used when an email template supports coupons.
  - **Include coupon in email** - turns on/off coupon sending.
  - **Shopping Cart Price Rule** - select shopping cart price rule, dependent on the size of the discount
  - **Coupon expires after __ days** - fill in the time of the coupon's expiration

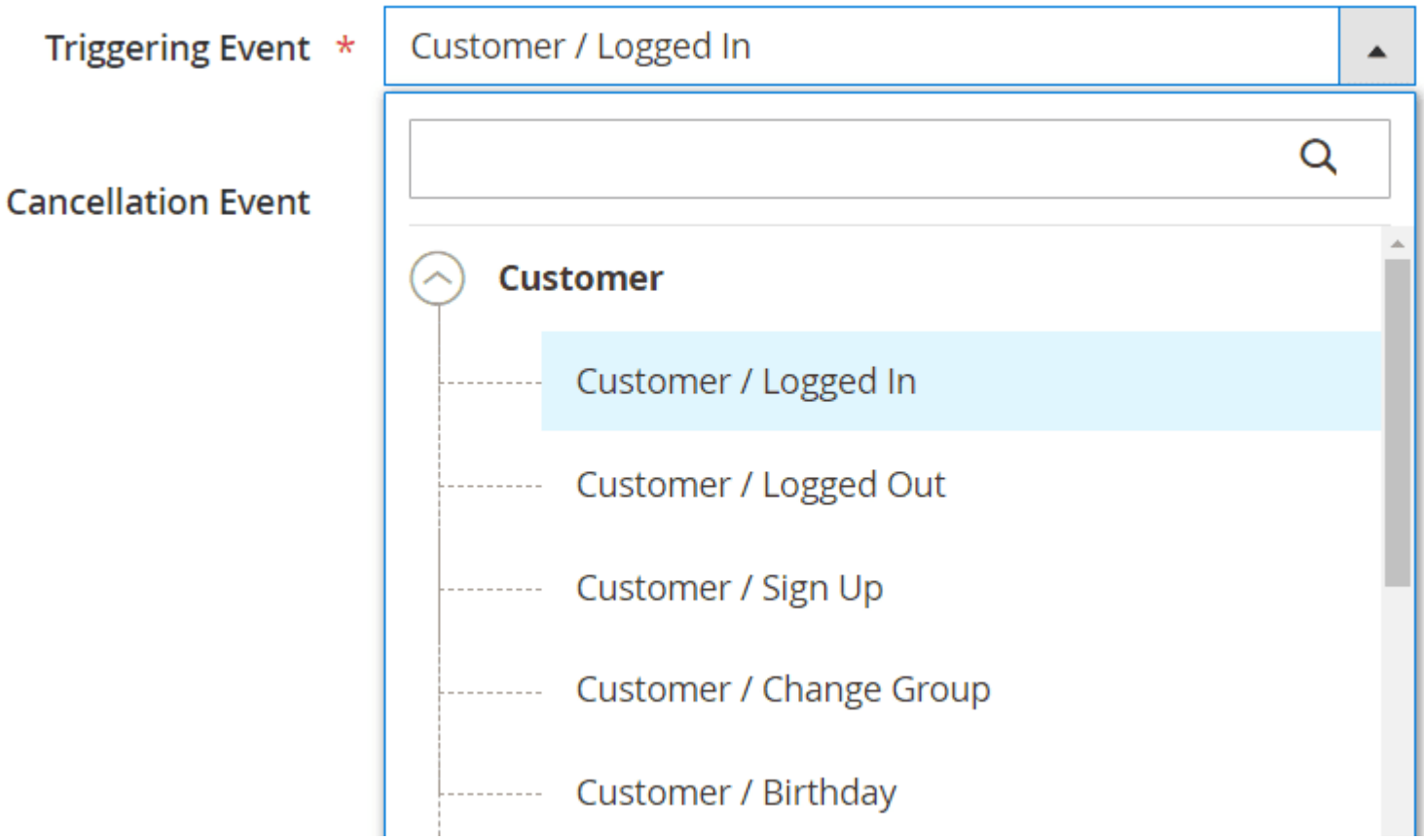
Note

Our extension can generate coupon codes **only** via the Shopping Cart Rule. Make sure that the option **Use Auto Generation** is **enabled** from the selected rule. Otherwise, coupons won't be generated.

- **Cross-sells** - used when an email template supports cross-sells products
  - **Include cross-sells in email**
  - **Cross-sells source** - select a cross-sells block whose products are added to the email
    1. Cross-sell products
    2. Related products
    3. Upsell products

List of Events

Customer Events



- Customer Logged In
- Customer Logged out
- New Customer Sign up

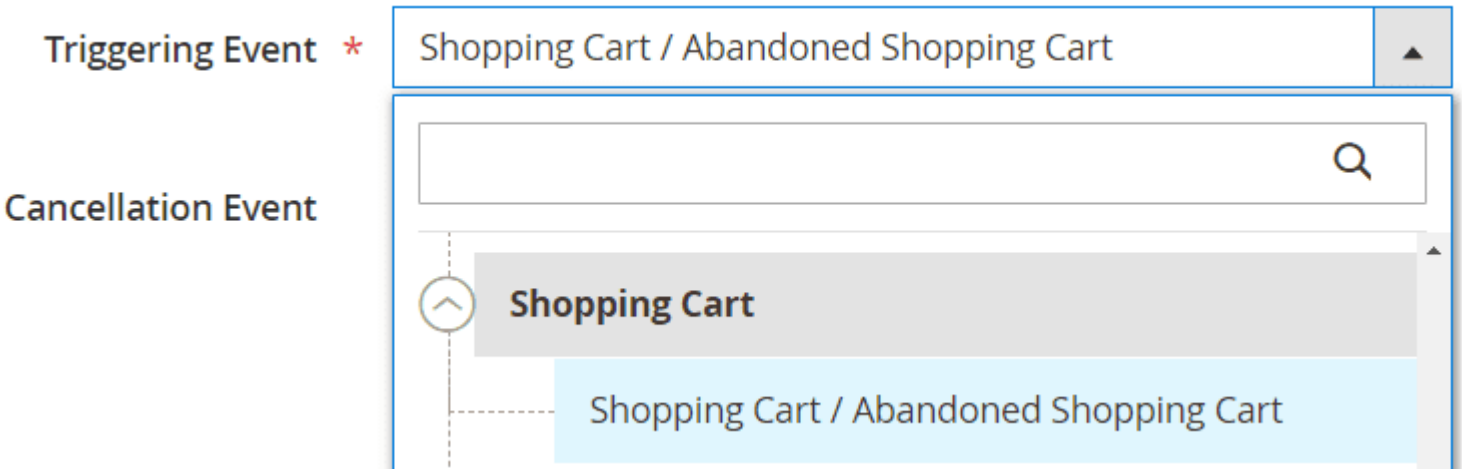
- Customer Birthday

- Newsletter subscription

  By default, Magento sends its own **Success Email Template** after a new client has subscribed to a newsletter.
  If you want to use our module for this type of email, you need to disable the standard Magento template. To disable it, navigate to **Stores** > **Settings** > **Configuration** > **Customers** > **Newsletter**, and selected the option **Disable** for the **Success Email Template** field.

- Newsletter Unsubscription

Shopping Cart

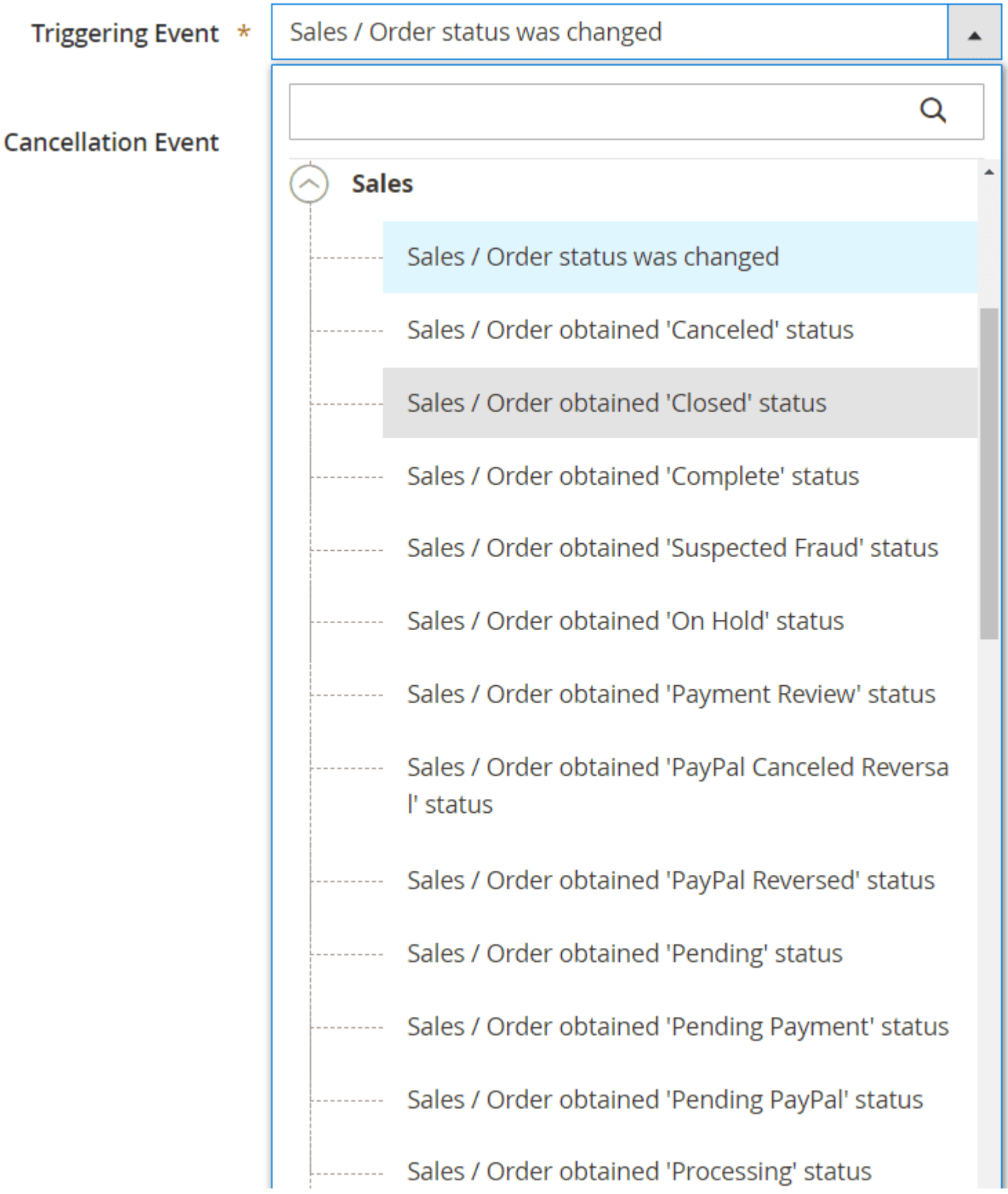


- Abandoned Shopping Cart

  The event is triggered when the shopping cart is not updated in the last **60 minutes**.

- Product price was changed

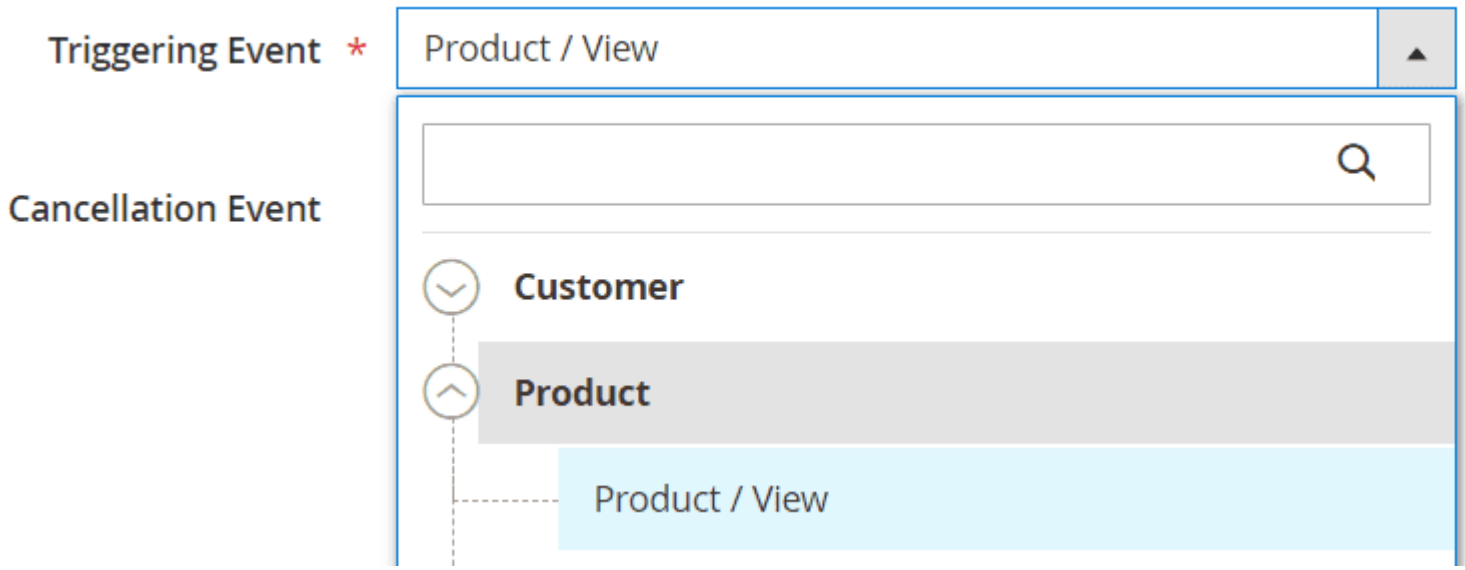
Order Events



- Order obtained a new status
- Order obtained '###' status

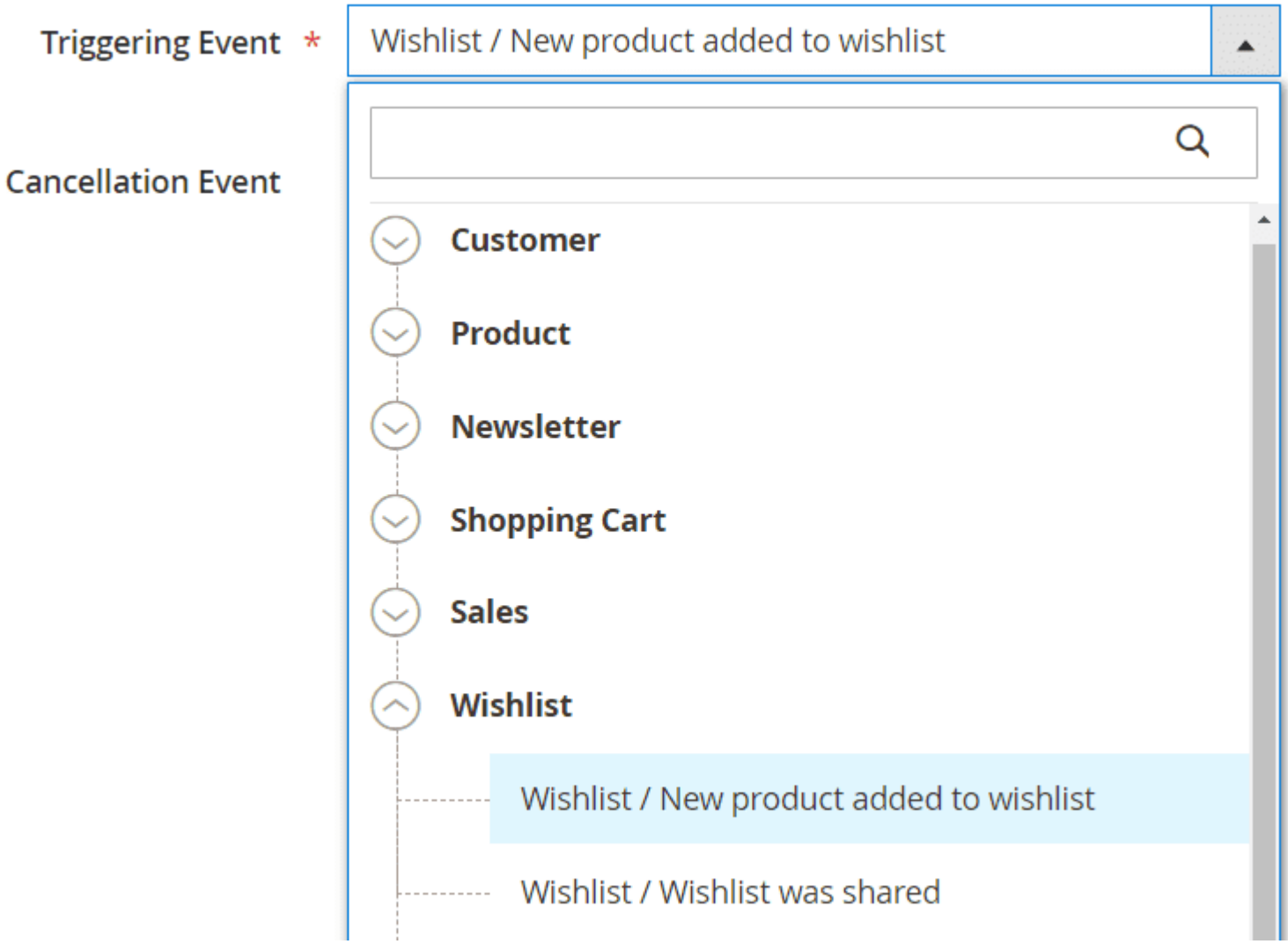
- Order obtained 'Pending' status
- Order obtained 'Processing' status
- Order obtained 'Completed' status
- ...

