Spaaza Magento 2 Module

Contents

Introduction

The Spaaza Magento Module connects Magento 2 to Spaaza. It provides basic synchronisation via the Spaaza API, an internal API to handle vouchers in the checkout and an external (REST) API to authenticate and synchronise customer data.

Basic synchronisation includes:

- Customer data
- Customer addresses
- Orders
- Credit memos

More details about the Spaaza API are available at the Spaaza documentation site at https://docs.spaaza.com .

More details about how the Spaaza Magento module extends the Magento REST API are available in the REST API document

The version history of the module is available in the versions document

Compatibility

Currently, the module has been tested with Magento 2.3.3.

Configuration Options

The Spaaza Magento module can be configured through the Magento admin interface. Configuration options are found via Stores » Configuration » Customers » Spaaza Loyalty. The following configuration options are available:

- **General**
    - **Synchronous Sync Enabled** (recommended setting: Yes) - synchronisation between Magento 2 and the Spaaza API which waits for the response from Spaaza before proceeding. For example, when a customer logs into Magento 2, a request is sent to Spaaza to pull any new customer data. It is recommended only to disable this setting when testing or debugging.

    - **Asynchronous Sync Enabled** (recommended setting: Yes) - synchronisation between Magento 2 and the Spaaza API which does not require an API response from Spaaza and is executed in the background using the cron scheduler.
    - **Debug Log Enabled** (recommended setting: No) - enable debug logging from the Spaaza module. It is recommended to enable this during debugging.

- **API Client**

    - **Base URL** (mandatory) - the base URL of the Spaaza API. This value is supplied in your Spaaza configuration options.
    - **App Hostname** (mandatory) - each Spaaza loyalty programme has a separate configuration setup, which has a unique name in hostname format. This will usually be in the form of retailername.spaaza.com. This value is supplied in your Spaaza configuration options.
    - **Chain ID** (mandatory) - each Spaaza retailer has a separate environment, known as a "Chain". A chain has a numeric ID within the Spaaza system. This is used to recognise requests made from Magento 2 to the Spaaza API. This value is supplied in your Spaaza configuration options.
    - **Auth Key** (mandatory) - Spaaza uses an OAuth-based API key system for requests made from Magento 2 to the Spaaza API. This field denotes the ID of the key. This value is supplied in your Spaaza configuration options.
    - **Auth Secret** (mandatory) - Spaaza uses an OAuth-based API key system for requests made from Magento 2 to the Spaaza API. This field denotes the secret of the key. This value is supplied in your Spaaza configuration options.
    - **Connect Timeout** (mandatory, defaults to system value) - The timeout in seconds to establish connections to the Spaaza API.
    - **Read Timeout** (mandatory, defaults to system value) - The timeout in seconds within which the response to a request to the Spaaza API must be received.

- **Requests Queue**

    - **Keep Successful Requests** (mandatory, defaults to system value) - the number of days for which a record of a successful request to the Spaaza API is stored by Magento.
    - **Keep Failed Requests** (mandatory, defaults to system value) - the number of days for which a record of a failed request to the Spaaza API is stored by Magento.
    - **Requests To Send Per Batch** (mandatory, defaults to system value) - the number of queued requests to send each time the scheduled cron job runs.
    - **Maximum Attempts** (mandatory, defaults to system value) - the number of times to attempt a request to the Spaaza API before the request is abandoned.

- **Vouchers**

    - **Label for Total** (optional) - in the checkout and orders part of the total can be made up of Spaaza-administered vouchers. This string is used to describe that amount in the interface.
    - **Show 0 Total** (mandatory) - whether to show vouchers in the checkout and order if they are present but their discount value is 0.

Request Queue

Most outgoing API requests from the Magento module to Spaaza will be performed immediately. A request also gets added to a so-called request queue, so it will be picked up by the cron process if it fails.

You can see the queue items in the Magento backend under System » Tools » Spaaza Requests. At this moment, it only shows basic information about the requests like the path, creation timestamp and the status. If you need more detailed information, you'll have to inspect the database.

