OrderFlow Magento 2 Integration Guide

OrderFlow Ltd.

Document Version: 4.2.4

Document Built: 2024-02-16

Introduction

Background

OrderFlow is a Warehouse Management and Order Processing System that specialises in eCommerce fulfilment.

OrderFlow helps retailers and fulfilment specialist performs key 'back office' functions, from managing their warehouses, to handling picking and packing processes,

In order to fulfil these roles OrderFlow needs to integrate with eCommerce systems. Magento is the world's leading Open Source Enterprise eCommerce Platform. This document covers OrderFlow's integration with Magento.

Integration Operations

There are four main operations that are covered in the integration between OrderFlow and Magento.

- **export of products** from Magento to OrderFlow. Magento has a rich product model which makes it ideally suited to managing a retailer's product catalog. The product export operation allows products defined in Magento to be exported to OrderFlow.
- **export of orders** from Magento to OrderFlow. Magento is a public facing application, the 'shopping cart' software that members of the public use to find and purchase products online. Orders received in this way need to be passed through to OrderFlow for fulfilment.
- **notification of inventory changes** from OrderFlow to Magento. As the system responsible for stock management and fulfilment, OrderFlow needs to keep Magento up to date with changees to product inventory levels, to ensure that products are neither oversold nor undersold.
- **notification of shipments** despatched from OrderFlow to Magento. When shipments have been despatched, OrderFlow needs to inform Magento to ensure that the appropriate orders are marked as complete, and to assist the retailer's customer services in keeping customers up to date with the delivery of their orders.

These operations are all covered in detail in this document.

OrderFlow Magento Extension

OrderFlow is a web based warehouse management and order processing environment specifically designed to support high volume B2C pack and despatch operations.

Our integration with Magento is a very important part of our strategy to provide the best fulfilment environment for merchants using this eCommerce platform. On the OrderFlow side, we make use of the Magento web services, On the Magento side, RealtimeDespatch provide a free and open source extension which can be installed in the Magento environment.

The OrderFlow Magento extension aims to provide a robust integration between OrderFlow and Magento that addresses known issues with an OrderFlow/Magento integration that relies solely on natively available Magento Web Services.

The extension has been developed by the Magento agency SixBySix, based in Glasgow, and wholly funded by OrderFlow. It is hosted on the GitHub open source project repository.

The extension includes the following features:

- Event based despatch of configured data from Magento to the OrderFlow API (including products, quantities, prices, shipping values, order id, discount codes etc).
- Partial stock pushes from OrderFlow to Magento API.
- Scheduled and on-demand pushes of product definitions from Magento to OrderFlow.
- Requeue and retry of failed updates, with error notifications visible in the Magento administration console.
- Context sensitive links in Magento Admin that allow users with the necessary rights to navigate directly to the OrderFlow environment to view the details of shipment or products within the warehouse environment.
- Configurable schedules for product exports and order export retries in addition to general configuration settings.
- Extensible, open codebase to allow further customisation for merchant specific requirements.

This document aims to cover everything that merchants and their agencies need to know about installing, using, configuring and receiving support for the OrderFlow Magento extension, both from the OrderFlow and the Magento administrator's perspective.

Licensing and Compatibility

The OrderFlow Magento extension is released under the Open Software Licence v3.0. The Licensor is OrderFlow Limited.

The module is compatible with the following Magento versions:

- Magento CE 2.x
- Magento EE 2.x

Support Arrangements

The OrderFlow Magento extension is designed to be easy to install and configure.

However, Magento is a complex environment, and each installation typically includes a number of other third party modules that need to work alongside the OrderFlow extension and may interact with it. Merchants manage their Magento installations either through the use of an in house support and development team, or by contracting a Magento Agency.

In these kinds of environments, additional support effort is often required to install and configure the module to work successfully.

This chapter explains the support arrangements available for the OrderFlow Magento module.

Key Parties

For the avoidance of confusion, it is worth pointing out who may be involved, and what their responsibilities are likely to be in the process of installing, configuring and supporting the OrderFlow extension.

The Merchant

The Merchant is the 'owner' of the overall process. Typically, the Merchant is also a OrderFlow customer, or a client of a 3PL who is an OrderFlow customer.

Extension Developer

The Extension Developer is SixBySix, also a Magento Developer. They do not have any direct responsiblity for administering the Merchant's environment.

Their efforts may be required to support specific issues relating to the functioning of the module, to develop additional module features, etc.

Any chargeable work incurred by the Extension Developer is covered by OrderFlow. In some cases, depending on the issue concerned, the costs incurred are passed on to the OrderFlow customer (normally, the Merchant).

If any conflict occurs between the OrderFlow extension and any other third party extension, the Extension Developer may be brought in on a chargeable basis to address this.

Note that if the Extension Developer is required to provide rapid response support for the module to coincide with a new launch, then advance notice will need to be given for this in order to ensure their availability.

The Extension Developer should typically not have direct access to the live Magento server. They will however require direct access to the test Magento server (including SSH access) to perform support tasks.

Primary Magento Agency

The Primary Magento Agency is responsible maintaining the Merchant's Magento environment, **administering their live and test servers**, etc, and for developing custom Magento features for the Merchant.

The Primary Magento Agency will be responsible for all **deployments**, including those to both test and live environments.

The Primary Magento Agency will be responsible for **maintaining a test environment** that closely represents the features present in the live environment. If necessary, the Primary Magento Agency may be asked to replicate the aspects of the live environment in test, in order to reproduce live issues and to allow the Extension Developer to access to a technical environment in which these issues can be debugged.

Support Process

All support for the module is coordinated through the OrderFlow ticketing system. For each OrderFlow customer, a Magento module-specific project will be set up, to which each of the above-named parties will have access.

OrderFlow plays a coordination role in this process, providing the following services:

- leading the initial induction process to ensure that the module is installed correctly
- ensuring that the relevant parties have access to the shared Merchant-specific project on the OrderFlow support ticketing system
- ensuring that the necessary system access is available to the relevant parties
- if an issue has been raised, doing the the initial triage to verify that the issue is valid, and to identify a possible cause
- if *additional* chargeable work is likely to be incurred, or if further investigation is required by the Extension Developer, to arrange for this to happen

OrderFlow will charge for support work undertaken on the same basis as support work for OrderFlow itself. Typically, this means that work is chargeable unless it is the result of a bug which has *not* been introduced by a specific change made to the Magento environment by the Merchant or the Primary Magento Agency.

Note that neither OrderFlow nor the Extension Developer (SixBySix) will take any responsibility for configuring or maintaining any Magento environment, or for deploying changes to this environment, apart from configuration changes made through the Magento Administration Console.

Required Access

The table below indicates the level of access that will be required to support the Magento module.

Access Required	Primary Agency	OrderFlow	Extension Developer
Test Admin Panel	Yes	Yes	Yes
Test SSH	Yes	Yes	Yes
Test Database	Yes	Yes	Yes
Live Admin Panel	Yes	No	No
Live SSH	Yes	No	No
Live Database	Yes	No	No

Note that where access to the live system is required for either OrderFlow or the Extension Developer, this will need to take place via a TeamViewer session with the help of the Primary Agency, arranged between the parties concerned.

Induction Process

Merchants using the OrderFlow Magento extension will be required to go through an induction process.

The purpose of an induction process is to ensure that problems that might arise with the extension can be dealt with in a cost effective and timely manner.

Having an induction process helps to ensure that the Merchant's and Primary Magento Agency's first experience of using the module is a positive one.

The induction process will involve the following:

- clarification on who the main parties are as described in the previous section.
- ensuring that system access as described above is already available.
- clarification the support process, and ensure that all parties are aware of their responsibilities.
- help in verifying the installation of the module in the test environment, with the assistance of this documents.
- any further training in the use of the Magento extension.