Product Events



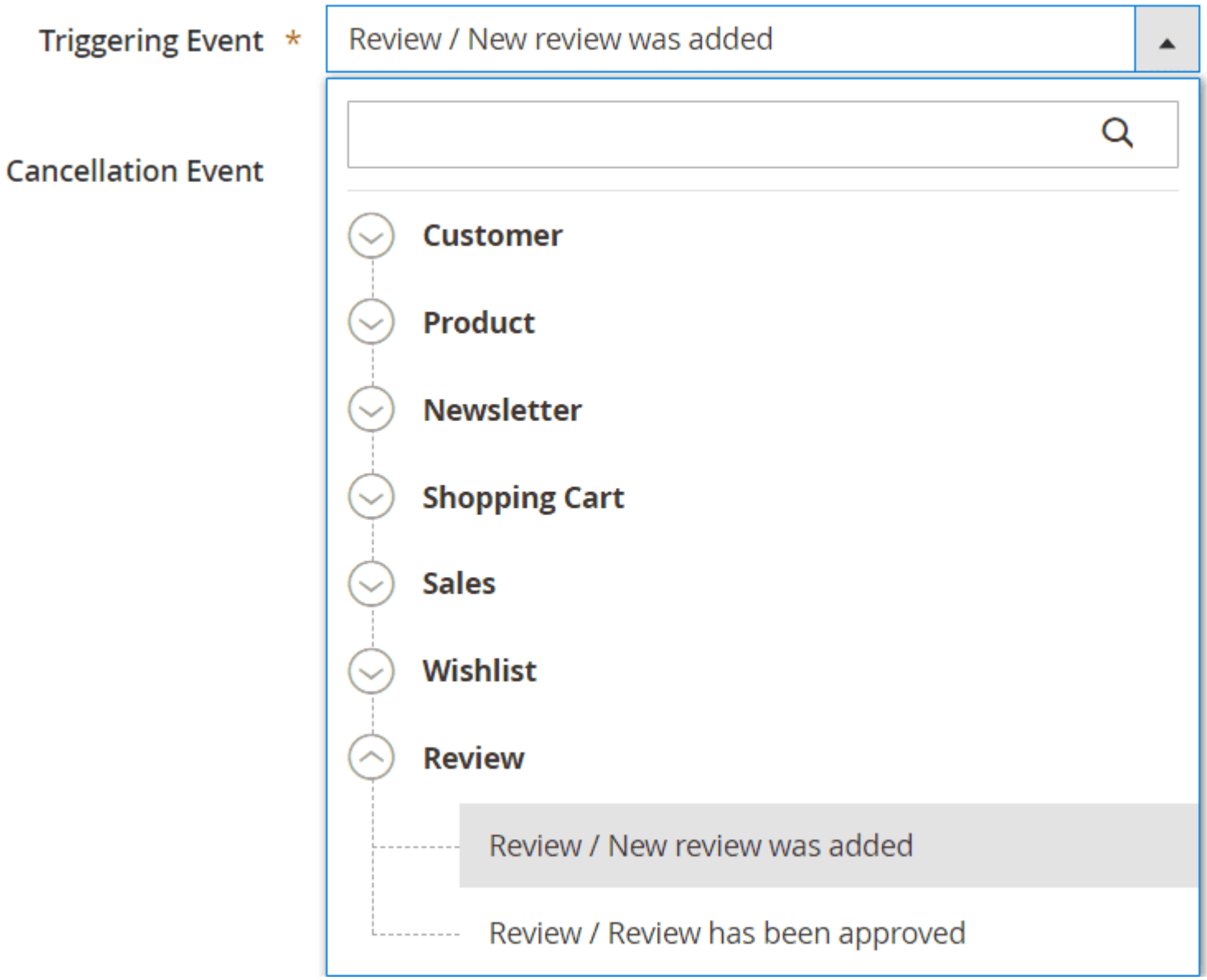
- Product view

Wishlist Events



- Product was added to wishlist
- Wishlist was shared

Review Events

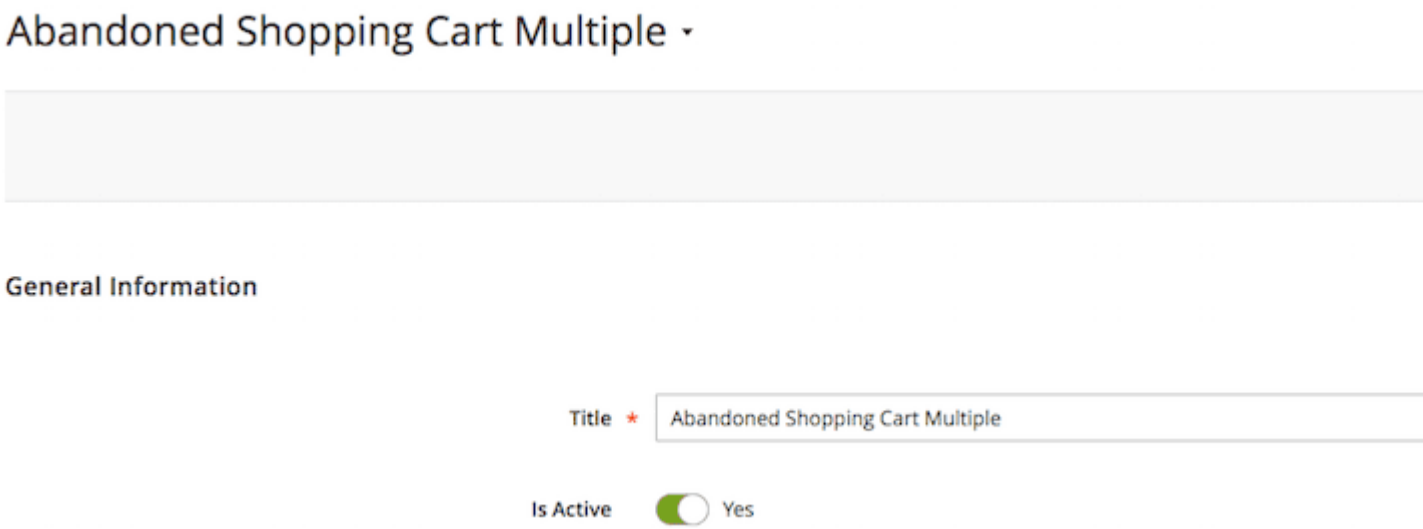


- New review was added
- Review has been approved

Testing Campaigns

Our extension allows you to test your campaign without starting a real one. Just take the following steps:

- Go to **Marketing -> Follow Up Email -> Manage Campaigns** and open the campaign for editing, or directly press the Trigger access button.
- Select the template which you would like to test, and open it in order to edit.
- On the top actions ribbon, you will find a **Send Test Email** button. Press it to start testing.
- Enter your staging email address and press **Send**.



- The Extension then will send an email as if it were part of an email chain.

Tip

You can boost testing using **Sandbox mode**, which can be turned on at **Stores -> Configuration -> Follow Up Email -> Test Information**.

In this mode, sending emails to actual customers will be suppressed. Instead, all of them will flow directly to the mailbox, set in the **Test Recipient Email** setting in the section above.

Note
For test emails, the extension will generate test data based on current customers.

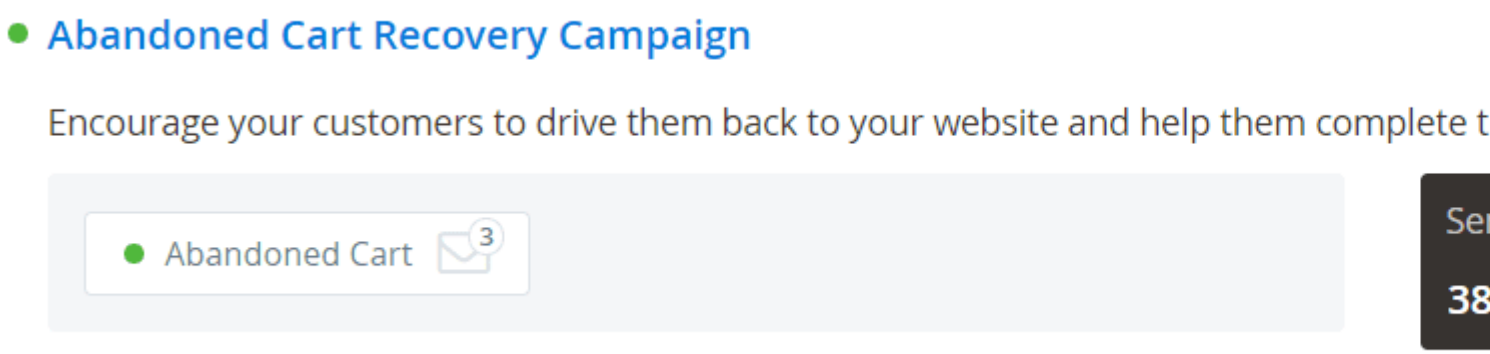
Examples of Campaign and Triggers Configurations

Our application comes with a set of handy examples, which can easily be adjusted to most cases of promotional plans. They are:

- **Abandoned Cart Recovery Campaign**
- **Customer Engagement Campaign**
- **Customer Re-Engagement Campaign**
- **Sales Follow-Up Campaign**

Let's delve into their details:

Abandoned Cart Recovery Campaign



The Abandoned Cart campaign is meant to remind customers of their unfinished shopping and encourage them to either complete their purchase or continue shopping for new products.

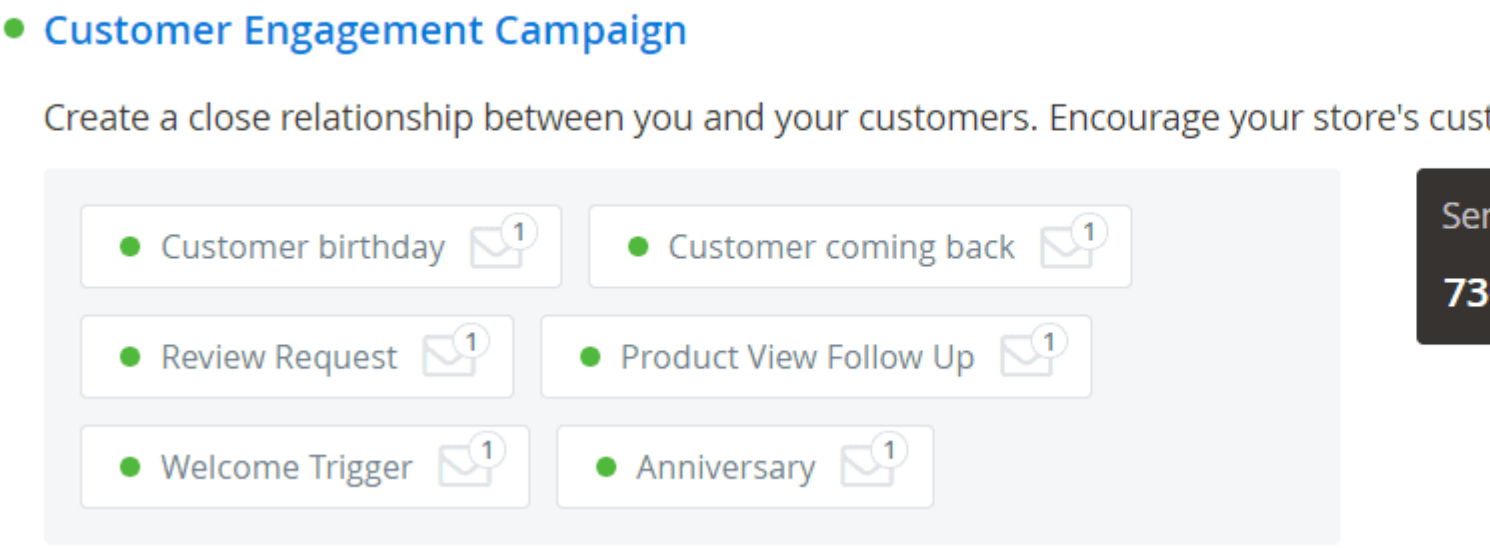
Typically the campaign consists of one trigger with three emails:

- **Trigger: Abandoned Cart**
  - **Event:**
    - **Triggering Event:** `Shopping Cart / Abandoned Shopping Cart`
    - **Cancellation Event:** `Sales / Order obtained Pending status`
  - **Audience:** `Quote: Total count of products greater than 0`

Email Chain:

- **Abandoned Cart (2 hours)**
  - Email Template: `Abandoned Cart`
  - Delivery Time Delay: `2 hours`
- **Abandoned Cart (5 days)**
  - Email Template: `Abandoned Cart`
  - Delivery Time Delay: `5 days`
- **Abandoned Cart (15 days)**
  - Email Template: `Abandoned Cart`
  - Delivery Time Delay: `15 days`

Customer Engagement Campaign



Engagement Campaign is the most complex of all campaigns. It consists of the most used triggers that can attract customers to your store and create a close relationship between you and your customers.

- **Trigger: Customer birthday**

  This trigger sends an email to the customer on his/her birthday. Since the corresponding event occurs at 00:00, we need an extra 10-hours delay.

  - **Event:**
    - **Triggering Event:** `Customer / Birthday`
    - **Cancellation Event:** `none`
  - **Audience:** `not set` (means all customers are eligible)

  Email Chain:

  - **Happy Birthday**
    - Email Template: `Happy Birthday`
    - Delivery Time Delay: `10 hours`

- **Trigger: Customer coming back**

  This trigger is used when a customer hasn't visited the store for over a year, and therefore, should receive information about our newest picks.

  - **Event:**
    - **Triggering Event:** `Customer / Logged In`
    - **Cancellation Event:** `none`
  - **Audience:** `Customer: Last activity (in days) equals or greater than 10`

  Email Chain:

  - **Customer coming back**
    - Email Template: `Customer coming back`
    - **Cross-sells**:
      - Include cross-sells in email: `Yes`

        - Cross-sells source: `Cross-sell products`

- **Trigger: Review Request**

  This trigger watches which products a customer has bought, and automatically requests that the customer give a product review if they have not returned a product for ten days (approx. period of return).

  - **Event:**
    - **Triggering Event:** `Sales / Order obtained 'Complete' status`
    - **Cancellation Event:** `none`
  - **Audience:** `not set`

  Email Chain:

  - **Review Request**
    - Email Template: `Review Request`
    - Delivery Time Delay: `10 days`
    - **Cross-sells**:
      - Include cross-sells in email: `Yes`
      - Cross-sells source: `Cross-sell products`

- **Trigger: Product View Follow Up**

  This trigger monitors which products a customer has viewed recently and automatically suggest similar products.

  - **Event:**
    - **Triggering Event:** `Product / View`
    - **Cancellation Event:** `Sales / Order obtained Pending status`
  - **Audience:**
    - `Recipient does not have emails with Ready to Go status in the Mail Log for the trigger(s) Anniversary within last 7 days`
    - `Product is one of 20 top selling products is Yes`

  Email Chain:

  - **Recently viewed product**
    - Email Template: `Recently Viewed Products`
    - Delivery Time Delay: `3 hours`
    - **Cross-sells**:
      - Include cross-sells in email: `Yes`
      - Cross-sells source: `Related products`

- **Trigger: Welcome Trigger**

  This is the most basic trigger. It activates after customer registration and sends a welcome message and a list of the latest picks.

  - **Event:**
    - **Triggering Event:** `Customer / Sign Up`
    - **Cancellation Event:** `none`
  - **Audience:** `not set`

  Email Chain:

  - **Welcome**

        - Email Template: `Welcome`
        - **Cross-sells**:
            - Include cross-sells in email: `Yes`
            - Cross-sells source: `Cross-sell products`

- **Trigger: Anniversary**

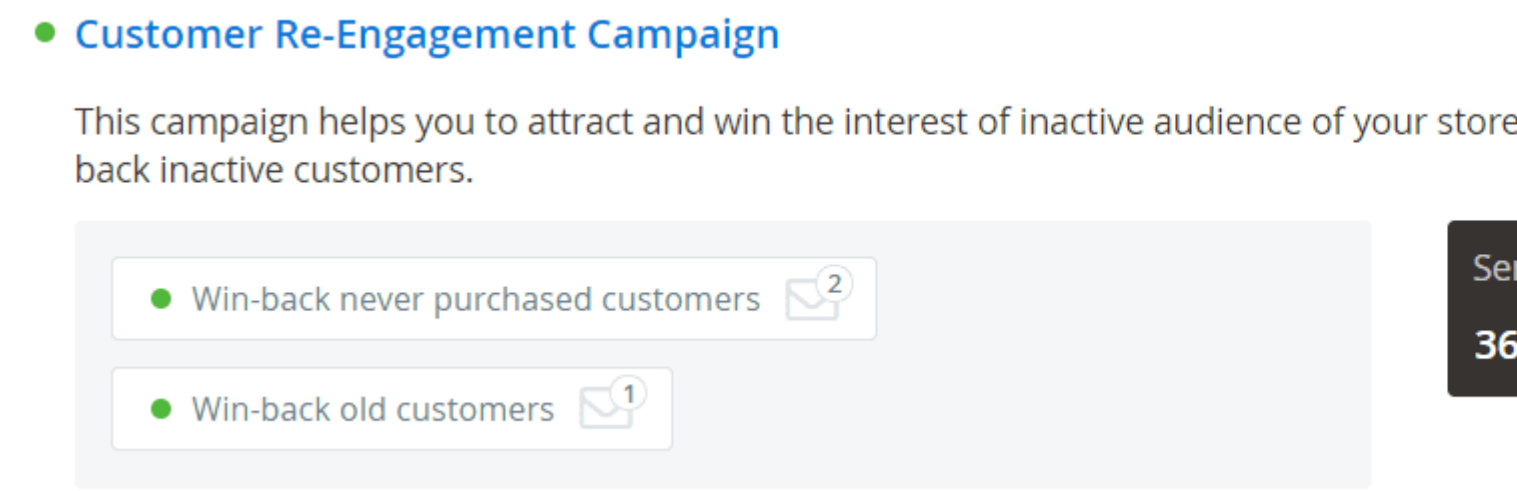
  This trigger allows you to create anniversary event for your customer, celebrating their yearly anniversary of using your store.

    - **Event:**
        - **Triggering Event:** `Customer / Sign Up`
        - **Cancellation Event:** `none`
    - **Audience:** `not set`

  Email Chain:

    - **Anniversary**
        - Email Template: `Anniversary`
        - Delivery Time Delay: `365 days`

Customer Re-Engagement Campaign



This campaign consists of triggers that should help you attract customers to return, namely those who had purchased items at your site in the past, but were not convinced to stay.

- **Trigger: Win-back never purchased customers**

  This trigger monitors customers who had registered, but never created a cart or purchased a product. After 30 days, we will send them an email with our newest picks.

    - **Event:**
        - **Triggering Event:** `Customer / Sign Up`
        - **Cancellation Event:** `Sales / Order obtained Pending status`
    - **Audience:** `not set` (means all customers are eligible)

Email Chain:

- **We miss you**
  - Email Template: `We miss you`
  - Delivery Time Delay: `30 days`
  - **Cross-sells**:
    - Include cross-sells in email: `Yes`
    - Cross-sells source: `Cross-sell products`
- **We miss you 2**
  - Email Template: `We miss you 2`
  - Delivery Time Delay: `60 days`
  - **Cross-sells**:
    - Include cross-sells in email: `Yes`
    - Cross-sells source: `Related products`

- **Trigger: Win-back old customers**

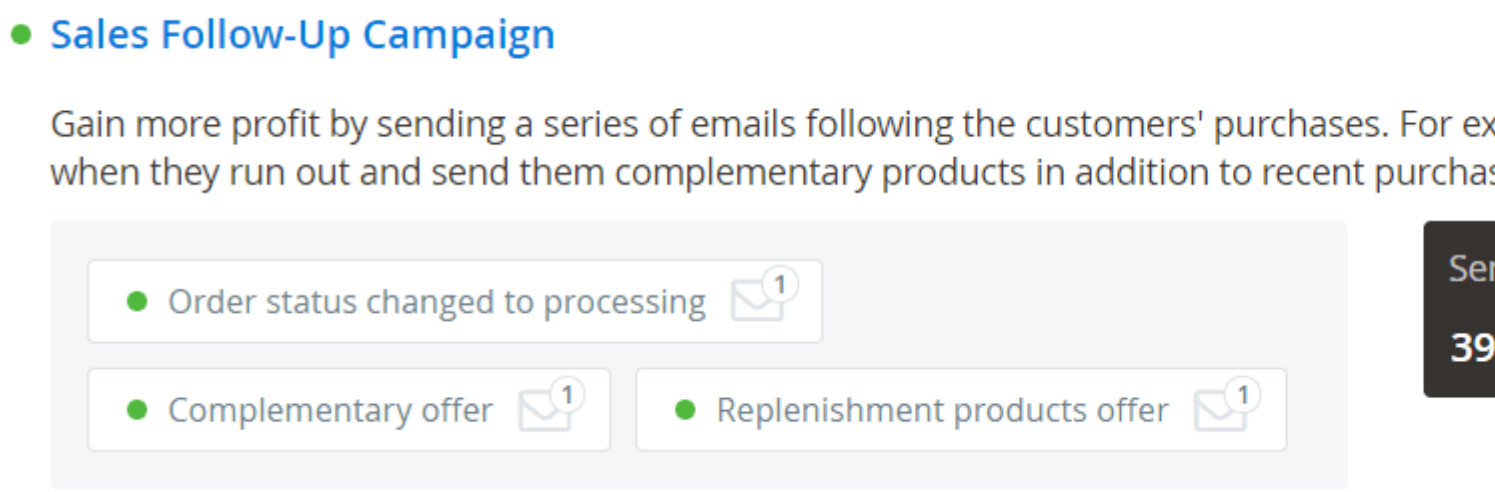
  This trigger is used when a customer purchased something but hasn't visited the store for two months. Therefore, they should receive a friendly reminder and information about our newest picks.

  - **Event:**
    - **Triggering Event:** `Sales / Order obtained 'Complete' status`
    - **Cancellation Event:** `Sales / Order obtained Pending status`
  - **Audience:** `not set`

  Email Chain:

  - **We miss you 2**
    - Email Template: `We miss you 2`
    - Delivery Time Delay: `60 days`
    - **Cross-sells**:
      - Include cross-sells in email: `Yes`
      - Cross-sells source: `Related products`

Sales Follow-Up Campaign



This campaign was created to suggest customers check out other products that are connected with their purchases.

- **Trigger: Order status changed to processing**

  This trigger is activated when a customer had placed an order, and it is processed. Therefore, the customer might be interested in purchasing an additional related item.

  - **Event:**
    - **Triggering Event:** `Sales / Order obtained 'Processing' status`
    - **Cancellation Event:** `none`
  - **Audience:** `not set` (means all customers are eligible)

  Email Chain:

  - **Order status changed**
    - Email Template: `Order status changed`
    - **Cross-sells**:
      - Include cross-sells in email: `Yes`
      - Cross-sells source: `Cross-sell products`

- **Trigger: Complementary offer**

  This trigger is used when a customer has placed an order, and it was processed two days prior, presuming it's still fresh in their mind. This means that we should suggest they consider buying another complementary item from the store.

  - **Event:**
    - **Triggering Event:** `Sales / Order obtained 'Pending' status`
    - **Cancellation Event:** `none`
  - **Audience:** `not set`

  Email Chain:

  - **Complementary offer**
    - Email Template: `Complementary offer: related products`
    - Delivery Time Delay: `5 days`
    - **Cross-sells**:
      - Include cross-sells in email: `Yes`
      - Cross-sells source: `Related products`

- **Trigger: Replenishment products offer**

  This trigger is used when a customer purchased something, and we would like to suggest they consider buying related accessories or replaceable components (batteries, for example).