Extension Attributes

The Spaaza module makes extensive use of *Extension Attributes*. All data that it needs to set in Magento models is stored using Extension Attributes. You can retrieve and set the data using the `getExtensionAttributes()` method of the respective models.

The following models have a 'SpaazaData' Extension Attribute:

- **Customer**
  To store Spaaza related attributes (opt-ins: loyalty program, mailing list, printed mailing list) and the Spaaza user id and member number.
- **Order**
  To store the Spaaza basket id once it has been sent to Spaaza, the voucher discounts and the voucher distribution to use for calculating credit memo totals.
- **Invoice**
  To store the voucher discounts.
- **Credit memo**
  To store the voucher discount totals and the Spaaza basket ID once it has been sent to Spaaza.
- **Quote address**
  To store the voucher discount totals and to know what vouchers have been applied to a shopping cart.

You can both set and get the Spaaza data, although for most data it's more logical to only get it. For the Spaaza opt-ins however, you can use these methods to change data. For example, you could use this snippet of code to change an opt in:

```
$customer = $customerRepository->get('user@example.com');

$spaazaData = $customer->getExtensionAttributes()->getSpaazaData();
$spaazaData->setProgrammeOptedIn(true);
$customerRepository->save($customer);
```

Customer data synchronisation

Every time a customer logs in, the module tries to pull the most recent customer data from Spaaza and set it in the customer in Magento. This customer data includes:

- First name
- Last name
- Date Of Birth (DOB)
- Gender
- Email
- Printed Mailing List subscription (Spaaza attribute)
- Mailing List subscription (Spaaza attribute)
- Loyalty program opt in (Spaaza attribute)

Additionally, it pulls the address data and saves it in the *default billing address* of the customer. Because Spaaza only knows one name of a customer (obviously), the synchronisation also sets this name in the default billing address. If the customer has no default billing address and there is address data in Spaaza, it will be created in Magento.

The module will also receive new customer data from Spaaza through the Magento REST API once any data has changed.

Order synchronisation

Once an order is paid (`sales_order_invoice_pay` event), it is sent to Spaaza using the request queue. The Spaaza attribute *basket_id* will be set and saved after the order has been successfully sent to Spaaza.

- Any discounts will be subtracted from the product/item sale price before sending it to Spaaza.
- The SKU of the sold products get sent to Spaaza as identification.
- The shipping amount gets subtracted from the grand total that gets sent to Spaaza as `basket_total_value` .
- It however will be sent as a separate `shipping_charge` .

- The module uses the `user_id` of the customer assigned to the order. If there is none (e.g. for external or kiosk orders) you can set a `member_number` or `user_id` on the order model's Extension Attribute.

Events

- `spaaza_order_product_sale_price` provides a way to alter the basket items before they are added to the payload of the request. You can alter the price and quantity using a transport object.
- Using `spaaza_order_request_before` you can alter the complete payload before it gets saved to the request object.

Credit Memo Synchronisation

A credit memo will be sent to Spaaza after it has been refunded (`sales_order_creditmemo_refund` event) using a so called *inline return* API call. At this moment, there is little to hook on in this process as it's pretty straightforward: the returned items will be referenced by their product owner codes (SKU) and sent accompanied by their quantities.

There is however something to note here: Because the credit memo totals are collected and saved in the database before the credit memo has been sent to Spaaza, we need to determine the Spaaza voucher total ourselves. This is done using the amount that gets saved in the order extension attribute after the order has been sent to Spaaza. In theory, this could lead to small amount differences between Spaaza and Magento, but these have not yet been observed happening.

Spending Vouchers

Most of the Spaaza module does its work silently - you don't need to add any (frontend) code to make it work. Because the voucher spending process is very customer specific, we (Spaaza) chose to provide some generic functionality for adding basket vouchers to a cart (and order) but not to add it in detail.

Most functionality you need is provided by the VoucherManagement class (`\Spaaza\Loyalty\Model\VoucherManagement`). Whenever I use `$voucherManagement` , it is an instance of that class.