A recommended first step for any Magento agency that is new to the module is to install the module on a clean Magento environment before attempting to install the module in the customer's test environment.

Customisation

The Primary Magento Agency should be able to take care of any **additional advanced configuration** that is required to customise data mappings for products and orders exported from Magento to OrderFlow. Additional support of this effort can be provided, if required, by the Extension Developer.

Installation

Prerequisites

These instructions assume that an administrator has command line access to the relevant Magento installation.

First Steps

Change into the root directory of your Magento installation

```
cd <your Magento install dir>
```

Retrieve The Module

We now need to retrieve the module's codebase by executing the following command:

```
composer require realtimedespatch/magento2-orderflow
```

This command looks up the latest module version via Packagist, and resolves, and installs it's dependencies.

Note that you will need access keys to download from the Magento server, follow http://devdocs.magento.com/guides/v2.0/install-gde/prereq/connect-auth.html on how to do this

Your public key is your username, private key as password.

Enable The Module

Next up, Magento requires us to register, and enable the module by executing the following command (*):

```
php bin/magento module:enable RealtimeDespatch_OrderFlow
```

Upgrade the Database

Followed by installation of the module's bespoke database schema. This is performed by executing the following command (*):

```
php bin/magento setup:upgrade
```

Recompile

To complete the process, we will be prompted to recompile Magento 2 (*):

```
php bin/magento setup:di:compile
```

(*) All of these command are executed via the Magento 2 Command Line Tool

Further References

- Magento 2 Command Line Tool
- Enable or disable modules

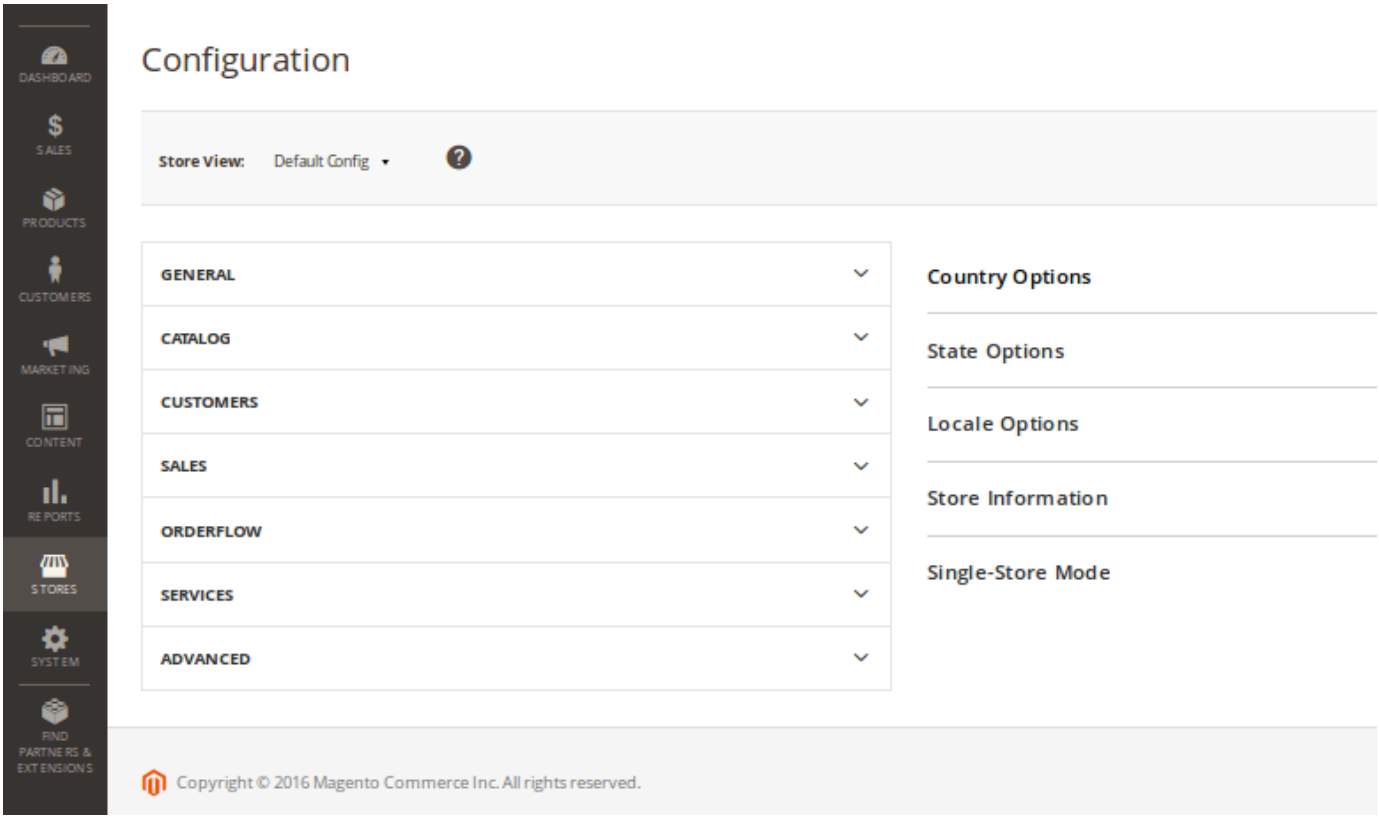
Magento Configuration

Assuming that the OrderFlow Magento extension has been installed successfully, we now need to configure it to work with OrderFlow.

Note that there is also some configuration required on the OrderFlow side, which we cover in the OrderFlow Configuration chapter.

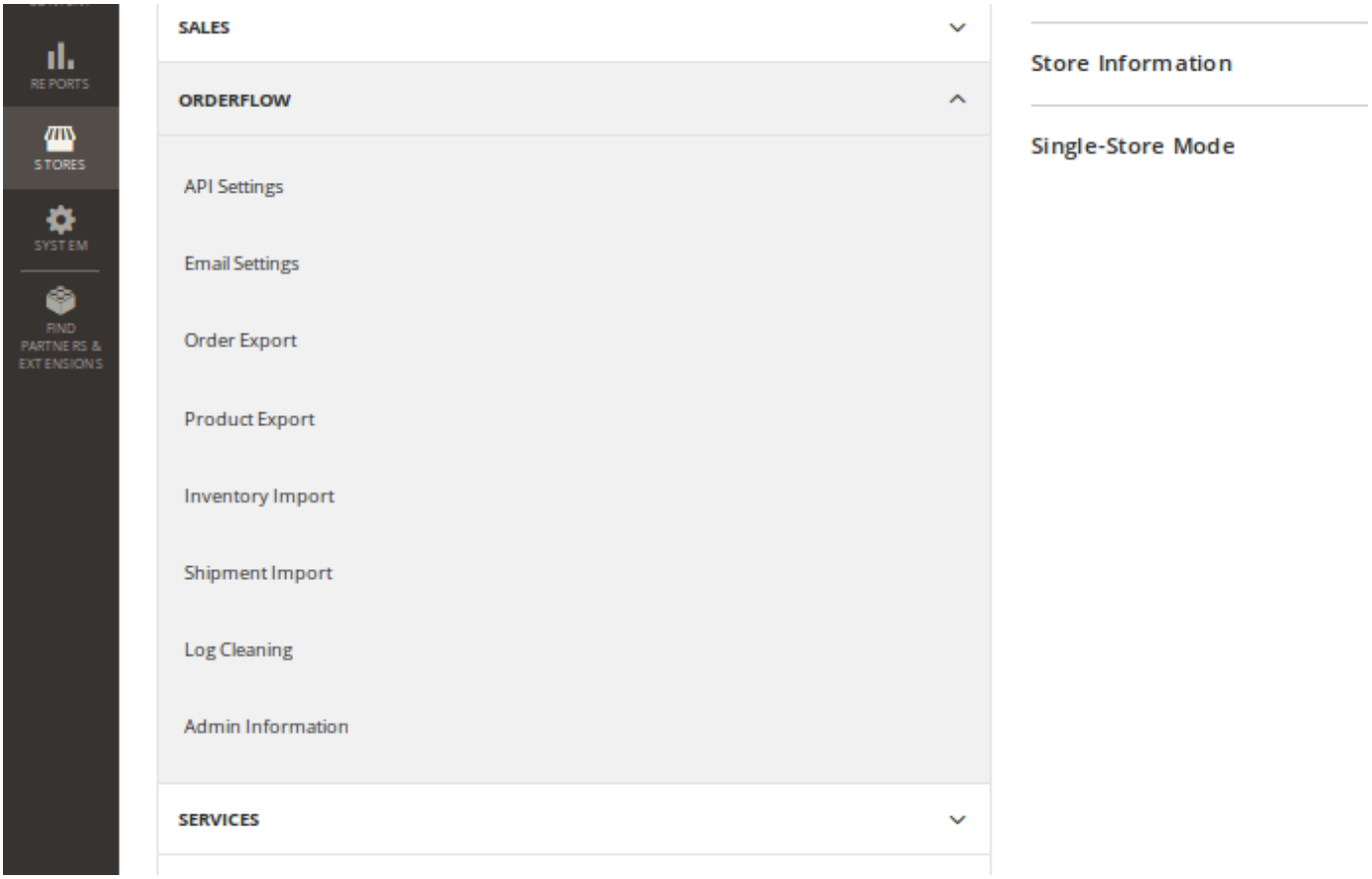
Navigate to **Stores -> Configuration** on the Magento Admin Panel.