  - **Event:**
    - **Triggering Event:** `Sales / Order obtained 'Complete' status`
    - **Cancellation Event:** `none`
  - **Audience:** `not set`

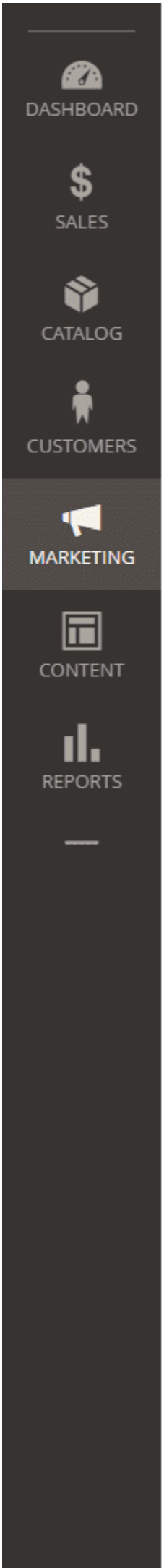
  Email Chain:

  - **Replenishment products**

- Email Template: `Replenishment products`
- Delivery Time Delay: `30 days`
- **Cross-sells**:
  - Include cross-sells in email: `Yes`
  - Cross-sells source: `Upsell products`

Managing Themes

Themes are the most generic templates that allow you to have all general elements of your email design (such as headers, styles, and footers) in one place. When designing a template, you can just select a theme and specify the message - the rest will be done by our extension.



Base Theme

Name * Base Theme

Type * HTML

Template Text

```
{{template config_path="design/email/header_template"}

<table width="100%">
  {% if 'header' | area %}
  <tr class="email-intro">
    <td>
      {{ 'header' | area: 'Email header here...' }}
    </td>
  </tr>
  {% endif %}
  {% if 'content' | area %}
  <tr class="email-intro">
    <td>
      {{ 'content' | area: 'Email text here...' }}
    </td>
  </tr>
  {% endif %}
  {% if 'footer' | area %}
  <tr class="email-intro">
    <td>
      {{ 'footer' | area: 'Email footer here...' }}
    </td>
```

To create a new theme, visit **Marketing -> Follow Up Email -> Email Designer -> Manage Themes** and press the **Add Theme** button.

The theme consists of the following properties:

- **Name** - name of the template.
- **Type** - template type. There's two available types: **HTML** (default) and **Text**.
- **Template** - the basic design template, which will use all subsequent templates.

The Template covers all email areas - including headers, footers and even the main content. These editable areas are defined using special liquid variables:

- `{{ 'header' | area: 'Email header here...' }}` - places a header area
- `{{ 'content' | area: 'Email text here...' }}` - places a main body area
- `{{ 'footer' | area: 'Email footer here...' }}` - places a footer area

Example

Since the template does not have to define all of the areas, we recommend that you use this construction - it will insert an area only if it is defined:

```
{% if 'content' | area %}
    {{ 'content' | area: 'Email text here...' }}
{% endif %}
```

If you prefer to use callout-style, use the `<?php echo $this->area('header') ?>` directive instead.

You can also use Variables and Callouts in your design.

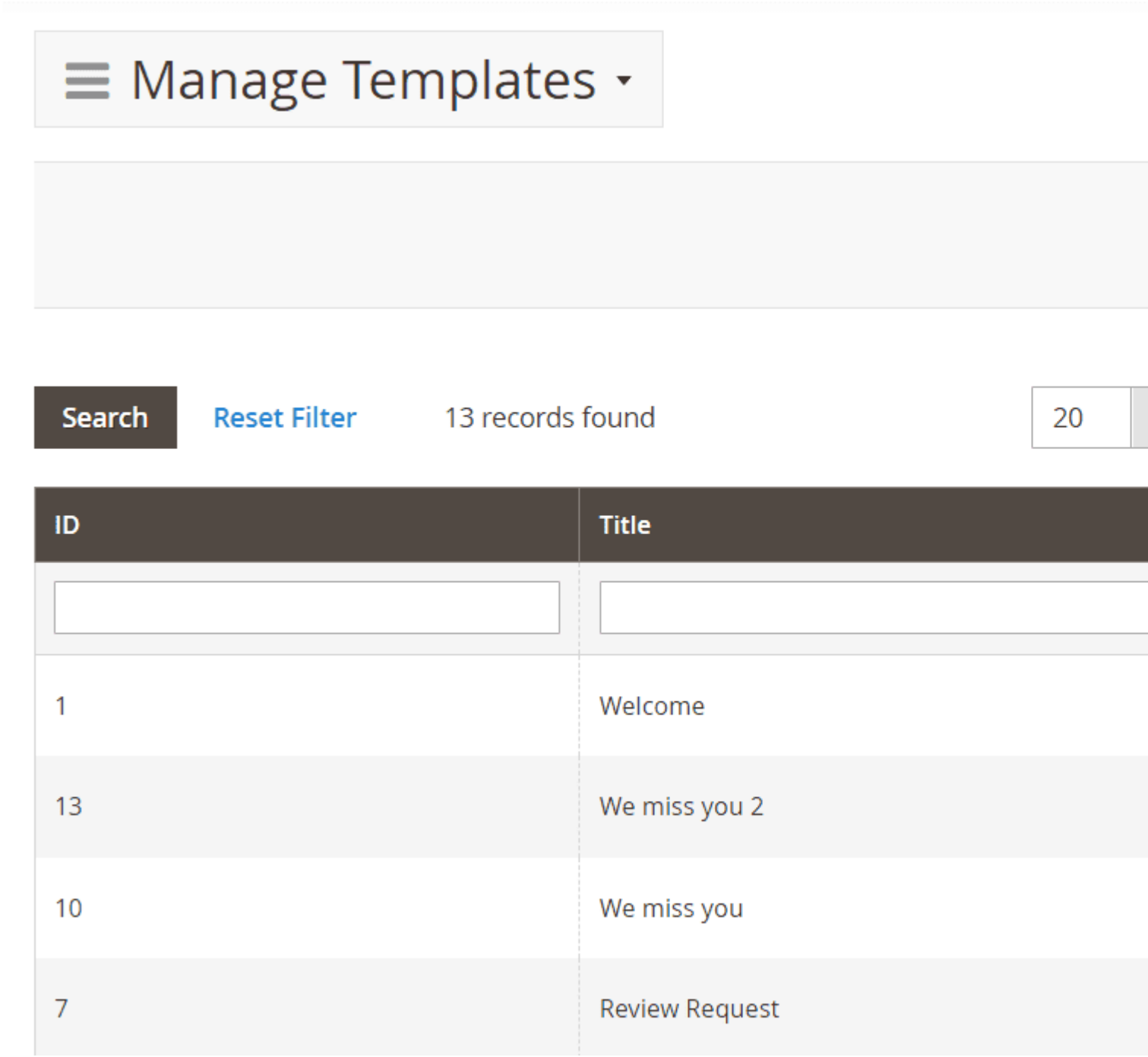
All these parts can be previewed on the Preview Pane in both **Desktop** and **Mobile** versions.

Tip

You can also change the template area, and have an instant preview without saving. For that, you need to set the **Auto-Refresh** checkbox at Preview Pane, or press the **Refresh** button.

Save the theme and proceed with **Templates** creation.

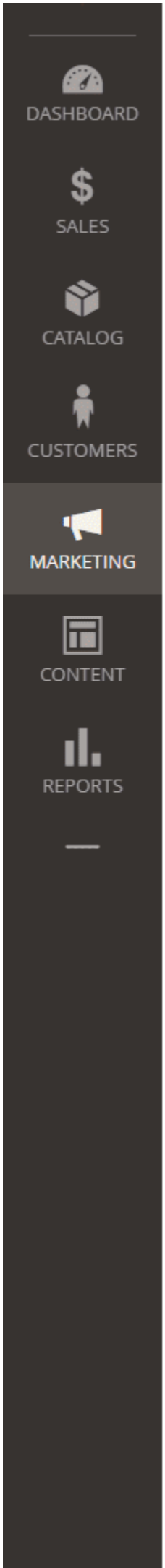
Managing Templates



All templates used in Follow-Up Emails are located at their respective Grid at **Marketing -> Email Designer -> Manage Templates**.

Actions performed on templates can be selected from the **Actions** drop-down. Here you can **Edit** or **Remove** Template.

How to Create New Template



Welcome

Name * Welcome

Theme * Base Theme

Subject * Welcome to {{ store.store_name }}!

Header

```
1 <h1>Welcome {{ customer.firstname }}!</h1>
```

Insert Variable

Content

```
1 <p>Thank you for joining <a href="{{ store.store_url }
```

Insert Variable

Footer

```
1 {% if coupon.code %}
2 <div class="offer">
3   To show our appreciation, we are giving you a <b>10%
4 </div>
```

Go to **Marketing -> Email Designed -> Manage Templates**, and press the **Add Template** button. It will bring you to the first stage of Template creation, with the following basic properties:

- **Name** - name of the template
- **Theme** - one of the themes, defined at **Marketing -> Email Designed -> Manage Themes**. Read more in the **Themes** section.
- **Subject** - the default subject which will be used in emails, based on this template.

Saving a template at this stage **does not** actually create a template, but a **draft stub**. After it is saved, you need to reopen it using the **Edit** action, and enter contents.

The Template Edit Page breaks into two panels - left (**Areas Pane**) and right (**Preview Pane**).

Areas are parts of your email. They depend on your store's design, and your desired look & feel, but should consist of three areas:

- **Header** - will be displayed at the very top of your email.
- **Content** - is the main body of your template. Here you can enter your message and other details.
- **Footer** - will be displayed at the very bottom of your email.

You can use **Variables** in any part of template. The **Insert Variables** button will help you to pick up the desired variable interactively.

All of these parts can be previewed on the Preview Pane in both **Desktop** and **Mobile** versions.

Tip

You can also change the template area, and have an instant preview without saving. For that, you need to set the **Auto-Refresh** checkbox at the Preview Pane, or press the **Refresh** button.

Example

The templates used the basic pattern to show all products from the quote/order in an email :

```
{% for item in this.all_visible_items %}

---

{% endfor %}
```

To send only the first active product in an email, you need to modify this pattern:

```
{% for item in this.first_visible_item %}

---

{% endfor %}
```

Variables & Methods

Our extension allows you to use variables and PHP callouts in your emails, which can greatly enhance and personalize them.

Note

Both variables and callouts can be used simultaneously, and are fully interchangeable so you can select your preferred syntax, and use it for all of your customizations.

Liquid is **the preferred syntax** for use with Email Templates. It allows you to avoid errors with the absence of attributes or values. Moreover, we provide a convenient helper dialog for inserting liquid variables into the editor.

Callouts are **deprecated** syntax, and used mainly for backward compatibility with older Follow-Up Email versions.

Liquid Variables

Liquid variables are a new way to enhance email templates. This syntax was introduced in version **1.1.15** and is the preferred syntax for use in Email Templates.

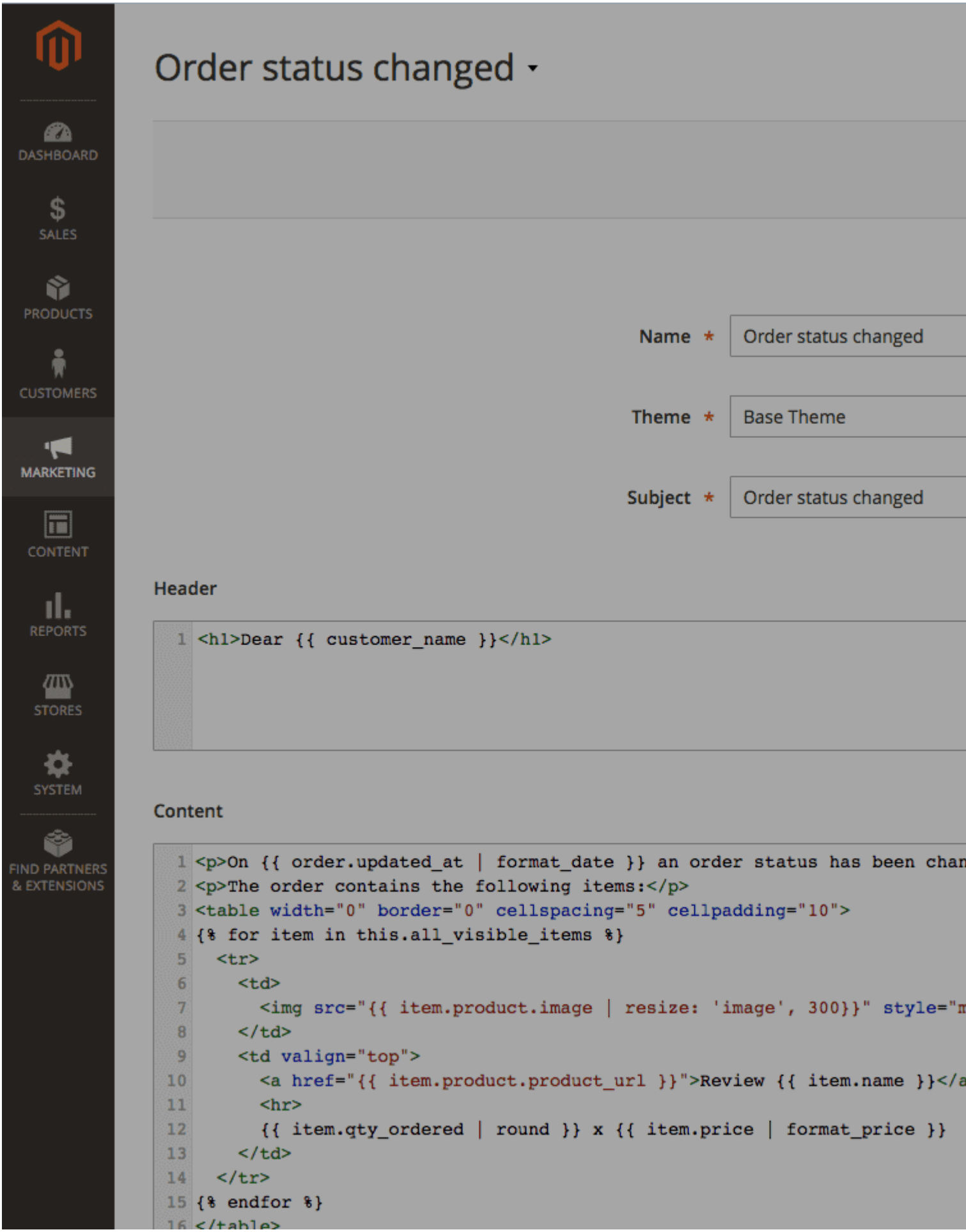
All variables should be enclosed in curly brackets. Each variable can also have a filter, added after pipe sign, and have one or more parameters.

Example

1. `{{ attribute | filter }}`
2. `{{ entity.attribute | filter | filter: param1 }}`
3. `{{ entity.entity.attribute | filter: param1,param2 }}`

These variables can be added interactively from edit pages of **Theme** or **Template**.

You just need to press the button **Insert Variable** near the content element, and select the variable you want to use - as shown below:


DASHBOARD
SALES
PRODUCTS
CUSTOMERS
MARKETING
CONTENT
REPORTS
STORES
SYSTEM
FIND PARTNERS & EXTENSIONS
Order status changed
Name * Order status changed
Theme * Base Theme
Subject * Order status changed
Header
1 <h1>Dear {{ customer_name }}</h1>
Content
1 On {{ order.updated_at | format_date }} an order status has been chan
2 The order contains the following items:
3 <table width="0" border="0" cellspacing="5" cellpadding="10">
4 {% for item in this.all_visible_items %}
5 <tr>
6 <td>
7 <img src="{{ item.product.image | resize: 'image', 300}}" style="m
8 </td>
9 <td valign="top">
10 <a href="{{ item.product.product_url }}">Review {{ item.name }}</a
11 <hr>
12 {{ item.qty_ordered | round }} x {{ item.price | format_price }}
13 </td>
14 </tr>
15 {% endfor %}
16 </table>

Modifying Variables with Filters

Filters are methods that allow you to alter or enhance the output of a variable. They also should be enclosed to the variable block `{{ }}`, but separated from the variable with a pipe (`|`) character. The parameters of filters are added using the colon (`:`) character.

Example

- `{{ item.product.name | truncate: '150' }}` - truncates the name of the product to 150 characters/
- `{{ item.product.weight | round: '2' }} kg` - rounds the weight of the product to 2 decimal digits

Here is the list of available filters, grouped into categories, with examples:

- **String/HTML Filters**
  - `downcase` - converts a string into lowercase.

    ```
    {{ item.product.name | downcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `dash digital watch`
  - `upcase` - converts a string into uppercase.

    ```
    {{ item.product.name | upcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `DASH DIGITAL WATCH`
  - `replace` - replaces all occurrences of a string with a substring.

    ```
    {{ item.product.name | replace: 'Digital', 'Analog' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Analog watch`
  - `append` - appends characters to a string.

    ```
    {{ item.product.name | append: ' - best choice' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Digital Watch - best choice`
  - `prepend` - prepends characters to a string.

    ```
    {{ item.product.name | prepend: 'Best choice - ' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Best choice - Dash Digital Watch`

- `capitalize` - capitalizes words in the input sentence.

  ```
  {{ item.product.color | capitalize }}
  ```

  - **Original**: `dark red`
  - **Output**: `Dark red`

- `escape` - escapes HTML tags in a string.

  ```
  {{ item.product.description | escape }
  ```

- `newline_to_br` - inserts a `<br>` linebreak HTML tag in front of each line break in a string.

  ```
  {{ item.product.short_description | newline_to_br }
  ```

- `remove` - removes all occurrences of a substring from a string.

  ```
  {{ item.product.name | remove: 'Digital' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `Dash Watch`

- `strip_html` - strips all HTML tags from a string.

  ```
  {{ item.product.description | strip_html }
  ```

- `truncate` - truncates a string down to 'x' characters.

  ```
  {{ item.product.name | truncate: '15' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `ash Digital Wa`

- `if_empty` - return argument, if the value is an empty string

  ```
  Dear {{ customer_name | if_empty: 'Client' }}!
  ```

  - **Original**: `empty string`
  - **Output**: `Dear Client!`

- `date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | date: '%d.%m.%Y %H:%M' }}
  ```

  - **Original**: `2016-02-18 10:11:12`
  - **Output**: `18.02.2016 10:11`

  Full list of formatters can be found **here**

- `format_date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | format_date: 3 }}
  ```

- **Original**: `2016-02-18 10:11:12`
- **Output**: `18/02/16`

Possible formatters: `0, 1 , 2 , 3`

- **Numeric Filters**

  - `ceil` - rounds the output up to the nearest integer.

    ```
    {{ item.product.weight | ceil }}
    ```

    - **Original**: `1.423`
    - **Output**: `2`

  - `floor` - rounds the output down to the nearest integer.

    ```
    {{ item.product.weight | floor }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `round` - rounds the output to the nearest integer or specified decimal digits.

    ```
    {{ item.product.weight | round: '2' }}
    ```

    - **Original**: `1.423`
    - **Output**: `1.42`

    ```
    {{ item.product.weight | round }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `number_format` - formats number to specified format (php function).

    ```
    {{ item.product.price | number_format: '2', '.', ',' }}
    ```

- **Price/Currency Filters**

  - `format_price` - formats price to default format.

    ```
    {{ item.product.price | format_price }}
    ```

    - **Original**: `100.42`
    - **Output**: `$100.42`

  - `convert` - converts a price from base currency to specified currency.

    ```
    {{ item.product.price | convert: 'EUR' }}
    ```

    - **Original**: `100`
    - **Output**: `92.28`

- **Array Filters**
  - `first` - return first element in array.
  - `last` - return last element in array.
  - `join` - join array to string using glue.
  - `size` - return the size of an array or a string.
- **URL Filters**
  - `resume` - resume customer's session and redirect it to base URL

    ```
    {{ item.product.product_url | resume }}
    ```
- **Image Filters**
  - `resize` - resize image

    ```
    {{ item.product.image | resize: 'small_image', 100, 100 }}
    ```

    - **Original**: `http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg`
    - **Output**: `http://example.com/pub/media/cache/100x100/catalog/product/m/h/mh0 black_main.jpg`

PHP Callouts

PHP Callouts is a very powerful tool to enhance your templates. It allows you to include PHP code directly to the HTML Code.

Here is the list of possible callouts with respective examples:

- **Global Methods**
  - `getUnsubscribeUrl` - a direct link to unsubscribe from current trigger.

    The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for the current trigger.

    This link does not unsubscribe customers from future emails (triggered by other events) or native Magento subscription.

Usage: `<a href="<?php echo $this->getUnsubscribeUrl() ?>">Unsubscribe</a>`

- `getUnsubscribeAllUrl` - a direct link to unsubscribe from all triggers

  The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for all triggers.

  This link does not unsubscribe customers from native Magento subscription.