Getting the available vouchers for a customer

```
foreach ($voucherManagement->getAvailableVouchers($customer) as $voucher) {
    echo 'Available voucher: ' . $voucher->getVoucherKey() . "\n";
}
```

These vouchers are an instance of `\Spaaza\Loyalty\Api\Data\VoucherInterface` . Also unclaimed vouchers get returned. In Spaaza, a voucher first needs to be in "claimed" state before you can spend it (see below).

Spending vouchers on a cart

All claimed vouchers will automatically be applied to a cart. You can "claim" a voucher by calling:

```
$voucherManagement->claimVoucher($voucher, $customer, $cart);
```

If you supply the `$cart` (quote) object, it also saves the cart so totals are collected and the voucher amount gets deducted from the cart total. Only vouchers that can be spent (are claimed and not locked to another checkout or in-store transaction) will be applied.

Getting the vouchers that are marked to be used/redeemed in the cart:

```
$voucherKeys = $voucherManagement->getUsedVoucherKeysInQuote($cart);

// To get the voucher objects:
$vouchers = [];
foreach ($voucherKeys as $voucherKey) {
    $vouchers[] = $voucherManagement->getVoucher($voucherKey, $customer);
}
```

Please beware that the `getVoucher()` method always synchronously gets the latest data from the Spaaza API. That adds a little processing time to the request.

You should see the voucher discount total for the vouchers marked to be used on the cart page and in the checkout totals.

…And you can remove the voucher from the cart again:

```
$voucherManagement->unclaimVoucher($voucher, $customer, $cart);
```

Checking (honour) vouchers

Plugin points

We (Spaaza)have annotated some methods with `@api` in the code to indicate good places to put a plugin around, before or after.

`\Spaaza\Loyalty\Model\Total\Quote\Voucher::collect()`

At every quote totals collection, the available vouchers are pulled from Spaaza. Before the voucher totals are collected you could do something with the pulled vouchers, like automatically claiming them. It's no problem to call `\Spaaza\Loyalty\Model\VoucherManagement::getAvailableVouchers()` again for this, as the results of a call to Spaaza are stored in a registry.

`\Spaaza\Loyalty\Model\VoucherManagement::canSpend()`

This method determines whether a voucher can be spent on a quote. It's called while collecting the voucher totals. You could do some checks on a voucher before it can be applied to a quote, like checking it against an external service.

```
// An example of checking an honour voucher against an external service

use \Spaaza\Loyalty\Api\Data\VoucherInterface;
use \Magento\Quote\Api\Data\CartInterface;
use \Spaaza\Loyalty\Model\VoucherManagement;

class VoucherManagementPlugin
{
    public function afterCanSpend(
        VoucherManagement $subject,
        $result,
        VoucherInterface $voucher,
        CartInterface $quote
    ) {
        if ($voucher->getVoucherType() == VoucherInterface::VOUCHER_TYPE_HONOUR) {
            if (!$this->checkAgainstExternalServer($voucher->getVoucherHonourCode())) {
                return false;
            }
        }
        return $result;
    }
}
```

iFrame Module

The Spaaza Magento module also includes a suggested install of the Spaaza iFrame module. This is an iFrame which can be used to display Spaaza content on your site, for each individual authenticated customer, avoiding the need to for a Magento 2 admin/developer to write their own content.

For example, the iFrame can be used to display a customer's points, loyalty level, can be used to show a customer's vouchers, claim them ready to be redeemed in an order, and show the latest available promotions and campaigns you have configured in the Spaaza console site.

The iFrame can also be styled according to your own stylesheet using regular CSS.

When installing or updating the Spaaza Magento module using Composer you will be asked if you would like to install the iFrame module.

Demo Module

A demo module is available which uses the functionality in this module to expose UI elements, allowing points, wallet amounts and vouchers to be used:

Spaaza Magento 2 Demo

Contact and Questions

If you have questions about the Spaaza Magento 2 module or wish to find out more about our incentive marketing and loyalty platform and services, please get in touch via the chat button on our website.