You should also be able to see OrderFlow in the Configuration menu, as shown in the screenshot below.



OrderFlow Configuration

Click on **OrderFlow** and the module configuration sub menu will appear.



API Settings

These are the main settings for connecting to the OrderFlow instance. The correct values for these will be depend on the configuration of the target OrderFlow environment, and will be set with the help of OrderFlow support.

Endpoint: This is the base OrderFlow API URL to which the extension will send messages. Note that this needs to include the server and 'context path' portion of the URL.

An example base URL would be *https://demo.orderflow-wms.co.uk/web/*.

Username: The OrderFlow user configured to receive API requests from the Magento extension.

Password: The password for the above user.

Organisation: The configured OrderFlow organisation to which API requests relate. Must correspond with the identifier for an organisation within OrderFlow to which the above named user has access.

Channel: The configured OrderFlow sales channel to which API requests relate. Note that in OrderFlow, channels belong to organisations, so this channel needs to be one which belongs to the organisation named above. Again, the value supplied must correspond with the identifier for a channel within OrderFlow to which the above named user has access.

The channel may already be set up in OrderFlow, otherwise it can be entered here and subsequently created in OrderFlow. By convention this channel is usually named *magento*.

Email Settings

Admin Name: The name of the administrator responsible for the installation.

Admin Email: The address for email alerts.

Order Export

The settings for the export of orders from Magento to OrderFlow are supplied here. Once orders have been invoiced in Magento, they are eligible for export to OrderFlow. Orders can be exported manually, but more typically orders will be exported automatically based on the schedule configuration in this section.

Enabled: Needs to be set for export of orders to take place.

Cron Expression: Controls the frequency of the automatic order export. An order will be exported if it is eligible for exporting, and has not already been exported. It is not necessary to set this too frequently - once every few minutes should be more than adequate.

Batch Size: Determines the maximum number of orders that will be exported each time the schedule runs.

Product Export

This controls behaviour of the export of product definitions from Magento to OrderFlow.

Enabled: Needs to be set for export of product definitions to take place. If enabled, then any change to a product definition will result in an update being sent to OrderFlow. Additionally, if enabled, an option will be present on the **Catalog -> Manage Products** grid to manually export to OrderFlow the definitions for selected products.

Include Image Field: Magento can include a product image (or a default image) when exporting a product or products to OrderFlow. This field either enables or disables including an image. Note that if this option is set, then if there is no product image AND no default Magento image specified, the export will fail. If the value is set to *Yes*, then confirm there is a default Magento image available in **System > Configuration > Catalog > Product Image Placeholders**, where you can upload a default image.

Cron Expression: controls the frequency of checks for updated products. Each time the cron expression is run, then up to a configured maximum number of new or updated product definitions are sent to OrderFlow. The configured maximum is controlled by the **Batch Size**.

Inventory Import

The inventory import is initiated by OrderFlow to send product inventory quantities to Magento.

For performance reasons, stock updates are not applied immediately. Instead, the requests are queued by the Magento extension, and applied via a Magento cron job.

The settings on this screen control the timing of these schedules, as well as the number of stock updates applied per schedule execution.

Enabled: Needs to be set to allow the import of inventory updates to take place.

Cron Expression: Determines the run frequency of the job to apply stock updates following requests received from OrderFlow.

Batch Size: Determines the maximum number of stock update requests that can be processed each time the inventory import job runs.

Allow Negative Quantities: Determines whether negative inventory quantities can be applied. An inventory level will be negative on OrderFlow for example when orders have been received for a product but there is no stock present for that product. Magento won't allow negative numbers to be applied, so if set to true, the extension will simply convert negative values received from OrderFlow to zero. The value for this field should be set to 'Yes'.

Shipment Import

The shipment import is initiated by OrderFlow when an order is despatched or part despatched, resulting in the creation of a shipment against the order in Magento.

As with the inventory imports, the shipment creation does not take place immediately; the request for shipment creation are queued, and applied when the shipment import schedule job executes.

The settings on this screen controls the execution of this scheduled job.

Enabled: Needs to be 'Yes' for the shipment import to take place.

Cron Expression: Determines the run frequency of the job to create shipments in Magento following the despatch of shipments in OrderFlow.

Batch Size: Determines the maximum number of shipments created in Magento each time the scheduled job runs.

Log Cleaning

The extension produces a significant amount of log and management data that is used both for the functioning of the module as well as being helpful for verification. The log cleaning options ensure that the volume log data that the extension generates is kept manageable, and does not grow indefinitely.

Cron Expression: The cron expression that controls the frequency of log cleaning operations. Daily running of the log cleaning operations is sufficient.

Web Log Duration: The period for which records generated by the extension will be retained in the Magento *var/log* directory.

XML Log Duration: The length of time for which request and response XML will be stored.

Admin Information

This option should be left enabled.