  Usage: `<a href="<?php echo $this->getUnsubscribeAllUrl() ?>">Unsubscribe</a>`

- `getViewInBrowserUrl` - a direct link to open email in browser

  Usage: `<a href="<?php echo $this->getViewInBrowserUrl() ?>">View it in your browser.</a>`

- `getResumeUrl` - a direct link to resume (restore, log in) customer session

  Usage: `<a href="<?php echo $this->getResumeUrl() ?>">Open</a>`

  i.e., the customer will be automatically authorized in the store.

  Additionally, you can pass a parameter to the method to redirect the customer to a specific URL after authorization.

  Example

```
<?php foreach($this->getOrder()->getAllVisibleItems() as $item): ?
  <tr>
    <td>
      <a href="<?php echo $this->getResumeUrl($item->getProduct()-
    </td>
  </tr>
<?php endforeach ?>
```

  i.e., the customer will be redirected to the product page to leave a review after automatic authorization.

- `getStoreUrl` - a direct link to the store home page

  Usage: `<?php echo $this->getStoreUrl() ?>`

- `getStoreName` - a curent store name

  Usage: `<?php echo $this->getStoreName() ?>`

- `getStorePhone` - a curent store phone

  Usage: `<?php echo $this->getStorePhone() ?>`

- `getStoreAddress` - a curent store address

  Usage: `<?php echo $this->getStoreAddress() ?>`

- `getStoreEmail` - a curent store general transactional email

  Usage: `<?php echo $this->getStoreEmail() ?>`

- **Customer Methods**

  - `getCustomerName` - returns customer's full name

    Usage: `Dear <?php echo $this->getCustomerName() ?>`

    You can pass a parameter to the method **getCustomerName()** which will be used instead of the customer name, if the customer's name is empty: `Dear <?php echo $this->getCustomerName(null, 'Customer') ?>`, results in **Dear Customer**, if customer's name is empty (since version 1.0.34).

  - `getFirstname` - returns customer's firstname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getFirstname() ?>`

  - `getLastname` - returns customer's lastname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getLastname() ?>`

  - `getCustomer` - retrurns customer's object (only for registered customers)

    Usage: `Hi <?php echo $this->getCustomer()->getFirstname() ?>`

    `<?php echo $this->getCustomer()->getEmail() ?>`

- **Shopping Cart Methods**

  - `getRestoreCartUrl` - a direct link to customer shopping cart

Usage: `<a href="<?php echo $this->getRestoreCartUrl() ?>">Finish Checkout!</a>`

- `getReorderCartUrl` - redirects the customer to the quote with the products purchased from the previous order

  Usage: `<a href="<?php echo $this->getReorderCartUrl() ?>">Reorder</a>`

- `getQuote()->getAllVisibleItems()` - return collection of products in cart for feature output

  Usage:

  ```
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Order Methods**

  - `getOrder()->getStatus()` - the status of order

    Usage: `order status is <?php echo $this->getOrder()->getStatus() ?>`

  - `getOrder()->getIncrementId()` - the order number

    Usage: `Order #<?php echo $this->getOrder()->getIncrementId() ?>`

  - `getOrder()->getStoreGroupName()` - the store name of order

    Usage: `You placed order in <?php echo $this->getOrder()->getStoreGroupName() ?>`

- `getOrder()->getAllVisibleItems()` - return list of products in order for feature output

  Usage:

  ```
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Additional Methods**

  Usage:

  ```
  <?php echo $this->getOrder()->getBaseTaxAmount() ?>
  <?php echo $this->getOrder()->getBaseGrandTotal() ?>
  <?php echo $this->getOrder()->getBaseShippingAmount() ?>
  <?php echo $this->getOrder()->getShippingDescription() ?> - returns
  ```

- **Coupons**

  - `getCoupon()->getCode()` - get the expiration date of an autogenerated coupon code

    Usage: `Expiration date: <?php echo $this->formatDate($this->getCoupon()->getExpirationDate()) ?>`

    Different date formats:

    ```
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    ```

    Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

i.e., we only display this text block if a coupon is available.

- `getCoupon()->getExpirationDate()` - the autogenerated coupon code

  Usage: `Your coupon code: <?php echo $this->getCoupon()->getCode() ?>`

  Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

  i.e., we only display this text block if a coupon is available.

- **Cross-sell products**

  - `getCrossSellHtml` - html block of cross sell products

    Usage: `<?php echo getCrossSellHtml() ?>`

    Example

```
<?php if ($this->getCrossSellHtml()): ?>
    <h1>See also:</h1>
    <?php echo $this->getCrossSellHtml() ?>
<?php endif ?>
```

    i.e., we only display this text block if products are available.

- **Products Methods**

  - `getProductUrl` - a direct link to the product

    Usage: `<?php echo $item->getProduct()->getProductUrl() ?>`

  - `getPrice` - a price of the product

Usage: `<?php echo $item->getProduct()->getPrice() ?>`

```
<?php echo $this->formatPrice($item->getProduct()->getPrice()) ?>
```

- `getPriceInclTax` - a price of the product with tax (saved in order/shopping cart)

  Usage: `<?php echo $item->getPriceInclTax() ?>`

  ```
  <?= $this->formatPrice($item->getPriceInclTax()) ?>
  ```

- `getName` - a name of the product

  Usage: `<a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php echo $item->getName() ?></a>`

- **Image Directive**

  Usage: `<img src="<?php echo $this->getImageUrl($item->getProduct(), 'image', 100) ?>"/>`

  ```
  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'small_image', 150) ?>"/>

  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'thumbnail) ?>"/>
  ```

- **Wishlist Methods**

  - `getWishlist()->getItemCollection()` - return collection of products in wishlist for feature output

    Usage:

    ```
    <?php foreach ($this->getWishlist()->getItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?= $item->getProduct()->getProductUrl() ?>"><?php echo $i
    <?php endforeach ?>
    ```

    Alternative way of retrieving wishlist products:

    ```
    <?php foreach ($this->getWishlistItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php
    <?php endforeach ?>
    ```

  - `getWishlistProduct()` - return last added product to wishlist for feature output

    Usage:

```
<a href="<?php echo $this->getWishlistProduct()->getProductUrl()
?>"><?php echo $this->getWishlistProduct()->getName() ?></a>

Price: <?php echo $this->getWishlistProduct()->getPrice() ?>
```

- **Helper Methods**

Tip
The code below can be used to see available methods/properties for each of the mentioned above objects (product, quote, quote item, order, order item, order shipping address, order payment, customer, wishlist):

- Print all properties for **order object**:

  Usage:

```
<?php
echo '<pre>';
print_r($this->getOrder()->debug());
echo ' </pre>';
die();
?>
```

- Print all properties for **order item object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **product object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->getProduct()->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **customer object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getCustomer()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

- Print all properties for **wishlist object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getWishlist()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

  Note

  Object data returned as an array consisting of all the available data for the specified object. The object properties are displayed in the following way:

  ```
  [property_code] => property value
  [another_property_code] => property value
  [one_more_property_code] => property value
  ```

  Each property can be accessed separately as follows:

  ```
  <?php echo $this->getOrder()->getPropertyCode() ?>
  <?php echo $this->getOrder()->getAnotherPropertyCode() ?>
  <?php echo $this->getOrder()->getShippingAddress()->getAnotherPropertyCode
  <?php echo $this->getOrder()->getPayment()->getPropertyCode() ?>
  <?php echo $this->getCustomer()->getPropertyCode() ?>
  <?php echo $this->getWishlist()->getOneMorePropertyCode() ?>
  <?php echo $item->getProduct()->getPropertyCode() ?>
  ```

Variables & Methods

Our extension allows you to use variables and PHP callouts in your emails, which can greatly enhance and personalize them.

Note

Both variables and callouts can be used simultaneously, and are fully interchangeable so you can select your preferred syntax, and use it for all of your customizations.

Liquid is **the preferred syntax** for use with Email Templates. It allows you to avoid errors with the absence of attributes or values. Moreover, we provide a convenient helper dialog for inserting liquid variables into the editor.

Callouts are **deprecated** syntax, and used mainly for backward compatibility with older Follow-Up Email versions.

Liquid Variables

Liquid variables are a new way to enhance email templates. This syntax was introduced in version **1.1.15** and is the preferred syntax for use in Email Templates.

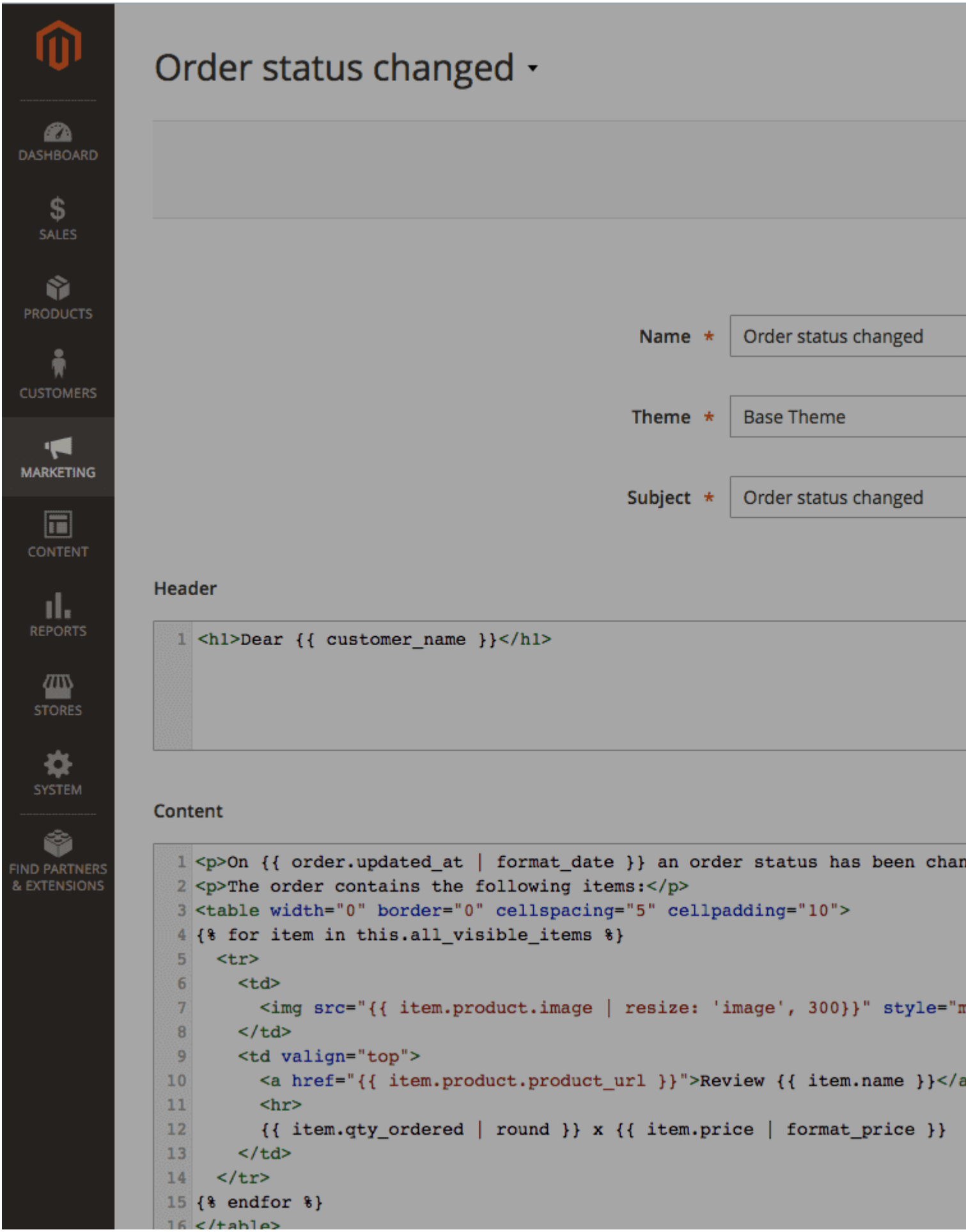
All variables should be enclosed in curly brackets. Each variable can also have a filter, added after pipe sign, and have one or more parameters.

Example

1. `{{ attribute | filter }}`
2. `{{ entity.attribute | filter | filter: param1 }}`
3. `{{ entity.entity.attribute | filter: param1,param2 }}`

These variables can be added interactively from edit pages of **Theme** or **Template**.

You just need to press the button **Insert Variable** near the content element, and select the variable you want to use - as shown below:


DASHBOARD
SALES
PRODUCTS
CUSTOMERS
MARKETING
CONTENT
REPORTS
STORES
SYSTEM
FIND PARTNERS & EXTENSIONS
Order status changed
Name * Order status changed
Theme * Base Theme
Subject * Order status changed
Header
1 <h1>Dear {{ customer_name }}</h1>
Content
1 On {{ order.updated_at | format_date }} an order status has been chan
2 The order contains the following items:
3 <table width="0" border="0" cellspacing="5" cellpadding="10">
4 {% for item in this.all_visible_items %}
5 <tr>
6 <td>
7 <img src="{{ item.product.image | resize: 'image', 300}}" style="m
8 </td>
9 <td valign="top">
10 <a href="{{ item.product.product_url }}">Review {{ item.name }}</a
11 <hr>
12 {{ item.qty_ordered | round }} x {{ item.price | format_price }}
13 </td>
14 </tr>
15 {% endfor %}
16 </table>

Modifying Variables with Filters

Filters are methods that allow you to alter or enhance the output of a variable. They also should be enclosed to the variable block `{{ }}`, but separated from the variable with a pipe (`|`) character. The parameters of filters are added using the colon (`:`) character.

Example

- `{{ item.product.name | truncate: '150' }}` - truncates the name of the product to 150 characters/
- `{{ item.product.weight | round: '2' }} kg` - rounds the weight of the product to 2 decimal digits

Here is the list of available filters, grouped into categories, with examples:

- **String/HTML Filters**
  - `downcase` - converts a string into lowercase.

    ```
    {{ item.product.name | downcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `dash digital watch`
  - `upcase` - converts a string into uppercase.

    ```
    {{ item.product.name | upcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `DASH DIGITAL WATCH`
  - `replace` - replaces all occurrences of a string with a substring.

    ```
    {{ item.product.name | replace: 'Digital', 'Analog' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Analog watch`
  - `append` - appends characters to a string.

    ```
    {{ item.product.name | append: ' - best choice' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Digital Watch - best choice`
  - `prepend` - prepends characters to a string.

    ```
    {{ item.product.name | prepend: 'Best choice - ' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Best choice - Dash Digital Watch`

- `capitalize` - capitalizes words in the input sentence.

  ```
  {{ item.product.color | capitalize }}
  ```

  - **Original**: `dark red`
  - **Output**: `Dark red`

- `escape` - escapes HTML tags in a string.

  ```
  {{ item.product.description | escape }
  ```

- `newline_to_br` - inserts a `<br>` linebreak HTML tag in front of each line break in a string.

  ```
  {{ item.product.short_description | newline_to_br }
  ```

- `remove` - removes all occurrences of a substring from a string.

  ```
  {{ item.product.name | remove: 'Digital' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `Dash Watch`

- `strip_html` - strips all HTML tags from a string.

  ```
  {{ item.product.description | strip_html }
  ```

- `truncate` - truncates a string down to 'x' characters.

  ```
  {{ item.product.name | truncate: '15' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `ash Digital Wa`

- `if_empty` - return argument, if the value is an empty string

  ```
  Dear {{ customer_name | if_empty: 'Client' }}!
  ```

  - **Original**: `empty string`
  - **Output**: `Dear Client!`

- `date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | date: '%d.%m.%Y %H:%M' }}
  ```

  - **Original**: `2016-02-18 10:11:12`
  - **Output**: `18.02.2016 10:11`

  Full list of formatters can be found **here**

- `format_date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | format_date: 3 }}
  ```

- **Original**: `2016-02-18 10:11:12`
- **Output**: `18/02/16`

Possible formatters: `0, 1 , 2 , 3`

- **Numeric Filters**

  - `ceil` - rounds the output up to the nearest integer.

    ```
    {{ item.product.weight | ceil }}
    ```

    - **Original**: `1.423`
    - **Output**: `2`

  - `floor` - rounds the output down to the nearest integer.

    ```
    {{ item.product.weight | floor }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `round` - rounds the output to the nearest integer or specified decimal digits.

    ```
    {{ item.product.weight | round: '2' }}
    ```

    - **Original**: `1.423`
    - **Output**: `1.42`

    ```
    {{ item.product.weight | round }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `number_format` - formats number to specified format (php function).

    ```
    {{ item.product.price | number_format: '2', '.', ',' }}
    ```

- **Price/Currency Filters**

  - `format_price` - formats price to default format.

    ```
    {{ item.product.price | format_price }}
    ```

    - **Original**: `100.42`
    - **Output**: `$100.42`

  - `convert` - converts a price from base currency to specified currency.

    ```
    {{ item.product.price | convert: 'EUR' }}
    ```

    - **Original**: `100`
    - **Output**: `92.28`

- **Array Filters**
  - `first` - return first element in array.
  - `last` - return last element in array.
  - `join` - join array to string using glue.
  - `size` - return the size of an array or a string.
- **URL Filters**
  - `resume` - resume customer's session and redirect it to base URL

    ```
    {{ item.product.product_url | resume }}
    ```
- **Image Filters**
  - `resize` - resize image

    ```
    {{ item.product.image | resize: 'small_image', 100, 100 }}
    ```

    - **Original**: `http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg`
    - **Output**: `http://example.com/pub/media/cache/100x100/catalog/product/m/h/mh0 black_main.jpg`

PHP Callouts

PHP Callouts is a very powerful tool to enhance your templates. It allows you to include PHP code directly to the HTML Code.

Here is the list of possible callouts with respective examples:

- **Global Methods**
  - `getUnsubscribeUrl` - a direct link to unsubscribe from current trigger.

    The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for the current trigger.

    This link does not unsubscribe customers from future emails (triggered by other events) or native Magento subscription.

Usage: `<a href="<?php echo $this->getUnsubscribeUrl() ?>">Unsubscribe</a>`

- `getUnsubscribeAllUrl` - a direct link to unsubscribe from all triggers

  The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for all triggers.

  This link does not unsubscribe customers from native Magento subscription.

  Usage: `<a href="<?php echo $this->getUnsubscribeAllUrl() ?>">Unsubscribe</a>`

- `getViewInBrowserUrl` - a direct link to open email in browser

  Usage: `<a href="<?php echo $this->getViewInBrowserUrl() ?>">View it in your browser.</a>`

- `getResumeUrl` - a direct link to resume (restore, log in) customer session

  Usage: `<a href="<?php echo $this->getResumeUrl() ?>">Open</a>`

  i.e., the customer will be automatically authorized in the store.

  Additionally, you can pass a parameter to the method to redirect the customer to a specific URL after authorization.

  Example

```
<?php foreach($this->getOrder()->getAllVisibleItems() as $item): ?
  <tr>
    <td>
      <a href="<?php echo $this->getResumeUrl($item->getProduct()-
    </td>
  </tr>
<?php endforeach ?>
```

  i.e., the customer will be redirected to the product page to leave a review after automatic authorization.

- `getStoreUrl` - a direct link to the store home page

  Usage: `<?php echo $this->getStoreUrl() ?>`

- `getStoreName` - a curent store name

  Usage: `<?php echo $this->getStoreName() ?>`

- `getStorePhone` - a curent store phone

  Usage: `<?php echo $this->getStorePhone() ?>`

- `getStoreAddress` - a curent store address

  Usage: `<?php echo $this->getStoreAddress() ?>`

- `getStoreEmail` - a curent store general transactional email

  Usage: `<?php echo $this->getStoreEmail() ?>`

- **Customer Methods**

  - `getCustomerName` - returns customer's full name

    Usage: `Dear <?php echo $this->getCustomerName() ?>`

    You can pass a parameter to the method **getCustomerName()** which will be used instead of the customer name, if the customer's name is empty: `Dear <?php echo $this->getCustomerName(null, 'Customer') ?>`, results in **Dear Customer**, if customer's name is empty (since version 1.0.34).

  - `getFirstname` - returns customer's firstname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getFirstname() ?>`

  - `getLastname` - returns customer's lastname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getLastname() ?>`

  - `getCustomer` - retrurns customer's object (only for registered customers)

    Usage: `Hi <?php echo $this->getCustomer()->getFirstname() ?>`

    `<?php echo $this->getCustomer()->getEmail() ?>`

- **Shopping Cart Methods**

  - `getRestoreCartUrl` - a direct link to customer shopping cart

Usage: `<a href="<?php echo $this->getRestoreCartUrl() ?>">Finish Checkout!</a>`

- `getReorderCartUrl` - redirects the customer to the quote with the products purchased from the previous order

  Usage: `<a href="<?php echo $this->getReorderCartUrl() ?>">Reorder</a>`

- `getQuote()->getAllVisibleItems()` - return collection of products in cart for feature output

  Usage:

  ```
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Order Methods**

  - `getOrder()->getStatus()` - the status of order

    Usage: `order status is <?php echo $this->getOrder()->getStatus() ?>`

  - `getOrder()->getIncrementId()` - the order number

    Usage: `Order #<?php echo $this->getOrder()->getIncrementId() ?>`

  - `getOrder()->getStoreGroupName()` - the store name of order

    Usage: `You placed order in <?php echo $this->getOrder()->getStoreGroupName() ?>`

- `getOrder()->getAllVisibleItems()` - return list of products in order for feature output

  Usage:

  ```
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Additional Methods**

  Usage:

  ```
  <?php echo $this->getOrder()->getBaseTaxAmount() ?>
  <?php echo $this->getOrder()->getBaseGrandTotal() ?>
  <?php echo $this->getOrder()->getBaseShippingAmount() ?>
  <?php echo $this->getOrder()->getShippingDescription() ?> - returns
  ```

- **Coupons**

  - `getCoupon()->getCode()` - get the expiration date of an autogenerated coupon code

    Usage: `Expiration date: <?php echo $this->formatDate($this->getCoupon()->getExpirationDate()) ?>`

    Different date formats:

    ```
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    ```

    Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

i.e., we only display this text block if a coupon is available.

- `getCoupon()->getExpirationDate()` - the autogenerated coupon code

  Usage: `Your coupon code: <?php echo $this->getCoupon()->getCode() ?>`

  Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

  i.e., we only display this text block if a coupon is available.

- **Cross-sell products**

  - `getCrossSellHtml` - html block of cross sell products

    Usage: `<?php echo getCrossSellHtml() ?>`

    Example

```
<?php if ($this->getCrossSellHtml()): ?>
    <h1>See also:</h1>
    <?php echo $this->getCrossSellHtml() ?>
<?php endif ?>
```

    i.e., we only display this text block if products are available.

- **Products Methods**

  - `getProductUrl` - a direct link to the product

    Usage: `<?php echo $item->getProduct()->getProductUrl() ?>`

  - `getPrice` - a price of the product

Usage: `<?php echo $item->getProduct()->getPrice() ?>`

```
<?php echo $this->formatPrice($item->getProduct()->getPrice()) ?>
```

- `getPriceInclTax` - a price of the product with tax (saved in order/shopping cart)

  Usage: `<?php echo $item->getPriceInclTax() ?>`

  ```
  <?= $this->formatPrice($item->getPriceInclTax()) ?>
  ```

- `getName` - a name of the product

  Usage: `<a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php echo $item->getName() ?></a>`

- **Image Directive**

  Usage: `<img src="<?php echo $this->getImageUrl($item->getProduct(), 'image', 100) ?>"/>`

  ```
  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'small_image', 150) ?>"/>

  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'thumbnail) ?>"/>
  ```

- **Wishlist Methods**

  - `getWishlist()->getItemCollection()` - return collection of products in wishlist for feature output

    Usage:

    ```
    <?php foreach ($this->getWishlist()->getItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?= $item->getProduct()->getProductUrl() ?>"><?php echo $i
    <?php endforeach ?>
    ```

    Alternative way of retrieving wishlist products:

    ```
    <?php foreach ($this->getWishlistItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php
    <?php endforeach ?>
    ```

  - `getWishlistProduct()` - return last added product to wishlist for feature output

    Usage:

```
<a href="<?php echo $this->getWishlistProduct()->getProductUrl()
?>"><?php echo $this->getWishlistProduct()->getName() ?></a>

Price: <?php echo $this->getWishlistProduct()->getPrice() ?>
```

- **Helper Methods**

Tip
The code below can be used to see available methods/properties for each of the mentioned above objects (product, quote, quote item, order, order item, order shipping address, order payment, customer, wishlist):

- Print all properties for **order object**:

  Usage:

```
<?php
echo '<pre>';
print_r($this->getOrder()->debug());
echo ' </pre>';
die();
?>
```

- Print all properties for **order item object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **product object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->getProduct()->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **customer object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getCustomer()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

- Print all properties for **wishlist object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getWishlist()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

  Note

  Object data returned as an array consisting of all the available data for the specified object. The object properties are displayed in the following way:

  ```
  [property_code] => property value
  [another_property_code] => property value
  [one_more_property_code] => property value
  ```

  Each property can be accessed separately as follows:

  ```
  <?php echo $this->getOrder()->getPropertyCode() ?>
  <?php echo $this->getOrder()->getAnotherPropertyCode() ?>
  <?php echo $this->getOrder()->getShippingAddress()->getAnotherPropertyCode
  <?php echo $this->getOrder()->getPayment()->getPropertyCode() ?>
  <?php echo $this->getCustomer()->getPropertyCode() ?>
  <?php echo $this->getWishlist()->getOneMorePropertyCode() ?>
  <?php echo $item->getProduct()->getPropertyCode() ?>
  ```

Variables & Methods

Our extension allows you to use variables and PHP callouts in your emails, which can greatly enhance and personalize them.

Note

Both variables and callouts can be used simultaneously, and are fully interchangeable so you can select your preferred syntax, and use it for all of your customizations.

Liquid is **the preferred syntax** for use with Email Templates. It allows you to avoid errors with the absence of attributes or values. Moreover, we provide a convenient helper dialog for inserting liquid variables into the editor.

Callouts are **deprecated** syntax, and used mainly for backward compatibility with older Follow-Up Email versions.

Liquid Variables

Liquid variables are a new way to enhance email templates. This syntax was introduced in version **1.1.15** and is the preferred syntax for use in Email Templates.

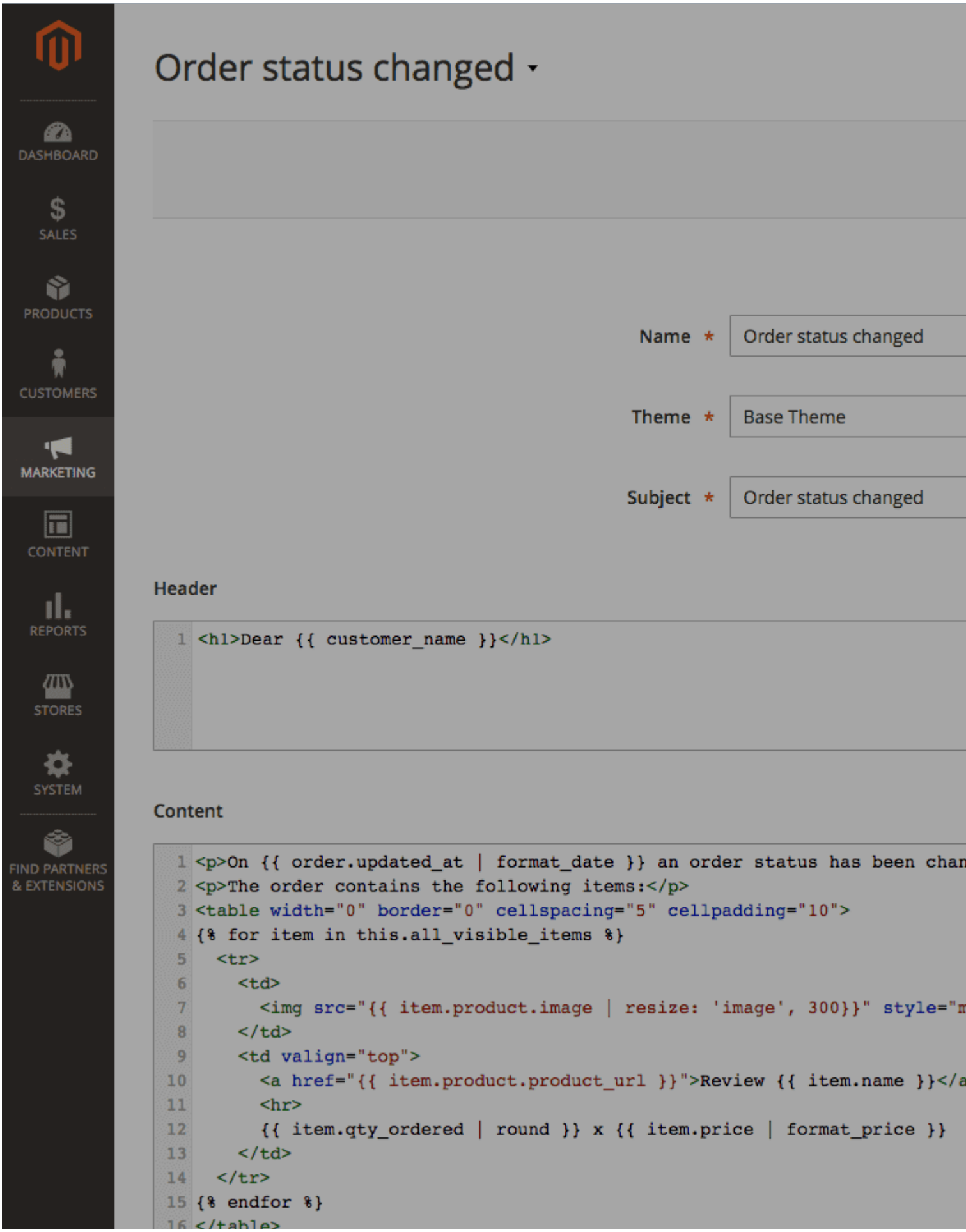
All variables should be enclosed in curly brackets. Each variable can also have a filter, added after pipe sign, and have one or more parameters.

Example

1. `{{ attribute | filter }}`
2. `{{ entity.attribute | filter | filter: param1 }}`
3. `{{ entity.entity.attribute | filter: param1,param2 }}`

These variables can be added interactively from edit pages of **Theme** or **Template**.

You just need to press the button **Insert Variable** near the content element, and select the variable you want to use - as shown below:


DASHBOARD
SALES
PRODUCTS
CUSTOMERS
MARKETING
CONTENT
REPORTS
STORES
SYSTEM
FIND PARTNERS & EXTENSIONS
Order status changed
Name * Order status changed
Theme * Base Theme
Subject * Order status changed
Header
1 <h1>Dear {{ customer_name }}</h1>
Content
1 On {{ order.updated_at | format_date }} an order status has been chan
2 The order contains the following items:
3 <table width="0" border="0" cellspacing="5" cellpadding="10">
4 {% for item in this.all_visible_items %}
5 <tr>
6 <td>
7 <img src="{{ item.product.image | resize: 'image', 300}}" style="m
8 </td>
9 <td valign="top">
10 <a href="{{ item.product.product_url }}">Review {{ item.name }}</a
11 <hr>
12 {{ item.qty_ordered | round }} x {{ item.price | format_price }}
13 </td>
14 </tr>
15 {% endfor %}
16 </table>

Modifying Variables with Filters

Filters are methods that allow you to alter or enhance the output of a variable. They also should be enclosed to the variable block `{{ }}`, but separated from the variable with a pipe (`|`) character. The parameters of filters are added using the colon (`:`) character.

Example

- `{{ item.product.name | truncate: '150' }}` - truncates the name of the product to 150 characters/
- `{{ item.product.weight | round: '2' }} kg` - rounds the weight of the product to 2 decimal digits

Here is the list of available filters, grouped into categories, with examples:

- **String/HTML Filters**
  - `downcase` - converts a string into lowercase.

    ```
    {{ item.product.name | downcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `dash digital watch`
  - `upcase` - converts a string into uppercase.

    ```
    {{ item.product.name | upcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `DASH DIGITAL WATCH`
  - `replace` - replaces all occurrences of a string with a substring.

    ```
    {{ item.product.name | replace: 'Digital', 'Analog' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Analog watch`
  - `append` - appends characters to a string.

    ```
    {{ item.product.name | append: ' - best choice' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Digital Watch - best choice`
  - `prepend` - prepends characters to a string.

    ```
    {{ item.product.name | prepend: 'Best choice - ' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Best choice - Dash Digital Watch`

- `capitalize` - capitalizes words in the input sentence.

  ```
  {{ item.product.color | capitalize }}
  ```

  - **Original**: `dark red`
  - **Output**: `Dark red`

- `escape` - escapes HTML tags in a string.

  ```
  {{ item.product.description | escape }
  ```

- `newline_to_br` - inserts a `<br>` linebreak HTML tag in front of each line break in a string.

  ```
  {{ item.product.short_description | newline_to_br }
  ```

- `remove` - removes all occurrences of a substring from a string.

  ```
  {{ item.product.name | remove: 'Digital' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `Dash Watch`

- `strip_html` - strips all HTML tags from a string.

  ```
  {{ item.product.description | strip_html }
  ```

- `truncate` - truncates a string down to 'x' characters.

  ```
  {{ item.product.name | truncate: '15' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `ash Digital Wa`

- `if_empty` - return argument, if the value is an empty string

  ```
  Dear {{ customer_name | if_empty: 'Client' }}!
  ```

  - **Original**: `empty string`
  - **Output**: `Dear Client!`

- `date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | date: '%d.%m.%Y %H:%M' }}
  ```

  - **Original**: `2016-02-18 10:11:12`
  - **Output**: `18.02.2016 10:11`

  Full list of formatters can be found **here**

- `format_date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | format_date: 3 }}
  ```

- **Original**: `2016-02-18 10:11:12`
- **Output**: `18/02/16`

Possible formatters: `0, 1 , 2 , 3`

- **Numeric Filters**

  - `ceil` - rounds the output up to the nearest integer.

    ```
    {{ item.product.weight | ceil }}
    ```

    - **Original**: `1.423`
    - **Output**: `2`

  - `floor` - rounds the output down to the nearest integer.

    ```
    {{ item.product.weight | floor }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `round` - rounds the output to the nearest integer or specified decimal digits.

    ```
    {{ item.product.weight | round: '2' }}
    ```

    - **Original**: `1.423`
    - **Output**: `1.42`

    ```
    {{ item.product.weight | round }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `number_format` - formats number to specified format (php function).

    ```
    {{ item.product.price | number_format: '2', '.', ',' }}
    ```

- **Price/Currency Filters**

  - `format_price` - formats price to default format.

    ```
    {{ item.product.price | format_price }}
    ```

    - **Original**: `100.42`
    - **Output**: `$100.42`

  - `convert` - converts a price from base currency to specified currency.

    ```
    {{ item.product.price | convert: 'EUR' }}
    ```

    - **Original**: `100`
    - **Output**: `92.28`

- **Array Filters**
  - `first` - return first element in array.
  - `last` - return last element in array.
  - `join` - join array to string using glue.
  - `size` - return the size of an array or a string.
- **URL Filters**
  - `resume` - resume customer's session and redirect it to base URL

    ```
    {{ item.product.product_url | resume }}
    ```
- **Image Filters**
  - `resize` - resize image

    ```
    {{ item.product.image | resize: 'small_image', 100, 100 }}
    ```

    - **Original**: `http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg`
    - **Output**: `http://example.com/pub/media/cache/100x100/catalog/product/m/h/mh0 black_main.jpg`

PHP Callouts

PHP Callouts is a very powerful tool to enhance your templates. It allows you to include PHP code directly to the HTML Code.

Here is the list of possible callouts with respective examples:

- **Global Methods**
  - `getUnsubscribeUrl` - a direct link to unsubscribe from current trigger.

    The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for the current trigger.

    This link does not unsubscribe customers from future emails (triggered by other events) or native Magento subscription.

Usage: `<a href="<?php echo $this->getUnsubscribeUrl() ?>">Unsubscribe</a>`

- `getUnsubscribeAllUrl` - a direct link to unsubscribe from all triggers

  The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for all triggers.

  This link does not unsubscribe customers from native Magento subscription.

  Usage: `<a href="<?php echo $this->getUnsubscribeAllUrl() ?>">Unsubscribe</a>`

- `getViewInBrowserUrl` - a direct link to open email in browser

  Usage: `<a href="<?php echo $this->getViewInBrowserUrl() ?>">View it in your browser.</a>`

- `getResumeUrl` - a direct link to resume (restore, log in) customer session

  Usage: `<a href="<?php echo $this->getResumeUrl() ?>">Open</a>`

  i.e., the customer will be automatically authorized in the store.

  Additionally, you can pass a parameter to the method to redirect the customer to a specific URL after authorization.

  Example

```
<?php foreach($this->getOrder()->getAllVisibleItems() as $item): ?
  <tr>
    <td>
      <a href="<?php echo $this->getResumeUrl($item->getProduct()-
    </td>
  </tr>
<?php endforeach ?>
```

  i.e., the customer will be redirected to the product page to leave a review after automatic authorization.

- `getStoreUrl` - a direct link to the store home page

  Usage: `<?php echo $this->getStoreUrl() ?>`

- `getStoreName` - a curent store name

  Usage: `<?php echo $this->getStoreName() ?>`

- `getStorePhone` - a curent store phone

  Usage: `<?php echo $this->getStorePhone() ?>`

- `getStoreAddress` - a curent store address

  Usage: `<?php echo $this->getStoreAddress() ?>`

- `getStoreEmail` - a curent store general transactional email

  Usage: `<?php echo $this->getStoreEmail() ?>`

- **Customer Methods**

  - `getCustomerName` - returns customer's full name

    Usage: `Dear <?php echo $this->getCustomerName() ?>`

    You can pass a parameter to the method **getCustomerName()** which will be used instead of the customer name, if the customer's name is empty: `Dear <?php echo $this->getCustomerName(null, 'Customer') ?>`, results in **Dear Customer**, if customer's name is empty (since version 1.0.34).

  - `getFirstname` - returns customer's firstname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getFirstname() ?>`

  - `getLastname` - returns customer's lastname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getLastname() ?>`

  - `getCustomer` - retrurns customer's object (only for registered customers)

    Usage: `Hi <?php echo $this->getCustomer()->getFirstname() ?>`

    `<?php echo $this->getCustomer()->getEmail() ?>`

- **Shopping Cart Methods**

  - `getRestoreCartUrl` - a direct link to customer shopping cart

Usage: `<a href="<?php echo $this->getRestoreCartUrl() ?>">Finish Checkout!</a>`

- `getReorderCartUrl` - redirects the customer to the quote with the products purchased from the previous order

  Usage: `<a href="<?php echo $this->getReorderCartUrl() ?>">Reorder</a>`

- `getQuote()->getAllVisibleItems()` - return collection of products in cart for feature output

  Usage:

  ```
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Order Methods**

  - `getOrder()->getStatus()` - the status of order

    Usage: `order status is <?php echo $this->getOrder()->getStatus() ?>`

  - `getOrder()->getIncrementId()` - the order number

    Usage: `Order #<?php echo $this->getOrder()->getIncrementId() ?>`

  - `getOrder()->getStoreGroupName()` - the store name of order

    Usage: `You placed order in <?php echo $this->getOrder()->getStoreGroupName() ?>`

- `getOrder()->getAllVisibleItems()` - return list of products in order for feature output

  Usage:

  ```
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Additional Methods**

  Usage:

  ```
  <?php echo $this->getOrder()->getBaseTaxAmount() ?>
  <?php echo $this->getOrder()->getBaseGrandTotal() ?>
  <?php echo $this->getOrder()->getBaseShippingAmount() ?>
  <?php echo $this->getOrder()->getShippingDescription() ?> - returns
  ```

- **Coupons**

  - `getCoupon()->getCode()` - get the expiration date of an autogenerated coupon code

    Usage: `Expiration date: <?php echo $this->formatDate($this->getCoupon()->getExpirationDate()) ?>`

    Different date formats:

    ```
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    ```

    Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

i.e., we only display this text block if a coupon is available.

- `getCoupon()->getExpirationDate()` - the autogenerated coupon code

  Usage: `Your coupon code: <?php echo $this->getCoupon()->getCode() ?>`

  Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

  i.e., we only display this text block if a coupon is available.

- **Cross-sell products**

  - `getCrossSellHtml` - html block of cross sell products

    Usage: `<?php echo getCrossSellHtml() ?>`

    Example

```
<?php if ($this->getCrossSellHtml()): ?>
    <h1>See also:</h1>
    <?php echo $this->getCrossSellHtml() ?>
<?php endif ?>
```

    i.e., we only display this text block if products are available.