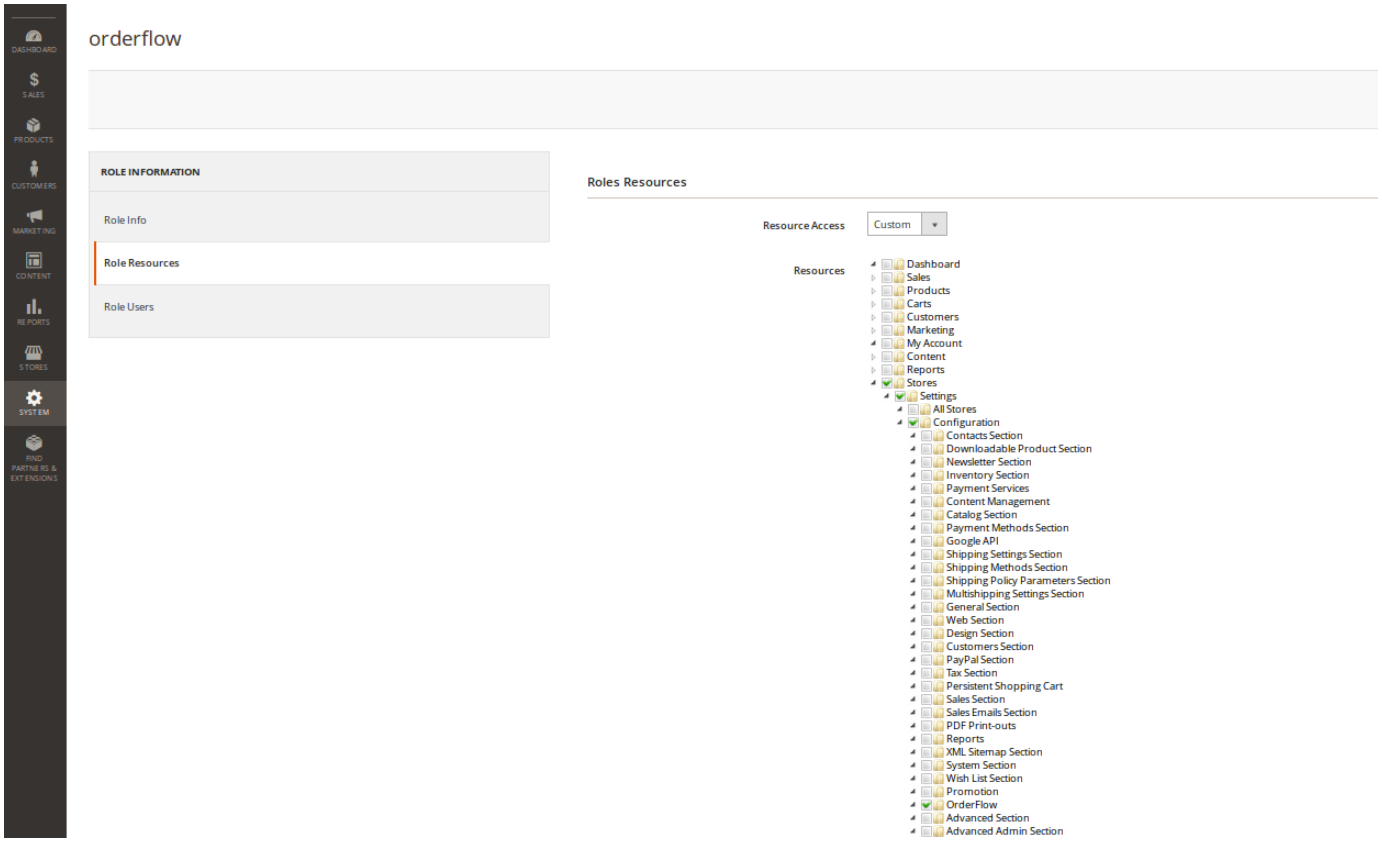
Web Services Configuration

OrderFlow uses the SOAP web services to communicate with the Magento extension. Configuration is required on the Magento side to enable the relevant web services.

A valid enabled Web Services User associated with an applicable Role must be in place for the extension to work. To create a Web Services User, first create an applicable role via the **System > User Roles** menu.

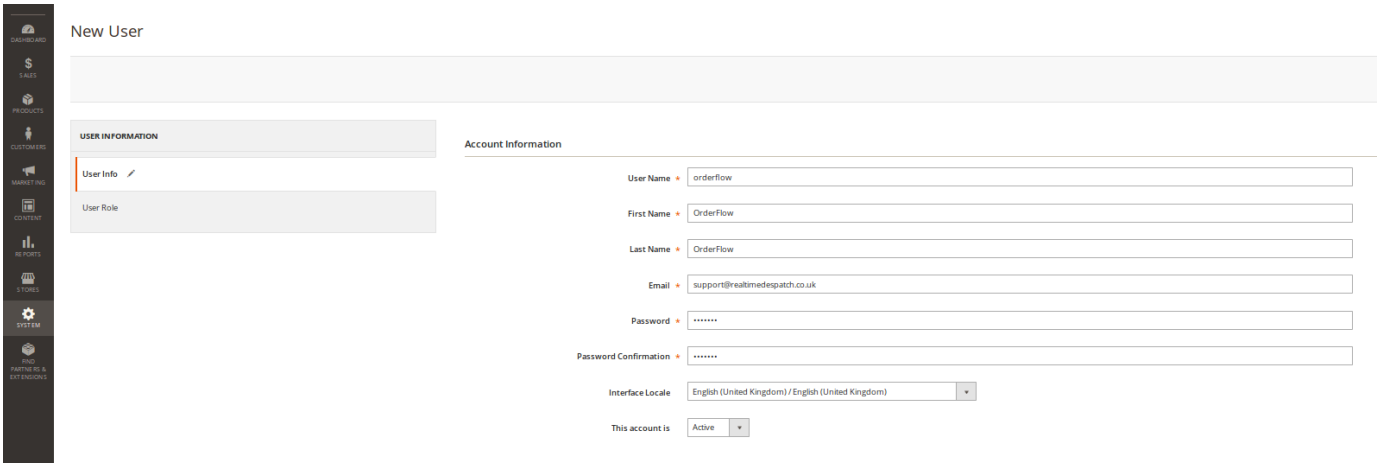
By convention, we typically use the name *orderflow* for the OrderFlow Magento extension Role created in this way.

From the **Role Resources** submenu, the minimal custom resource access required is for the 'OrderFlow Import' resource, as shown below:



You can then create a *User* via **System > All Users > Add New User**.

Enter a value for the required fields, and ensure that the account is active.



By convention, we typically use the name *orderflow* for the OrderFlow Magento extension user created in this way.

Make a note of the *Password* for the user, as this will be required when setting the relevant password when configuring OrderFlow.

You will then need to associate the user with the role created above in order to give the user access to the extension resource.

OrderFlow Configuration

There instructions are primarily targeted at the OrderFlow administrator responsible for carrying out the OrderFlow side of the setup for the OrderFlow and Magento integration. For these steps you will need admin privileges on the target OrderFlow instance.

Module Configuration

As OrderFlow is a modular system, you will first need to ensure that the necessary modules are present:

Navigate to the **Advanced -> System -> Loaded Modules**, and confirm that the following modules are loaded.

```
rtd2-process
rtd2-integration
rtd2-integration-magento2
rtd2-web-integration
```

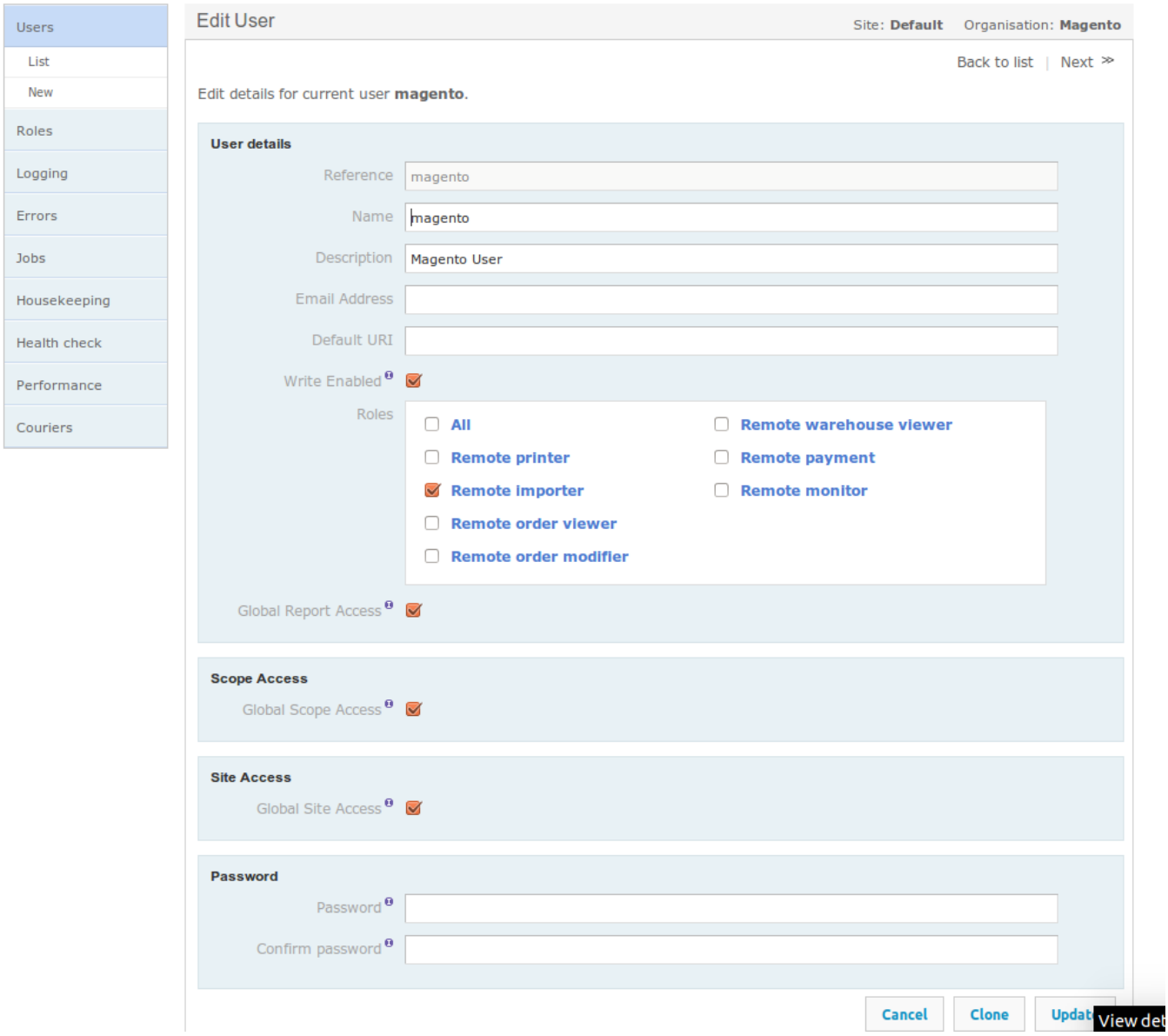
If any of these modules are not present, then this will need to be rectified in order to complete the integration. The steps for doing this are outside of the scope of this document.

User Configuration

In order to allow Magento to connect with OrderFlow, make sure that there is a user present that corresponds with the user setup in API Settings configuration.

Note that the user will need:

- access to the */remoteorder/imports/importitems.xml* and */remoteorder/imports/importitems.xml* resources. This is typically achieved by associating the user with the *Remote Importer* role.
- access to the organisation and channel required on which the connection is to take place.



Application Properties

The next step in to set up the application properties to correspond with the settings of the Magento installation described in Magento Configuration. These include values for the endpoint URL, API user and API Key as described in the Magento Web Services Configuration section.

General Properties

From **Setup -> Properties -> Search** in OrderFlow, you will need to check the value for the property **inventory.process.based.notification** is set to `true`. (This property was added for backward compatibility purposes, but with the value set to *false*, inventory import using the extension will not be possible.)

Magento-specific Properties

From the same menu, do a search, but filter on the group 'Magento API'. This is where the Magento API properties are configured. We are only concerned with the following SOAP properties.

Magento 2 API SOAP base URL: this is the base URL for the Magento API endpoint. In our example, the value used is *http://orderflow.magento/soap/default*.

Magento 2 API SOAP User: this corresponds with the name of the user set up in the Magento Web Services Configuration.

Magento 2 API SOAP Password: this corresponds with the password for the user set up in the Magento Web Services Configuration.

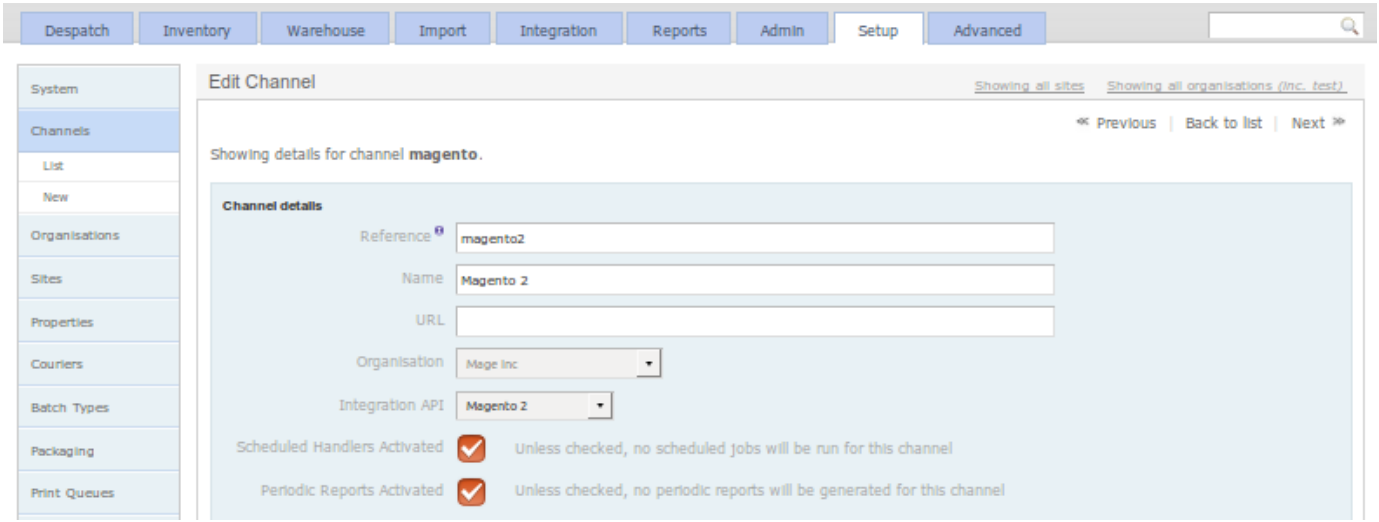
Note that the properties used may need to be scoped by organisation or channel to align with the channel and organisation used with the specific Magento instance. For single channel or organisation OrderFlow environments, this will not be necessary.

Channel and Organisation Configuration

OrderFlow is a multi-channel and multi-organisation system that allows a single instance of OrderFlow to interface with multiple instances of Magento and other eCommerce platforms.

For a particular integration, the OrderFlow Channel and Organisation through which the specific Magento instance will communicate to OrderFlow will need to be correctly configured.

Give the channel the same name as entered in the Magento configuration e.g. magento. Set the organisation that will own the channel, choose Magento for the Integration API. The *Scheduled Handlers Activated* and *Periodic Reports Activated* options should be enabled.