- **Products Methods**

  - `getProductUrl` - a direct link to the product

    Usage: `<?php echo $item->getProduct()->getProductUrl() ?>`

  - `getPrice` - a price of the product

Usage: `<?php echo $item->getProduct()->getPrice() ?>`

```
<?php echo $this->formatPrice($item->getProduct()->getPrice()) ?>
```

- `getPriceInclTax` - a price of the product with tax (saved in order/shopping cart)

  Usage: `<?php echo $item->getPriceInclTax() ?>`

  ```
  <?= $this->formatPrice($item->getPriceInclTax()) ?>
  ```

- `getName` - a name of the product

  Usage: `<a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php echo $item->getName() ?></a>`

- **Image Directive**

  Usage: `<img src="<?php echo $this->getImageUrl($item->getProduct(), 'image', 100) ?>"/>`

  ```
  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'small_image', 150) ?>"/>

  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'thumbnail) ?>"/>
  ```

- **Wishlist Methods**

  - `getWishlist()->getItemCollection()` - return collection of products in wishlist for feature output

    Usage:

    ```
    <?php foreach ($this->getWishlist()->getItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?= $item->getProduct()->getProductUrl() ?>"><?php echo $i
    <?php endforeach ?>
    ```

    Alternative way of retrieving wishlist products:

    ```
    <?php foreach ($this->getWishlistItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php
    <?php endforeach ?>
    ```

  - `getWishlistProduct()` - return last added product to wishlist for feature output

    Usage:

```
<a href="<?php echo $this->getWishlistProduct()->getProductUrl()
?>"><?php echo $this->getWishlistProduct()->getName() ?></a>

Price: <?php echo $this->getWishlistProduct()->getPrice() ?>
```

- **Helper Methods**

Tip
The code below can be used to see available methods/properties for each of the mentioned above objects (product, quote, quote item, order, order item, order shipping address, order payment, customer, wishlist):

- Print all properties for **order object**:

  Usage:

```
<?php
echo '<pre>';
print_r($this->getOrder()->debug());
echo ' </pre>';
die();
?>
```

- Print all properties for **order item object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **product object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->getProduct()->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **customer object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getCustomer()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

- Print all properties for **wishlist object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getWishlist()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

  Note

  Object data returned as an array consisting of all the available data for the specified object. The object properties are displayed in the following way:

  ```
  [property_code] => property value
  [another_property_code] => property value
  [one_more_property_code] => property value
  ```

  Each property can be accessed separately as follows:

  ```
  <?php echo $this->getOrder()->getPropertyCode() ?>
  <?php echo $this->getOrder()->getAnotherPropertyCode() ?>
  <?php echo $this->getOrder()->getShippingAddress()->getAnotherPropertyCode
  <?php echo $this->getOrder()->getPayment()->getPropertyCode() ?>
  <?php echo $this->getCustomer()->getPropertyCode() ?>
  <?php echo $this->getWishlist()->getOneMorePropertyCode() ?>
  <?php echo $item->getProduct()->getPropertyCode() ?>
  ```

Variables & Methods

Our extension allows you to use variables and PHP callouts in your emails, which can greatly enhance and personalize them.

Note

Both variables and callouts can be used simultaneously, and are fully interchangeable so you can select your preferred syntax, and use it for all of your customizations.

Liquid is **the preferred syntax** for use with Email Templates. It allows you to avoid errors with the absence of attributes or values. Moreover, we provide a convenient helper dialog for inserting liquid variables into the editor.

Callouts are **deprecated** syntax, and used mainly for backward compatibility with older Follow-Up Email versions.

Liquid Variables

Liquid variables are a new way to enhance email templates. This syntax was introduced in version **1.1.15** and is the preferred syntax for use in Email Templates.

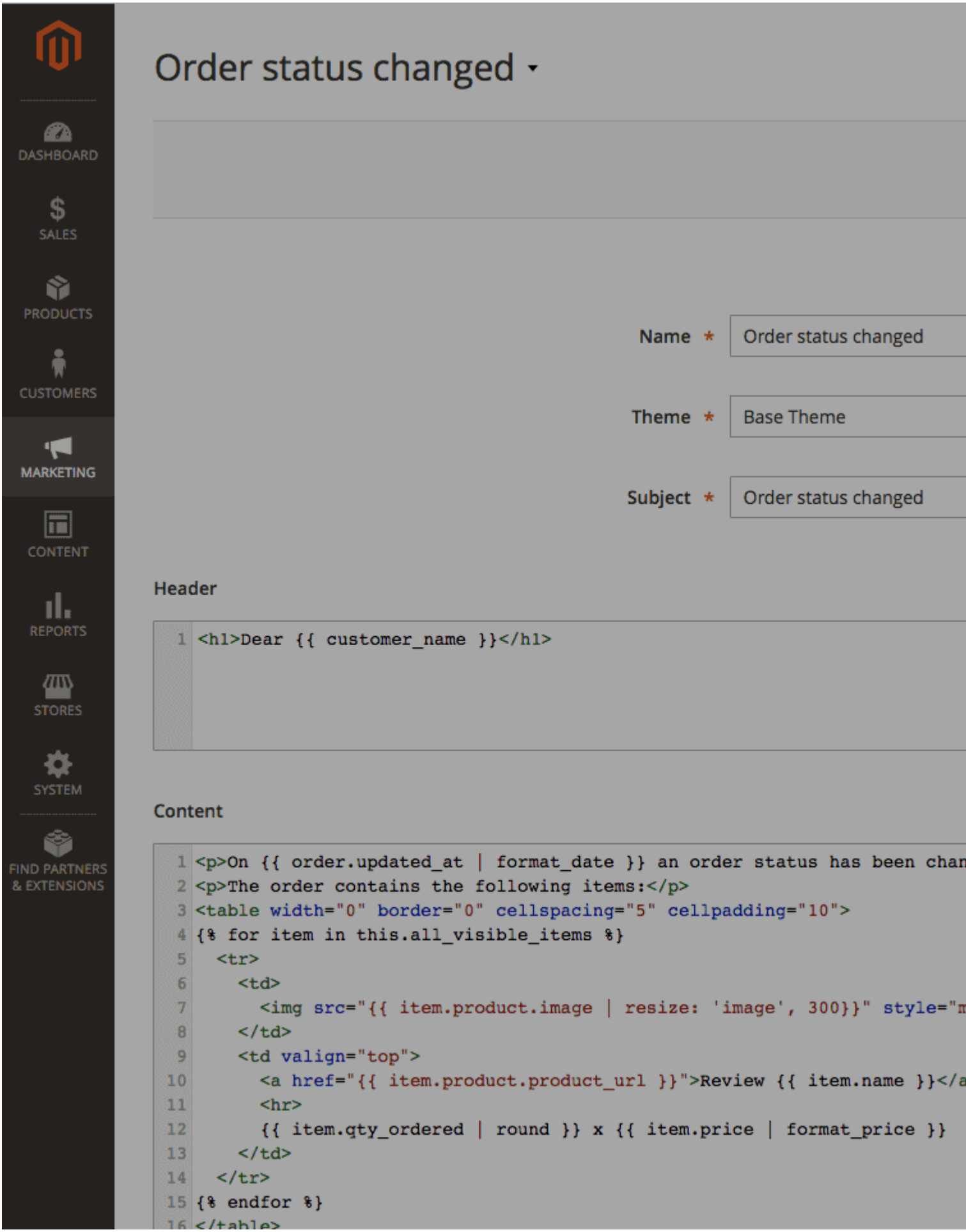
All variables should be enclosed in curly brackets. Each variable can also have a filter, added after pipe sign, and have one or more parameters.

Example

1. `{{ attribute | filter }}`
2. `{{ entity.attribute | filter | filter: param1 }}`
3. `{{ entity.entity.attribute | filter: param1,param2 }}`

These variables can be added interactively from edit pages of **Theme** or **Template**.

You just need to press the button **Insert Variable** near the content element, and select the variable you want to use - as shown below:


DASHBOARD
SALES
PRODUCTS
CUSTOMERS
MARKETING
CONTENT
REPORTS
STORES
SYSTEM
FIND PARTNERS & EXTENSIONS
Order status changed
Name * Order status changed
Theme * Base Theme
Subject * Order status changed
Header
1 <h1>Dear {{ customer_name }}</h1>
Content
1 On {{ order.updated_at | format_date }} an order status has been chan
2 The order contains the following items:
3 <table width="0" border="0" cellspacing="5" cellpadding="10">
4 {% for item in this.all_visible_items %}
5 <tr>
6 <td>
7 <img src="{{ item.product.image | resize: 'image', 300}}" style="m
8 </td>
9 <td valign="top">
10 <a href="{{ item.product.product_url }}">Review {{ item.name }}</a
11 <hr>
12 {{ item.qty_ordered | round }} x {{ item.price | format_price }}
13 </td>
14 </tr>
15 {% endfor %}
16 </table>

Modifying Variables with Filters

Filters are methods that allow you to alter or enhance the output of a variable. They also should be enclosed to the variable block `{{ }}`, but separated from the variable with a pipe (`|`) character. The parameters of filters are added using the colon (`:`) character.

Example

- `{{ item.product.name | truncate: '150' }}` - truncates the name of the product to 150 characters/
- `{{ item.product.weight | round: '2' }} kg` - rounds the weight of the product to 2 decimal digits

Here is the list of available filters, grouped into categories, with examples:

- **String/HTML Filters**
  - `downcase` - converts a string into lowercase.

    ```
    {{ item.product.name | downcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `dash digital watch`
  - `upcase` - converts a string into uppercase.

    ```
    {{ item.product.name | upcase }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `DASH DIGITAL WATCH`
  - `replace` - replaces all occurrences of a string with a substring.

    ```
    {{ item.product.name | replace: 'Digital', 'Analog' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Analog watch`
  - `append` - appends characters to a string.

    ```
    {{ item.product.name | append: ' - best choice' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Dash Digital Watch - best choice`
  - `prepend` - prepends characters to a string.

    ```
    {{ item.product.name | prepend: 'Best choice - ' }}
    ```

    - **Original**: `Dash Digital Watch`
    - **Output**: `Best choice - Dash Digital Watch`

- `capitalize` - capitalizes words in the input sentence.

  ```
  {{ item.product.color | capitalize }}
  ```

  - **Original**: `dark red`
  - **Output**: `Dark red`

- `escape` - escapes HTML tags in a string.

  ```
  {{ item.product.description | escape }
  ```

- `newline_to_br` - inserts a `<br>` linebreak HTML tag in front of each line break in a string.

  ```
  {{ item.product.short_description | newline_to_br }
  ```

- `remove` - removes all occurrences of a substring from a string.

  ```
  {{ item.product.name | remove: 'Digital' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `Dash Watch`

- `strip_html` - strips all HTML tags from a string.

  ```
  {{ item.product.description | strip_html }
  ```

- `truncate` - truncates a string down to 'x' characters.

  ```
  {{ item.product.name | truncate: '15' }}
  ```

  - **Original**: `Dash Digital Watch`
  - **Output**: `ash Digital Wa`

- `if_empty` - return argument, if the value is an empty string

  ```
  Dear {{ customer_name | if_empty: 'Client' }}!
  ```

  - **Original**: `empty string`
  - **Output**: `Dear Client!`

- `date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | date: '%d.%m.%Y %H:%M' }}
  ```

  - **Original**: `2016-02-18 10:11:12`
  - **Output**: `18.02.2016 10:11`

  Full list of formatters can be found **here**

- `format_date` - converts a string to a specified date-time format.

  ```
  {{ item.product.created_at | format_date: 3 }}
  ```

- **Original**: `2016-02-18 10:11:12`
- **Output**: `18/02/16`

Possible formatters: `0, 1 , 2 , 3`

- **Numeric Filters**

  - `ceil` - rounds the output up to the nearest integer.

    ```
    {{ item.product.weight | ceil }}
    ```

    - **Original**: `1.423`
    - **Output**: `2`

  - `floor` - rounds the output down to the nearest integer.

    ```
    {{ item.product.weight | floor }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `round` - rounds the output to the nearest integer or specified decimal digits.

    ```
    {{ item.product.weight | round: '2' }}
    ```

    - **Original**: `1.423`
    - **Output**: `1.42`

    ```
    {{ item.product.weight | round }}
    ```

    - **Original**: `1.423`
    - **Output**: `1`

  - `number_format` - formats number to specified format (php function).

    ```
    {{ item.product.price | number_format: '2', '.', ',' }}
    ```

- **Price/Currency Filters**

  - `format_price` - formats price to default format.

    ```
    {{ item.product.price | format_price }}
    ```

    - **Original**: `100.42`
    - **Output**: `$100.42`

  - `convert` - converts a price from base currency to specified currency.

    ```
    {{ item.product.price | convert: 'EUR' }}
    ```

    - **Original**: `100`
    - **Output**: `92.28`

- **Array Filters**
  - `first` - return first element in array.
  - `last` - return last element in array.
  - `join` - join array to string using glue.
  - `size` - return the size of an array or a string.
- **URL Filters**
  - `resume` - resume customer's session and redirect it to base URL

    ```
    {{ item.product.product_url | resume }}
    ```
- **Image Filters**
  - `resize` - resize image

    ```
    {{ item.product.image | resize: 'small_image', 100, 100 }}
    ```

    - **Original**: `http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg`
    - **Output**: `http://example.com/pub/media/cache/100x100/catalog/product/m/h/mh0 black_main.jpg`

PHP Callouts

PHP Callouts is a very powerful tool to enhance your templates. It allows you to include PHP code directly to the HTML Code.

Here is the list of possible callouts with respective examples:

- **Global Methods**
  - `getUnsubscribeUrl` - a direct link to unsubscribe from current trigger.

    The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for the current trigger.

    This link does not unsubscribe customers from future emails (triggered by other events) or native Magento subscription.

Usage: `<a href="<?php echo $this->getUnsubscribeUrl() ?>">Unsubscribe</a>`

- `getUnsubscribeAllUrl` - a direct link to unsubscribe from all triggers

  The customer will be unsubscribed from all already scheduled emails (**Follow Up Email -> Mail Log (Queue)**) for all triggers.

  This link does not unsubscribe customers from native Magento subscription.

  Usage: `<a href="<?php echo $this->getUnsubscribeAllUrl() ?>">Unsubscribe</a>`

- `getViewInBrowserUrl` - a direct link to open email in browser

  Usage: `<a href="<?php echo $this->getViewInBrowserUrl() ?>">View it in your browser.</a>`

- `getResumeUrl` - a direct link to resume (restore, log in) customer session

  Usage: `<a href="<?php echo $this->getResumeUrl() ?>">Open</a>`

  i.e., the customer will be automatically authorized in the store.

  Additionally, you can pass a parameter to the method to redirect the customer to a specific URL after authorization.

  Example

```
<?php foreach($this->getOrder()->getAllVisibleItems() as $item): ?
  <tr>
    <td>
      <a href="<?php echo $this->getResumeUrl($item->getProduct()-
    </td>
  </tr>
<?php endforeach ?>
```

  i.e., the customer will be redirected to the product page to leave a review after automatic authorization.

- `getStoreUrl` - a direct link to the store home page

  Usage: `<?php echo $this->getStoreUrl() ?>`

- `getStoreName` - a curent store name

  Usage: `<?php echo $this->getStoreName() ?>`

- `getStorePhone` - a curent store phone

  Usage: `<?php echo $this->getStorePhone() ?>`

- `getStoreAddress` - a curent store address

  Usage: `<?php echo $this->getStoreAddress() ?>`

- `getStoreEmail` - a curent store general transactional email

  Usage: `<?php echo $this->getStoreEmail() ?>`

- **Customer Methods**

  - `getCustomerName` - returns customer's full name

    Usage: `Dear <?php echo $this->getCustomerName() ?>`

    You can pass a parameter to the method **getCustomerName()** which will be used instead of the customer name, if the customer's name is empty: `Dear <?php echo $this->getCustomerName(null, 'Customer') ?>`, results in **Dear Customer**, if customer's name is empty (since version 1.0.34).

  - `getFirstname` - returns customer's firstname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getFirstname() ?>`

  - `getLastname` - returns customer's lastname (since version 1.0.36)

    Usage: `Dear <?php echo $this->getLastname() ?>`

  - `getCustomer` - retrurns customer's object (only for registered customers)

    Usage: `Hi <?php echo $this->getCustomer()->getFirstname() ?>`

    `<?php echo $this->getCustomer()->getEmail() ?>`

- **Shopping Cart Methods**

  - `getRestoreCartUrl` - a direct link to customer shopping cart

Usage: `<a href="<?php echo $this->getRestoreCartUrl() ?>">Finish Checkout!</a>`

- `getReorderCartUrl` - redirects the customer to the quote with the products purchased from the previous order

  Usage: `<a href="<?php echo $this->getReorderCartUrl() ?>">Reorder</a>`

- `getQuote()->getAllVisibleItems()` - return collection of products in cart for feature output

  Usage:

  ```
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getQuote()->getAllVisibleItems() as $item): 
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Order Methods**

  - `getOrder()->getStatus()` - the status of order

    Usage: `order status is <?php echo $this->getOrder()->getStatus() ?>`

  - `getOrder()->getIncrementId()` - the order number

    Usage: `Order #<?php echo $this->getOrder()->getIncrementId() ?>`

  - `getOrder()->getStoreGroupName()` - the store name of order

    Usage: `You placed order in <?php echo $this->getOrder()->getStoreGroupName() ?>`

- `getOrder()->getAllVisibleItems()` - return list of products in order for feature output

  Usage:

  ```
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php echo $item->getName() ?>
  <?php endforeach ?>
  ```

  How to display only the first product:

  Example

  ```
  <?php $i = 0 ?>
  <?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?
      <?php if ($i++ >= 1): ?>
          <?php break ?>
      <?php endif ?>
      ...
      other methods
      ...
  <?php endforeach ?>
  ```

- **Additional Methods**

  Usage:

  ```
  <?php echo $this->getOrder()->getBaseTaxAmount() ?>
  <?php echo $this->getOrder()->getBaseGrandTotal() ?>
  <?php echo $this->getOrder()->getBaseShippingAmount() ?>
  <?php echo $this->getOrder()->getShippingDescription() ?> - returns
  ```

- **Coupons**

  - `getCoupon()->getCode()` - get the expiration date of an autogenerated coupon code

    Usage: `Expiration date: <?php echo $this->formatDate($this->getCoupon()->getExpirationDate()) ?>`

    Different date formats:

    ```
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    <?php echo $this->formatDate($this->getCoupon()->getExpirationDate
    ```

    Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

i.e., we only display this text block if a coupon is available.

- `getCoupon()->getExpirationDate()` - the autogenerated coupon code

  Usage: `Your coupon code: <?php echo $this->getCoupon()->getCode() ?>`

  Example

```
<?php if ($this->getCoupon()): ?>
    Let us offer you a discount to complete your purchase.<br>
    Your coupon code: <?php echo $this->getCoupon()->getCode() ?>
<?php endif ?>
```

  i.e., we only display this text block if a coupon is available.

- **Cross-sell products**

  - `getCrossSellHtml` - html block of cross sell products

    Usage: `<?php echo getCrossSellHtml() ?>`

    Example

```
<?php if ($this->getCrossSellHtml()): ?>
    <h1>See also:</h1>
    <?php echo $this->getCrossSellHtml() ?>
<?php endif ?>
```

    i.e., we only display this text block if products are available.

- **Products Methods**

  - `getProductUrl` - a direct link to the product

    Usage: `<?php echo $item->getProduct()->getProductUrl() ?>`

  - `getPrice` - a price of the product

Usage: `<?php echo $item->getProduct()->getPrice() ?>`

```
<?php echo $this->formatPrice($item->getProduct()->getPrice()) ?>
```

- `getPriceInclTax` - a price of the product with tax (saved in order/shopping cart)

  Usage: `<?php echo $item->getPriceInclTax() ?>`

  ```
  <?= $this->formatPrice($item->getPriceInclTax()) ?>
  ```

- `getName` - a name of the product

  Usage: `<a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php echo $item->getName() ?></a>`

- **Image Directive**

  Usage: `<img src="<?php echo $this->getImageUrl($item->getProduct(), 'image', 100) ?>"/>`

  ```
  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'small_image', 150) ?>"/>

  <img src="<?php echo $this->getImageUrl($item->getProduct(),
  'thumbnail) ?>"/>
  ```

- **Wishlist Methods**

  - `getWishlist()->getItemCollection()` - return collection of products in wishlist for feature output

    Usage:

    ```
    <?php foreach ($this->getWishlist()->getItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?= $item->getProduct()->getProductUrl() ?>"><?php echo $i
    <?php endforeach ?>
    ```

    Alternative way of retrieving wishlist products:

    ```
    <?php foreach ($this->getWishlistItemCollection() as $item): ?>
        <img src="<?=$this->getImageUrl($item->getProduct(), 'image', 300)
        <a href="<?php echo $item->getProduct()->getProductUrl() ?>"><?php
    <?php endforeach ?>
    ```

  - `getWishlistProduct()` - return last added product to wishlist for feature output

    Usage:

```
<a href="<?php echo $this->getWishlistProduct()->getProductUrl()
?>"><?php echo $this->getWishlistProduct()->getName() ?></a>

Price: <?php echo $this->getWishlistProduct()->getPrice() ?>
```

- **Helper Methods**

Tip
The code below can be used to see available methods/properties for each of the mentioned above objects (product, quote, quote item, order, order item, order shipping address, order payment, customer, wishlist):

- Print all properties for **order object**:

  Usage:

```
<?php
echo '<pre>';
print_r($this->getOrder()->debug());
echo ' </pre>';
die();
?>
```

- Print all properties for **order item object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **product object**:

  Usage:

```
<?php foreach ($this->getOrder()->getAllVisibleItems() as $item): ?>
    <?php
    echo '<pre>';
    print_r($item->getProduct()->debug());
    echo ' </pre>';
    die();
    ?>
<?php endforeach ?>
```

- Print all properties for **customer object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getCustomer()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

- Print all properties for **wishlist object**:

  Usage:

  ```
  <?php
  echo '<pre>';
  print_r($this->getWishlist()->debug());
  echo ' </pre>';
  die();
  ?>
  ```

  Note

  Object data returned as an array consisting of all the available data for the specified object. The object properties are displayed in the following way:

  ```
  [property_code] => property value
  [another_property_code] => property value
  [one_more_property_code] => property value
  ```

  Each property can be accessed separately as follows:

  ```
  <?php echo $this->getOrder()->getPropertyCode() ?>
  <?php echo $this->getOrder()->getAnotherPropertyCode() ?>
  <?php echo $this->getOrder()->getShippingAddress()->getAnotherPropertyCode
  <?php echo $this->getOrder()->getPayment()->getPropertyCode() ?>
  <?php echo $this->getCustomer()->getPropertyCode() ?>
  <?php echo $this->getWishlist()->getOneMorePropertyCode() ?>
  <?php echo $item->getProduct()->getPropertyCode() ?>
  ```

Mail Log (Queue)

This extension allows you to track triggered emails.

Go to **Marketing** > **Follow Up Email** > **Mail Log (Queue)**. You will see the following fields:

- **ID** - ID of the mail
- **Status** - current status of the trigger email. You can track emails with these statuses:
  - **Ready to go** - email is ready to be delivered

    - **Sent** - email has already been delivered
    - **Canceled** - email delivery was cancelled
    - **Error** - error has occured while email was being delivered
    - **Missed** - email delivery didn't occur
    - **Unsubscribed** - customer unsubscribed from the email newsletter
- **Trigger** - name of the email trigger
- **Scheduled At** - time when email was added to the queue
- **Sent At** - time when email was delivered to the recipient
- **Recipient Email** - customer email
- **Recipient Name** - customer name
- **Action** - actions on the selected emails in the queue:
    - **Cancel** - cancel processing of the trigger email in the queue
    - **Send** - send the trigger email to the **Recipient Email** (available in the mass actions)
    - **Change Status** - allows you to change the **Status** of the trigger email

Click on the triggered email. You will see detailed information about the triggered email in the next tabs: **General information**, **History**, **Variables**.

Event Log

This extension allows you to track triggered events at the store.

Go to **Marketing > Follow Up Email > Event Log**. You will see the following fields:

- **ID** - ID of the event.
- **Event** - a certain action of a visitor (login, registration, placing an order) or action of a system (change order status, change of price), which can be used as a trigger for emails.
- **Created At** - time when the event occurred.
- **Arguments** - additional data of the event (uniq_key, customer email, name, quote_id, order_id, store_id, time).
- **Triggers** - information about the amount and status of triggers for the current event.
- **Actions** - actions on the selected events:
    - **Reset & Process** - allows you to reset the current event's trigger status and bring it back to the queue.
    - **Delete** - remove the event from the list.

Note

Click the button **Fetch New Events** to show new triggered events at the store.

GDPR Compliance Tips

Since our module collects user data for sending emails, please check out some tips for making the extension fully compliant with the GDPR:

1. Send emails only to confirmed customers. You can use the rules to filter the target audience:

Example

- **Customer: Is subscriber of the newsletter is Yes**

2. There is a cron job **email_clean_history** which removes a month's worth of old emails and events
3. You can disable automatic guest user data capturing
4. You can manually delete all of the stored user data:
   - To remove emails, go to **Marketing > Follow Up Email > Mail Log**, select the required emails, choose the **Delete** option in the **Actions** dropdown, and press **Submit**.
   - To remove registered events, go to **Marketing > Follow Up Email > Event Log**, select the required events, choose the **Delete** option in the **Actions** dropdown, and press **Submit**.
   - To reset a particular statistics go to **Marketing > Follow Up Email > Settings**, press the button **Reset Statistics** at the Statistics tab.
5. Do not forget to include the unsubscription link to the emails using the variable `{{ url.unsubscribe_newsletter_url }}` - to unsubscribe from all triggers and the newsletter.

Troubleshoot

This section describes the most common problems that customers report and how they can be resolved:

- **The email does not show the coupon code**
- **The related, upsell or cross-sell products are not displayed**
- **How do I translate an email template?**

The email does not show the coupon code

The coupon code is not visible in the email.

Solution:

1. Make sure the coupon code is enabled in the email settings of your trigger.
2. If there are no available Cart Price Rules in the email settings, make sure that the option **Use Auto Generation** is checked at least in one rule.

   Note

   The **Use Auto Generation** checkbox is available in the Cart Price rules only when you choose the **Specific Coupon** option at the **Coupon** field of your Cart Price Rule.

The related, upsell or cross sell products are not displayed

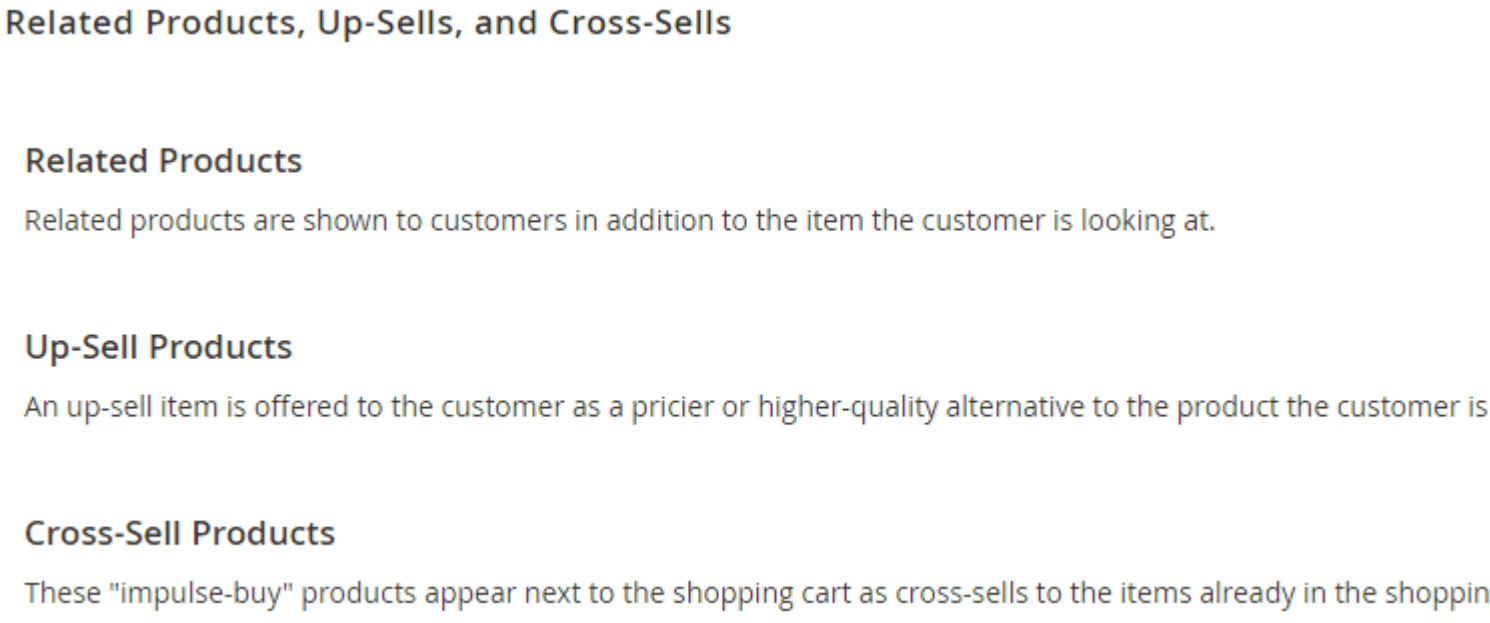
The block with the selected cross-sell, related, or upsell products is not displayed in the email.

The products for the cross-sell block are selected based on the products associated with the customer's orders or shopping cart.
Before using one of the types as the source for cross-sell block, make sure that an appropriate type of product is configured.

Note
In the preview emails, the module shows random products in the cross-sell block.

Solution:

Make sure your products have the cross-sell, related, or upsell products:



1. Open one of your products in the admin panel in edit mode.
2. Scroll down and expand the **Related Products, Up-Sells, and Cross-Sells** section.
3. You'll see products there if you have configured these settings earlier, but if there are no products, you should configure them.

For more information refer to Magento documentation: https://docs.magento.com/m2/ce/user_guide/catalog/settings-advanced-related-products.html. Also you may find interesting to insert our Related Products for Magento 2 in your emails.

How do I translate an email template?

There are several ways to translate emails:

1. Separate email templates per language:
    1. Create a new email template in the desired language.
    2. Create a separate trigger per store and for emails using templates in the corresponding language.
2. Use the Magento translation mechanism:
   For example, you want to translate the text **Shop Now** in the **Welcome** email template. To translate it, copy the source string with this text to the CSV file located in your theme directory. The translation may look as follows:

Example

```
<a href="{{ store.store_url }}">Shop Now</a>
<a href="{{ store.store_url }}">Shop Right Now</a>
```

Or if you want translate the subject:

Example

```
Hello {{ customer_name }}!, Hola {{ customer_name }}!
```

Note
If the string you want to translate contains commas, then you should wrap the translation string with the double quotation mark ("). But if the string in addition to comma contains a double quotation mark, then you should replace all double quotes in the original text with the single quotation mark (').

After extension update php code in the templates stopped working

According to the Magento security policies starting with the **module-email** version 2.1.28 and **module-email-designer** version 1.1.31 ability to use php code in the templates was disabled.

If this ability is vital for your store, it is possible to revert the function code located at the file `vendor/mirasvit/module-email-designer/src/EmailDesigner/Service/TemplateEngine/Php.php`

- Should be modified function **`getHtml`** with the next code:

```
private function getHtml($tplPath)
{
    ob_start();
    include $tplPath;
    $html = ob_get_clean();

    return $html;
}
```

After a Magento or extension update, the event log is empty and emails don't send.

To fix this, open the `/app/etc/env.php` file and check if you have the consumers section.

```
...
    'cron_consumers_runner' => [
        'cron_run' => false,
        'max_messages' => 20000,
```

```
        'consumers' => [
            'consumer1',
            'consumer2'
        ]
    ],
...
```

If you have it, you should add our job `'mirasvit.event.mq.register'` to the list. Once added, run the command `php -f bin/magento queue:consumers:start mirasvit.event.mq.register` to start it.

Example:

```
...
    'cron_consumers_runner' => [
        'cron_run' => false,
        'max_messages' => 20000,
        'consumers' => [
            'product_action_attribute.update',
            'inventory.source.items.cleanup',
            'mirasvit.event.mq.register'
        ]
    ],
...
```

How to upgrade the extension

To upgrade the extension, take the following steps:

1. Back up your store's database and web directory.
2. Log in to the SSH console of your server and navigate to root directory of the Magento 2 store.
3. Run the command `composer require mirasvit/module-email:* --update-with-dependencies` to update the current extension with all dependencies.

   Note

   In some cases, the command above is not applicable, or you are unable to update just the current module, and need to upgrade all Mirasvit modules in a bundle. In this case, the command above will have no effect.

   Instead, run the `composer update mirasvit/*` command. It will update all Mirasvit modules installed in your store.

4. Run the command `php -f bin/magento setup:upgrade` to install updates.
5. Run the command `php -f bin/magento cache:clean` to clean the cache.

6. Deploy static view files

   ```
   rm -rf pub/static/*; rm -rf var/view_preprocessed/*; php -f
   bin/magento setup:static-content:deploy
   ```

Disabling Extension

Temporarily Disable

To temporarily disable the extension, please take the following steps:

1. Log in to the SSH console on your server and navigate to the root directory of the Magento 2 store.
2. Run the command `php -f bin/magento module:disable Mirasvit_Email Mirasvit_EmailDesigner Mirasvit_EmailReport` to disable the extension.
3. Log in to the Magento backend and refresh the store cache (if enabled).

Extension Removal

To uninstall the extension, please take the following steps:

1. Log in to the SSH console on your server and navigate to the root directory of the Magento 2 store.
2. Run the command `composer remove mirasvit/module-email` to remove the extension.
3. Log in to the Magento backend and refresh the store cache (if enabled).

Change Log

2.5.2

(2023-08-15)

Fixed

- Fixed the issue with cancelation events when multiple abandoned carts processed in one cron run

---

2.5.1

(2023-08-02)

Fixed

- fixed an error in the email templates: "Creation of dynamic property ... $context is deprecated"
- fixed an error: "Undefined variable $result in /vendor/mirasvit/module-email/src/Email/Model/Config/Source/TriggerSource.php"

2.5.0

(2023-07-06)

Improvements

- Added protection against deleting themes used in templates
- Added protection against deleting templates used in created campaigns
- After clicking on the email unsubscribe button, the unsubscribe message will not be displayed on the main page, but on a separate page

Fixed

- Fixed an issue with not working trigger 'active from' and 'active to' dates
- Fixed an error when trying to send a test email
- Fixed an issue with not updating email template after selecting 'Auto Refresh' button
- Fixed an error when trying to send email 'Argument [#1]() () must be of type Magento\Sales\Model\Order\Address, null given'

2.4.9

(2023-06-15)

Fixed

- Undefined method on the unsubscription grid page (admin)

2.4.8

(2023-06-13)

Fixed

- Issue with processing events, when trigger is not assigned to store

---

2.4.7

(2023-06-05)

Improvements

- Revamped the campaigns overview interface
- Changed the Queue and Event interfaces to UI components
- Added the ability to validate events based on trigger audience rules

---

2.4.6

(2023-03-27)

Fixed

- Fixed issue with abandoned cart emails when triggered Cancellation events didn't cancel sending emails

2.4.5

(2023-02-07)

Fixed

- Canceling email only for events related to the current entity (previously all emails assigned to email were canceled)
- PHP8.1 compatibility issue. Mirasvit\Email\Model\Unsubscription::unsubscribe. ($triggerId) must be of type ?int, string given

---

2.4.3

(2022-10-13)

Improvements

- Small code improvements for Magento Marketplace

2.4.2

(2022-10-12)

Improvements

- small changes for Magento Marketplace

2.4.1

(2022-09-16)

Fixed

- updated button styles in the preinstalled templates
- fixed an issue with the missing products in the wishlist emails
- fixed an issue with the changing 'Delivery Time Delay' period in the email template

---

2.4.0

(2022-08-16)

Improvements

- Support of Magento MQ

Fixed

- PHP8.1 compatibility issue

2.3.5

(2022-08-09)

Fixed

- fixed an issue with the variables errors in the email templates

---

2.3.4

(2022-07-15)

Fixed

- Fixed the issue with processing events
- Fixed an issue with using number_format function

2.3.3

(2022-07-13)

Fixed

- fixed the error with base64_encode function
- fixed the error with sending test emails
- fixed the error with redirecting email links

2.3.2

(2022-06-28)

Fixed

- Fixed the issue with emails not being sent

2.3.1

(2022-06-23)

Fixed

- Fixed the issue with errors related to return types

2.3.0

(2022-06-22)

Improvements

- Remove db_schema_whitelist.json
- Code refactoring
- module-email-design and module-email-report merged into module-email
- Removed service and repository interfaces
- Updated admin interface

Fixed

- Fixed the issue with variables (docblock)

2.2.4

(2022-06-20)

Improvements

- remove db_schema_whitelist.json

Fixed

- Fixed the issue with variables (docblock)

2.2.3

(2022-05-25)

Fixed

- Fixed an issue with the page styles on Magento 2.4.4
- Issue with decalrative schema

2.2.0

(2022-05-23)

Improvements

- Migrate to declarative schema

2.1.45

(2020-11-03)

Fixed

- fixed an issue with the missing test email send button at the Magento 2.4.x
- fixed an issue with the checkout processing
- fixed an issue with the event grid filtering

2.1.44

(2020-09-07)

Fixed

- fixed an issue with the checkout processing M2.4

2.1.43

(2020-07-29)

Improvements

- Support of Magento 2.4

2.1.42

(2020-06-18)

Fixed

- fixed an issue with the emails delivery via 'Send' action button at the Mail Log (Queue)

2.1.41

(2020-05-27)

Fixed

- show warning message before sending test email that trigger 'Event' was not specified
- fixed an issue with the showing html text in the emails

2.1.40

(2020-03-13)

Fixed

- added Mageplaza Smtp compatibility

2.1.39

(2020-03-04)

Fixed

- fixed an issue with incorrect redirect urls of the cross-sell products

---

2.1.38

(2020-02-26)

Fixed

- fixed an issue with incorrect redirect urls of the cross-sell products

---

2.1.37

(2020-02-18)

Features

- added new template coupon variable 'Get coupon expiration date'

2.1.36

(2020-02-10)

Features

- added new template url variable 'Get URL used to create a new checkout cart for reorder'

2.1.35

(2020-01-08)

Improvements

- added functionality to manage emails subscriptions and unsubscriptions at Marketing > Follow Up Email > Unsubscription List
- added recipient email validation before sending

2.1.34

(2020-01-08)

Improvements

- added mass action for the events: 'Reset & Process'

2.1.33

(2019-12-16)

Improvements

- added trigger event "Change Group"

2.1.32

(2019-12-05)

Fixed

- show only in stock and enabled products at the template cross sell block

---

2.1.31

(2019-12-03)

Improvements

- added compatibility with the Mageplaza Smtp

Fixed

- fixed an issue with the active coupon status after his expiration date

---

2.1.30

(2019-11-05)

Fixed

- Issue with call to undefined method EmailMessage::setSubject()

2.1.29

(2019-10-31)

Fixed

- Compatibility with Magento 2.3.3

2.1.28

(2019-08-14)

Fixed

- EQP code refactoring

2.1.27

(2019-07-30)

Fixed

- Conflict with mageplaza smtp

2.1.26

(2019-07-11)

Fixed

- fixed an issue with the events processing

2.1.25

(2019-06-26)

Fixed

- fixed an issue with the incorrect displaying 'Scheduled At' emails date

---

2.1.24

(2019-04-22)

Fixed

- Compatibility with Ebizmarts Mandrill

2.1.23

(2019-03-27)

Fixed

- fixed an issue with the incorrect 'Sender From' name at the emails

---

2.1.22

(2019-03-22)

Fixed

- fixed an issue with missing Google Analytics parameters at the email urls

---

2.1.21

(2019-03-18)

Fixed

- Compatibility with M2.1
- fixed an issue with the sending test emails
- Issue deleting trigger' email chain from campaign

2.1.20

(2019-02-21)

Fixed

- Fix compilation error for Magento 2.1

Documentation

- Update path to Event settings

2.1.19

(2019-02-11)

Fixed

- Issue saving active to/from dates for trigger #116

2.1.18

(2019-02-08)

Fixed

- Fix error blocking Magento installation
- View email in browser shows wrong email
- Possible issue with sending emails. Error message like "The recipient address < ...> is not a valid RFC-5321"

2.1.17

(2018-12-24)

Fixed

- Error after navigation by link through email

2.1.16

(2018-12-10)

Fixed

- Trigger's rule edit section does not work

2.1.15

(2018-11-30)

Improvements

- support for M2.3

2.1.14

(2018-10-22)

Fixed

- Wrong emails report
- CSS issue

2.1.12

(2018-10-03)

Improvements

- Improved Reports

2.1.11

(2018-09-26)

Fixed

- Test emails continue to be sent automatically
- Error opening a campaign when email chain template has been removed
- DI compilation error

Documentation

- changes to getCustomerName callout

2.1.10

(2018-09-06)

Fixed

- Do not enqueue email if there is no recipient email address

---

2.1.9

(2018-09-05)

Fixed

- Mass send function does not work in Mail Log

Documentation

- Troubleshoot for translation of email templates

2.1.8

(2018-08-30)

Fixed

- Test email is not sent, when admin URL does not match front URL

2.1.7

(2018-08-29)

Fixed

- unsubscribe variable does not work

Documentation

- troubleshoot for coupon code and block with cross-sell products

---

2.1.6

(2018-08-03)

Fixed

- Campaign edit page is not loaded after update: emaildesigner keyword added 2 times

---

2.1.5

(2018-07-31)

Fixed

- Manage campaigns pages is not loaded on Magento 2.1.0 version

2.1.4

(2018-07-13)

Features

- Ability to disable Magento Newsletter's Success Email Template

Documentation

- How to disable the default Magento Newsletter's Success Email Template

---

2.1.3

(2018-07-09)

Improvements

- Refactoring

2.1.2

(2018-07-09)

Fixed

- fixed an issue with incorrect output of Facebook and Twitter Urls in emails

---

2.1.1

(2018-07-03)

Improvements

- Properly retrieve cross-sell products
- Ability to translate email template text via i18n CSV files
- Mute 'payment' error when rendering native Magento email templates and log them instead
- Use latest order information in test and preview emails

Fixed

- Properly create campaigns from template
- Email template editor loses focus on Safari
- Avoid error for test and preview emails during product image rendering
- Properly render native Magento email templates

- Error on 'setup:di:compile' on Magento versions prior to 2.2 (affects from 1.1.20)

2.1.0

(2018-06-28)

Features

- Ability to use native Magento email templates

Improvements

- Improve product image select algorithm in email templates

Fixed

- Product price change triggers the Abandoned Cart event of a cart which contains this product

---

2.0.4

(2018-05-31)

Fixed

- Invalid trigger link on Queue Preview page
- Cannot save trigger: properly handle resource permissions

2.0.3

(2018-05-24)

Fixed

- Error during running command 'setup:update' after module installation

---

2.0.2

(2018-05-17)

Fixed

- Error opening campaigns on Magento prior to 2.2 versions: 'Element modal is not expected'

2.0.1

(2018-05-11)

Fixed

- Error during update: use correct version of Email Designer module
- Error during compiling DI files

2.0.0

(2018-05-11)

Features

- New major update: UI, Campaigns, Statistic and other improvements

Documentation

- GDPR Compliance Tips

1.1.25

(2018-04-27)

Improvements

- Compatibility with latest email report module

Fixed

- Error previewing templates - negative offset in SQL
- Issue saving active to/from dates

1.1.24

(2018-03-02)

Fixed

- remove 'custom' area from the default email design

Documentation

- Update installation instruction

1.1.23

(2018-02-27)

Fixed

- Emails are sent to all store views instead of chosen only (affects since 1.1.19)

1.1.22

(2018-02-16)

Fixed

- Do not use same coupon code across different emails of the same trigger #60

Documentation

- correct command to update module
- product view event
- liquid is supported with Email Themes now
- Documentation for liquid syntax #45

1.1.21

(2018-02-12)

Improvements

- Move Email Theme editor to liquid syntax mirasvit/module-email-designer#4
- Liquid syntax allows to avoid blocking of templates rendering by server extension "ModSecurity"

Bugfixes

- Do not show coupon block in emails even if it disabled

1.1.20

(2018-02-09)

Features

- New condition 'Recipient does not have emails for triggers' #53
- Product View event #33
- Default trigger for Product View event #58
- Email template for Product View event #54

Bugfixes

- Cross sell products are not displayed in emails #55

Improvements

- Register only active events mirasvit/module-event#9

1.1.19

(2018-01-30)

Improvements

- Process events in realtime with message-queue #19

1.1.18

(2018-01-26)

Bugfixes

- Solve error during performing setup:di:compile command
- Fix error for restore checkout method - use correct alias for Quote model

---

1.1.17

(2018-01-24)

Improvements

- Create a new cart if the previous one has already been finished earlier #46

1.1.16

(2018-01-18)

Bugfixes

- test emails are not sent

Documentation

- Update for manual installation instruction

1.1.15

(2018-01-16)

Features

- new UI for working with variables in emails mirasvit/module-email-designer
- Liquid template engine and menu with available variables

Bugfixes

- correctly emulate store to retrieve correct product URL
- fixed an issue with the incorrect restore cart url redirect from email ()

Improvements

- add liquid variables
- Set queue status to 'Error' if it throws errors during sending

---

1.1.10

(2017-12-11)

Fixed

- Use product URLs with a correct store base URL
- Show global Follow Up Email settings for website and store view
- Use sender name and email from store scope when available
- Do not duplicate trigger on save
- Fixed an issue with the incorrect restore cart url redirect from email

---

1.1.9

(2017-12-01)

Fixed

- Validate only triggering events on the event check stage

1.1.8

(2017-11-23)

Improvements

- Rename column's title
- Remove old files

1.1.7

(2017-11-03)

Fixed

- Ignore test events

1.1.6

(2017-11-01)

Fixed

- Problem with trigger excluded weekdays setting

1.1.5

(2017-11-01)

Fixed

- Installation error
- Error opening trigger listing due to loading trigger rules

Documentation

- Update installation docs

1.1.4

(2017-10-30)

Fixed

- Use correct MySQL column type for updated/created at columns

---

1.1.3

(2017-10-30)

Fixed

- Compatibility with PHP > 7.0.0

1.1.2

(2017-10-27)

Fixed

- Properly migrate rules

1.1.1

(2017-10-27)

Fixed

- Disable console command
- Remove unused classes
- Update dates

1.1.0

(2017-10-26)

Improvements

- Add Sample Triggers
- Update Trigger's Rules On Event Changing
- Show Only Events Available For Follow Up Email
- Integrate With Module-event

Fixed

- Remove Unnecessary Files
- Correct Trigger Link In Queue View
- Ignore Shopping Carts That Have Associated Orders
- Filter Payment Methods Without Labels

1.0.57

(2017-09-28)

Fixed

- Fix error during compilation

1.0.56

(2017-09-27)

Improvements

- Compatibility with Magento 2.2

1.0.55

(2017-09-05)

Fixed

- Properly create Administrator triggers

1.0.54

(2017-09-04)

Documentation

- Improve documentation

1.0.53

(2017-09-04)

Fixed

- Compatibility with Magento 2.2.0rc

1.0.52

(2017-09-01)

Fixed

- Fix incorrect dependency error during compilation

1.0.51

(2017-09-01)

Improvements

- UI improvements

1.0.50

(2017-08-31)

Fixed

- Use correct source of events

1.0.49

(2017-08-31)

Documentation

- Delete old information

1.0.47

(2017-08-30)

Improvements

- Improve UI

1.0.46

(2017-08-11)

Documentation

- Mail Log

1.0.45

(2017-08-10)

Improvements

- Cancel emails whose email chain was removed from trigger
- Add identifier to cross sell block

Fixed

- Allow deselect the cancellation event
- correctly define email delay when sending email at specific time using 'at' option

Documentation

- Documentation for coupon code expiration date and displaying only first item

---

1.0.44

(2017-07-19)

Fixed

- YAML require

1.0.43

(2017-07-14)

Improvements

- Ability to delete email queues

1.0.42

(2017-07-14)

Bugfixes

- Escape colon character in yaml files

Improvements

- Performance (added indexes to db tables)

1.0.41

(2017-06-29)

Documentation

- Description of template methods and Mail Log section

1.0.40

(2017-06-29)

Features

- New condition 'Product Attribute Value Comparison'

Improvements

- Show more information about email queue cancellation event

1.0.39

(2017-06-21)

Bugfixes

- Compatibility with Ebizmarts Mandrill

1.0.38

(2017-06-20)

Bugfixes

- Problem with serializing email arguments
- Do not add 'test' subject for emails sent manually through email queue

Improvements

- Display cancellation event key

1.0.37

(2017-06-13)

Fixed

- Issue with queue

1.0.36

(2017-06-12)

Bugfixes

- Email is not sent to admin if multiple email addresses specified

---

1.0.35

(2017-05-12)

Features

- New conditions for order condition group

Bugfixes

- Error displaying cross sell products

1.0.34

(2017-05-04)

Bugfixes

- Properly place fragment part in generated URLs
- Fix issue with the product subselection condition

1.0.33

(2017-05-03)

Bugfixes

- Compatibility with the versions before introducing ability to send emails every X period (affects since 1.0.32)
- Add cross-sell html to base theme

1.0.32

(2017-04-28)

Features

- Ability to send emails every X days/weeks/months/years

Bugfixes

- Header is displayed like a field

Improvements

- Order event, use customer name from address if it's empty in order

---

1.0.31

(2017-04-19)

Bugfixes

- Show correct time values at the Mail Log message

Improvements

- Separate cron group for Follow Up Email extension

1.0.30

(2017-04-12)

Features

- New method 'getResumeUrl' to automatically login customers

1.0.29

(2017-03-29)

Features

- Ability to validate concrete number of products in cart/order
- New event 'Customer Review Approved'

1.0.28

(2017-03-24)

Bugfixes

- Fix error while restoring shopping cart

1.0.27

(2017-03-20)

Features

- Ability to validate events in mass action according trigger's rules

Bugfixes

- Correctly set numbers of affected records in response messages
- Fix issue with the condition "Shopping Cart products available for purchase"

1.0.26

(2017-03-17)

Bugfixes

- Trigger email chains reset after changing status of triggers in mass action

---

1.0.25

(2017-03-16)

Improvements

- Check order object before processing it

1.0.24

(2017-03-07)

Bugfixes

- Fix issue with 'Shipping Address' rules

1.0.23

(2017-03-02)

Bugfixes

- Use correct store ID for new product review event (affects all)

1.0.21

(2017-02-20)

Fixed

- Fixed an issue with abandoned cart trigger

1.0.20

(2017-01-30)

Bugfixes

- Correctly display email queue view page (affects all)

1.0.19

(2017-01-25)

Bugfixes

- Fixed problem when using SKU condition in products selection

---

1.0.18

(2017-01-23)

Fixed

- Fixed an issue with utm_ tags

1.0.17

(2017-01-06)

Fixed

- Fixed an issue with restoring cart

1.0.16

(2017-01-05)

Features

- Implemented ability to send follow up emails only to specified (administrator) email. Is useful for receive internal reminders (new review, order, customer etc)

Fixed

- Fixed an issue with abandoned cart event

1.0.15

(2016-12-23)

Improvements

- Triggers grid

1.0.14

(2016-12-16)

Bugfixes

- Fixed an issue with registering the event 'Order obtained new status' (affects all)
- Fixed an issue with Review Request template (error if product already removed)

Improvements

- Ability to use product attributes in rules

1.0.13

(2016-09-14)

Fixed

- Limit number of cart rules

1.0.12

(2016-09-08)

Fixed

- Compatibility issue
- Set attribute element as a text and add available options for region condition

---

1.0.11

(2016-08-11)

Improvements

- New rule condition 'Shopping cart products available for purchase'

1.0.10

(2016-07-28)

Fixed

- Fixed an issue with store base url

1.0.8

(2016-06-24)

Fixed

- Compatibility with Magento 2.1

1.0.7

(2016-05-27)

Improvements

- Added store filter to events grid

1.0.6

(2016-05-20)

Improvements

- Support of different mail transfer agents

Fixed

- Fixed issue with SalesRule naming (after update to 2.0.6)
- Fixed issue with multi-store emails
- Changed external links params (code to hash)
- Issues with rules
- Fixed an issue with empty Restre Cart url
- Fixed and issue with cross sells
- Fixed possible issue with non-secure url for ajax capture

1.0.4

(2016-04-11)

Fixed

- Fixed an issue with menu

1.0.3

(2016-03-28)

Improvements

- Added new tab to customer edit page with FUE emails
- Ability to setup coupon generation rules (length, prefix, suffix, dash every X chars)
- Check coupon type (fixed or auto generation)
- Improved clean history (logs) feature
- Improved current time (local/gmt) validation

- i18n

Fixed

- Fixed an issue with cross-sell products
- Fixed wrong link in menu
- Fixed an issue with wrong link to Settings

Documentation

- Updated installation steps

1.0.2

(2016-02-18)

Fixed

- Fixed an issue with cronjob (wrong path to class)
- Fixed an issue with parse error in crontab.xml

Improvements

- Added new column to trigger grid with general information
- Added ability to preview cross-sell products in template preview

Documentation

- Added base user manual

Submodule mirasvit/module-email-designer

1.0.15

(2017-10-30)

Fixed

- ?ompatibility with PHP > 7.0.0

1.0.14

(2017-09-04)

Fixed

- Fix for compatibility with Magento 2.2.0rc

1.0.13

(2017-08-31)

Improvements

- Create repository for templates
- Method 'getItemOptions' for displaying options selected for ordered item

---

1.0.12

(2017-05-29)

Improvements

- Methods to retrieve wishlist products
- Fallback mechanism for method 'getCustomerName()' in order context

---

1.0.11

(2017-04-28)

Improvements

- Fallback mechanism for method 'getCustomerName()' in order context

---

1.0.10

(2017-03-16)

Bugfixes

- Fix some variables do not exist until they explicitly called (affects all)

1.0.9

(2017-01-27)

Bugfixes

- Display value for method 'getCustomerEmail' in preview emails (affects all)

1.0.8

(2017-01-26)

Bugfixes

- Display coupon code in preview mode (affects all)
- Fixed an issue with image path

1.0.6 1.0.7

(2016-09-08)

Fixed

- Fixed an issue with image URL

1.0.5

(2016-06-21)

Fixed

- Fixed an issue with hard coded store id

1.0.3 1.0.4

(2016-05-06)

Fixed

- Fixed an issue with multi-store

1.0.2

(2016-04-18)

Fixed

- Fixed an issue during compilation (setup:di:compile-multi-tenant)
- Fixed an issue with fatal error when orders/carts are not exists

Improvements

- Ability to use wishlists in template
- i18n
- Showing php error, if template syntax not correct

Submodule mirasvit/module-report

1.2.27

(2017-12-07)

Fixed

- filters by "Customers > Products" and "Abandoned Carts > Abandoned Products" columns

---

1.2.26

(2017-12-06)

Fixed

- filter by "Products" column

1.2.25

(2017-12-05)

Fixed

- Issue with active dimension column

1.2.24

(2017-11-30)

Fixed

- Issue with export in Magento 2.1.8

1.2.23

(2017-11-27)

Fixed

- Issue with "Total" value of non-numeric columns

1.2.22

(2017-11-15)

Fixed

- Issue with export to XML

1.2.21

(2017-11-03)

Fixed

- Properly replicate temporary tables
- An issue with builing relations
- Issue with finding way to join tables

1.2.20

(2017-10-30)

Fixed

- An issue with sales overview report when customer segments used

---

1.2.19

(2017-10-30)

Fixed

- Issue with export to CSV (Magento 2.1.9)

1.2.18

(2017-10-26)

Fixed

- Issue with long replication

1.2.17

(2017-10-20)

Fixed

- Fixed css bug
- Compare for leap year

1.2.16

(2017-09-28)

Fixed

- Compatibility with php 7.1.9

1.2.15

(2017-09-26)

Fixed

- M2.2

1.2.14

(2017-09-18)

Fixed

- Fix report email notification using 'Send Now' function

1.2.13

(2017-08-09)

Fixed

- Conflict with other reports extensions

1.2.12

(2017-08-02)

Improvements

- New Report Columns

1.2.11

(2017-07-19)

Fixed

- Display option labels instead of values for dashboard widgets

---

1.2.10

(2017-07-12)

Fixed

- Issue with Eav attributes

1.2.9

(2017-07-11)

Improvements

- New Charts

1.2.8

(2017-06-21)

Fixed

- Proper filter product details report by current product ID

1.2.7

(2017-06-21)

Improvements

- Refactoring

1.2.6

(2017-06-01)

1.2.5

(2017-05-31)

Improvements

- Added field to relation

1.2.4

(2017-05-15)

Fixed

- Issue with column ordering

1.2.3

(2017-05-04)

Bugfixes

- Fixed an issue with compound columns of type simple

Improvements

- Changed default multiselect control to ui-select
- Chart resizing

1.2.2

(2017-03-21)

Improvements

- Performance

Fixed

- Fixed an issue with join returing customers

1.2.1

(2017-03-06)

Improvements

- Disabled wrong filters for day/hour/month/quarter/week/year

Fixed

- Fixed an issue with table joining
- Fixed an issue with filters
- Issue with rounding numbers in chart

1.2.0

(2017-02-27)

Fixed

- Minor issues
- Fixed an issue with replication

1.1.14

(2017-01-31)

Fixed

- Dashboard

1.1.12

(2017-01-25)

Fixed

- Backward compatibility
- Fixed an issue with bookmarks

1.1.11

(2017-01-20)

Fixed

- fixed an issue with tz

1.1.9, 1.1.10

(2017-01-13)

Fixed

- Fixed an issue with timezones
- Fixed an issue with dates

1.1.7, 1.1.8

(2016-12-15)

Fixed

- Fixed an issue in toolbar
- Fixed an issue with date filter

1.1.6

(2016-12-09)

Improvements

- Compatibility with M2.2

1.1.5

(2016-09-27)

Fixed

- Fixed an issue with moment js

1.1.4

(2016-09-13)

Fixed

- Removed limit on export reports (was 1000 rows)

1.1.3

(2016-09-05)

Improvements

- Changed product type column type

1.1.2

(2016-09-01)

Improvements

- Added Product Type column

1.1.1

(2016-08-15)

Fixed

- Fixed an issue with exporting

1.1.0

(2016-07-01)

Fixed

- Rename report.xml to mreport.xsd (compatiblity with module-support)

1.0.4

(2016-06-24)

Fixed

- Compatibility with Magento 2.1

1.0.3

(2016-05-31)

Fixed

- Fixed an issue with currency symbol

1.0.2

(2016-05-27)

Fixed

- Add store filter

1.0.1

(2016-05-25)

Fixed

- Removed font-awesome

1.0.0

(2016-05-19)

Improvements

- Export
- Refactoring
- Table join logic

Fixed

- Fixed an issue with joining tables
- Chart - multi columns

Submodule mirasvit/module-email-report

1.0.4

(2017-10-30)

Fixed

- compatibility with PHP > 7.0.0

1.0.3

(2017-09-19)

Fixed

- require correct version of module-report

1.0.2

(2017-09-04)

Fixed

- fix for compatibility with Magento 2.2.0rc

1.0.1

(2017-09-01)

Fixed

- Disable unfinished report provided by module

1.0.0

(2017-08-30)

Features

- integrate with Follow Up Email

0.0.0-alpha1

(2016-04-18)

Features

- Ability to manage email campaigns

Submodule mirasvit/module-message-queue

1.0.2

(2017-11-21)

Fixed

- compatibility with Magento EE

1.0.1

(2017-10-31)

Fixed

- do not enqueue messages if 'queue' table is not created yet

---

1.0.0

(2017-10-31)

Feature

- RabbitMQ and MySQL drivers

Publish message

```
/** @var \Mirasvit\Mq\Api\PublisherInterface $publisher */
$publisher = $this->objectManager->create('Mirasvit\Mq\Api\PublisherInterface')

$publisher->publish('mirasvit.event', [microtime(true)]);
```

Listen

di.xml

```
<type name="Mirasvit\Mq\Api\Repository\ConsumerRepositoryInterface">
    <arguments>
        <argument name="consumers" xsi:type="array">
            <item name="notificator" xsi:type="array">
                <item name="queue" xsi:type="string">mirasvit.event</item>
                <item name="callback" xsi:type="string">Mirasvit\Testing\Model\
            </item>
        </argument>
    </arguments>
</type>
```

Submodule mirasvit/module-event

1.1.10

(2017-12-01)

Fixed

- Properly retrieve attribute values
- correctly detect 'Product / QTY Reduced' event

1.1.9

(2017-11-24)

Fixed

- Missing customer_name parameter in the 'customer birthday' event

---

1.1.8

(2017-11-23)

Fixed

- Properly validate total count/qty of products in cart/order
- Set Customer: Group condition as multiselect
- Register 'order status change' event only when status really changed

---

1.1.7

(2017-11-22)

Fixed

- use customer email as unique key for newsletter events

1.1.6

(2017-11-01)

Improvements

- Move error event to module-notificator

1.1.5

(2017-10-31)

Fixed

- register method may return boolean false

1.1.4

(2017-10-30)

Fixed

- Properly load customer model

1.1.3

(2017-10-30)

Fixed

- error with review related events

1.1.2

(2017-10-26)

Improvements

- Customer condition 'Last Activity'
- Handle API errors
- Order condition 'Order Updated At Time (24H format)'
- move email capture function from follow up email to module event
- Add all follow up email conditions
- Save customer_name, customer_email values with event registration

Fixed

- Error event

1.1.1

(2017-10-19)

Fixed

- Bump version number

1.1.0

(2017-10-19)

Features

- Shipping Address Conditions
- Event 'Customer Birthday'
- Event 'Review Approved'
- Event 'New item added to wishlist'
- Event 'Wishlist shared'

Fixed

- rename attribute code

1.1.0-beta6

(2017-10-12)

Features

- Event 'Product QTY Decreased'
- new event 'New System Notification'
- Ability to add groups of conditions to events
- Product subselection conditions
- Event 'Admin Logged In'
- Event 'Failed Login Admin'
- Ability to add custom attributes and conditions to EventData

Improvements

- Last heartbeat schedule condition
- Add Store Event Data with store related conditions

Fixed

- add custom attributes only if they added

1.1.0-beta5

(2017-09-27)

Improvements

- compatibility with Magento 2.2

1.1.0-beta4

(2017-09-27)

Improvements

- expand all params specified in the getEventData method

1.1.0-beta3

(2017-09-19)

Fixed

- do not register events before module installation

1.1.0-beta2

(2017-09-18)

Fixed

- Order status event

1.1.0-beta1

(2017-09-18)

Improvements

- convert all events to plugins

1.0.1

(2017-06-15)

Fixed

- DI
- Pool