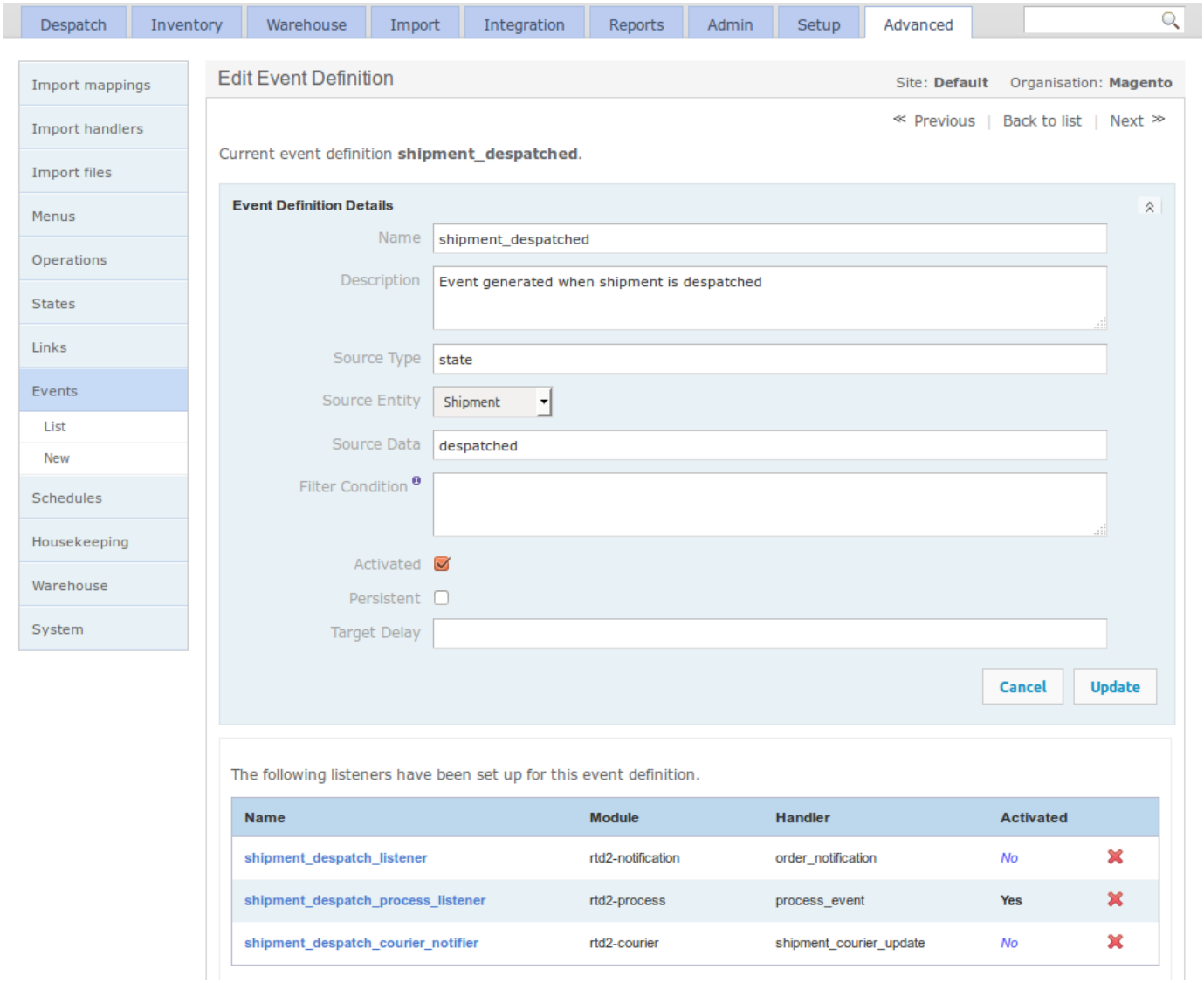
For inventory updates to be posted back via using the Magento channel, the organisation's *Primary Channel* should point to the Magento channel.

Event Configuration

To enable shipment despatch notifications from OrderFlow to Magento, use the **Advanced > Event** menu to navigate to the event definitions screen.

Then select the *shipment_despatch* event. From there, enable the *shipment_despatch_process_listener* if it is not already enabled, and disable the *shipment_despatch_listener* if it is enabled, as shown in the screenshot below.

(The latter event is present for backward compatibility purposes only, and is now deprecated.)



Note: that you might want to turn off any event listener that uses the **rtd2-notification** module with the **order_notification** handler, as these will generate errors in the logs.

Remember to refresh the event definitions using the *Refresh* button at the on the event definitions screen.

Import Mapping

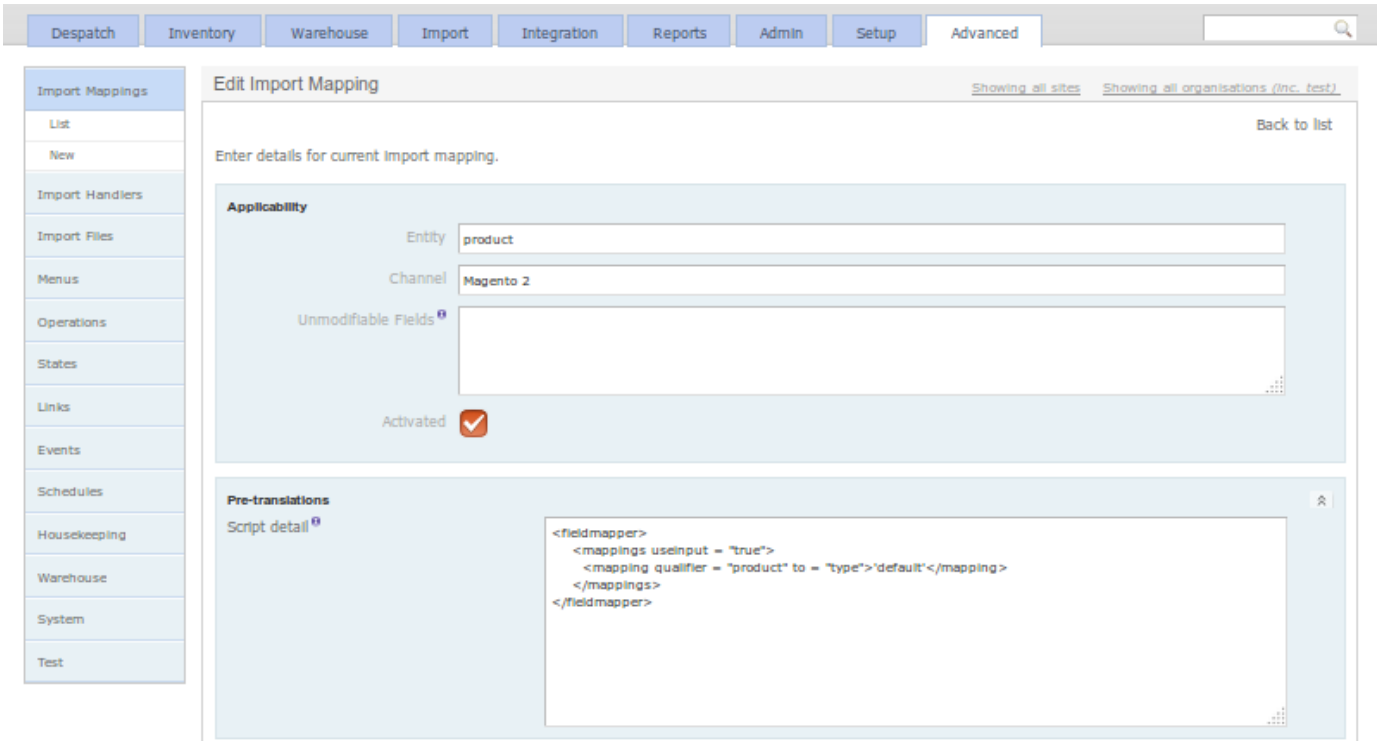
As Magento does not send a product type when exporting product definitions, you will need to configure the import mapping to handle this. Navigate to *Advanced > Import Mappings > List* and select the product entry that is associated with the Magento channel. Add the following to the *Pre-translations* script field, then use the 'Update' button to apply this configuration:

```
<mapping qualifier = "product" to = "type">'default'</mapping>
```

If there is no import mapping for the Magento-enabled channel, then you will need to add one. Click on the New new button, set *Type* to 'Product', set the *Organisation* or *Channel*, and enable the *Activated* field. Then add the following text into *Pre-translations*, and update the form:

```
<fieldmapper>
    <mappings useinput = "true">
    <mapping qualifier = "product" to = "type">'default'</mapping>
    </mappings>
</fieldmapper>
```

An example of this in action is shown below.



Product Export

With the module successfully installed, this section describes how to use the OrderFlow Magento extension.

It describes how the extension functionality can be applied, both manually and automatically. It also describes how the automatic options can be monitored.

The explanations below assume that the module has been enabled, that job scheduling functionality described in Magento Configuration has been turned on.

Export Types

The module supports three flavours of product export.

implicit: When a change is made to a product it is queued to be exported to OrderFlow

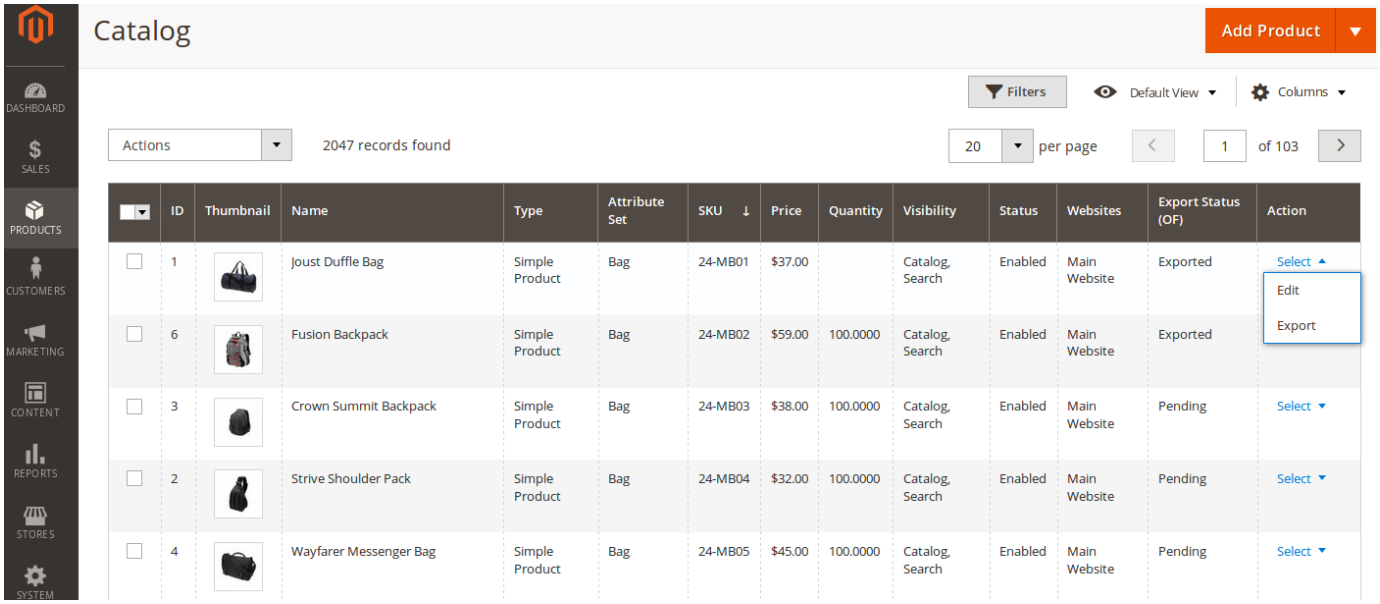
manual: the product export can be triggered manually from the catalog product grid.

bulk: All products marked as 'pending' under the Export Status will be picked up by the cron to export

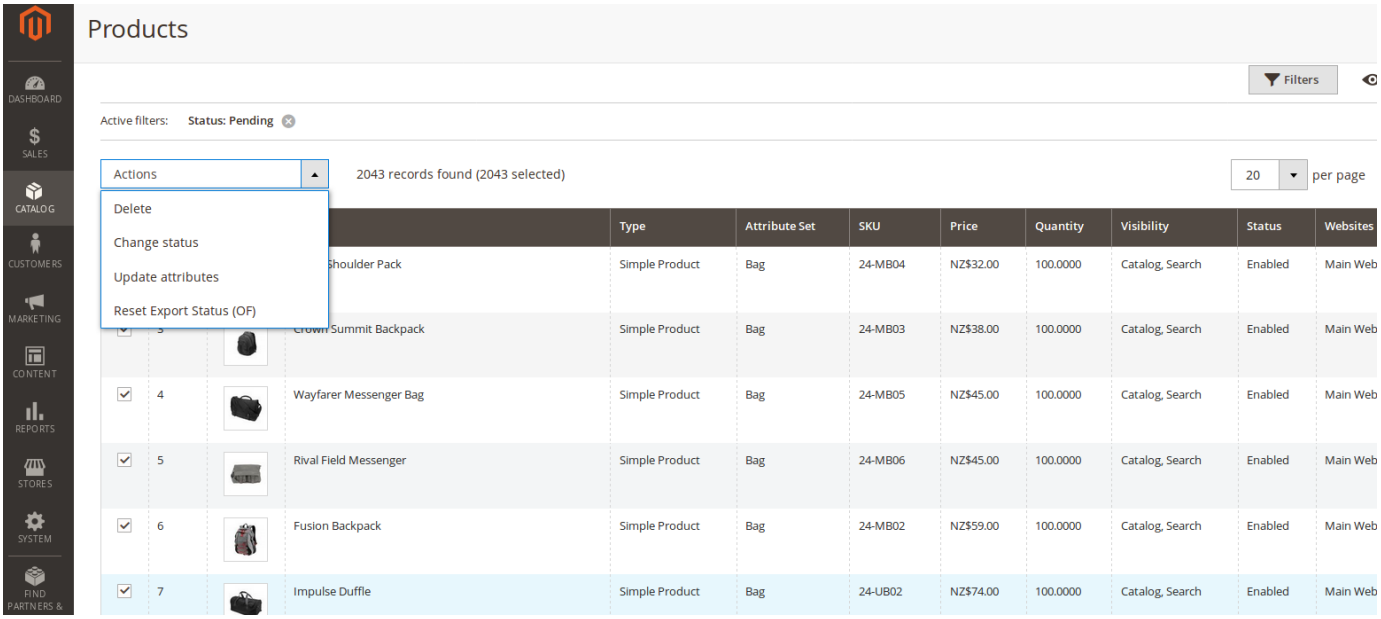
We'll go through each of these actions in turn.

Implicit product export is automatically triggered when a products properties are changed. This switches the products rxport status back to 'Pending'

Manual product export can be triggered from the Magento Admin Panel **Products -> Catalog** grid, using the 'Export' under the action column, as shown below:



Bulk product exports are triggered by the cron running, this will pick up all products that are marked as pending and notify OrderFlow. To reset all products to the pending state select all products and then from the drop down menu select **Reset Export Status (OF)**.



Magento Workflow

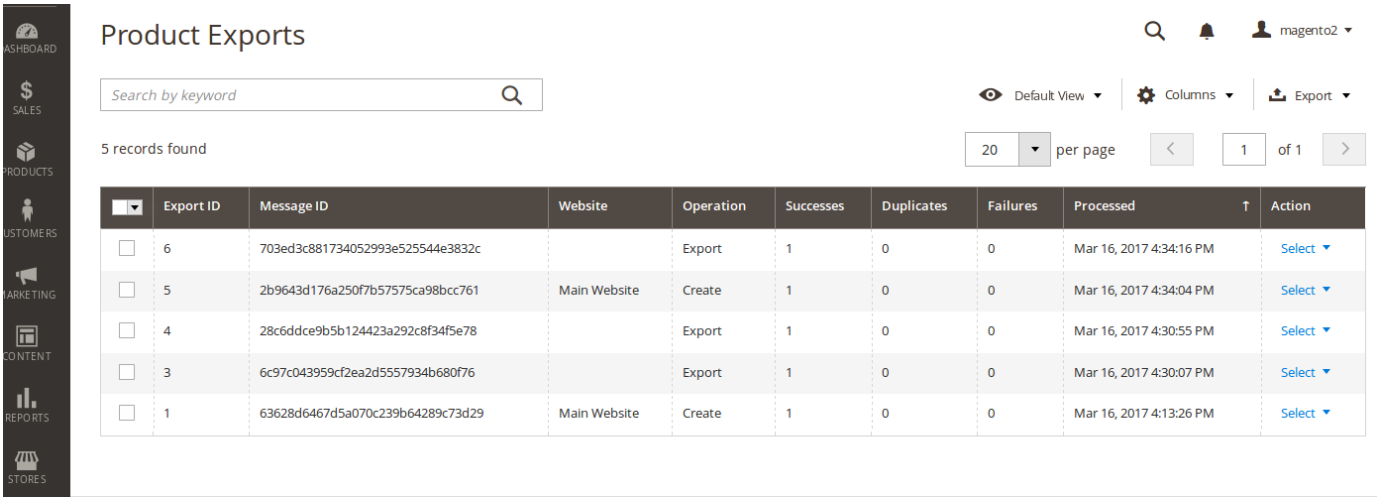
On Magento 2, the process of exporting a product to OrderFlow uses the following state flow:

- An product is created, or modified.
- The product is placed into the 'Pending' status.
- Magento contacts OrderFlow at regular intervals to let it know that there is a new or updated product available to be retrieved.
- The product is placed into the 'Queued' status.
- OrderFlow makes a request to Magento via the native APIs to retrieve the product.
- The entity is placed in the 'Exported' status.

A similar workflow applies for exporting orders to OrderFlow.

Product Export History - Magento

The history of product exports is available from the **System -> OrderFlow Exports -> Product** menu.

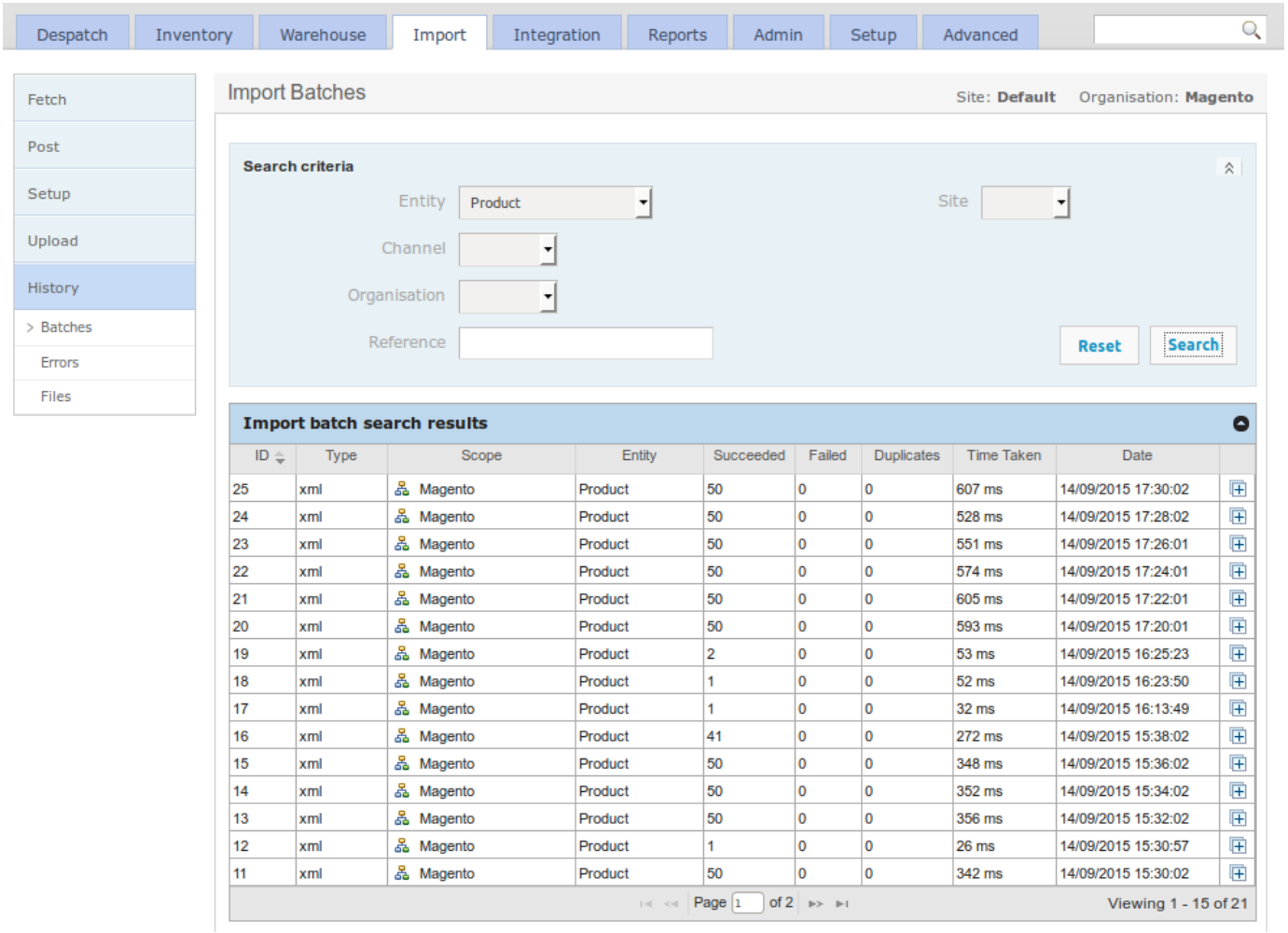


The **summary** product export admin grid includes an Export ID, Message ID, the returned responses (Successes, Duplicates, Failures) and the ability to view requests in detail.

The **detail** product export form repeats the summary information, as well as letting you view the Export Lines details. Depending on log cleaning settings within the configuration, details on the raw requests and response XML are available from this screen.

Product Import History - OrderFlow

The history of product imports can also be seen in OrderFlow in the **Import -> History -> Batches** menu, filtered by the *Entity* 'Product', as shown below:



Testing the Product Export

This section describes a simple end to end step for verifying the connectivity of the module as well as the product export process. It assumes that there are simple product definitions present in the Magento catalog.

From the Magento Admin Panel **Products > Catalog** menu, select a simple product by clicking its 'select' button under the Action column. Select 'Export' menu item.

If successful, you will see this message:

```
Product xxxxx has been queued for export to OrderFlow.
```

Next, navigate within OrderFlow to the **Import > History > Batches** menu. The first entry in the list will be the most recent import. Click on it and you should see the **Import batch details** and below, the message received from Magento. An example message text receipt may look as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<imports>
    <import type="product" operation="merge" externalReference="wbk002c-Black-S"
        <![CDATA[externalReference=wbk002c-Black-S
        description=Black Nolita Cami-Black-S
        weight=10
        priceNet=25.89
        sellable=1
        priceGross=25.89
        magentoType=simple]]>
    </import>
</imports>
```

The OrderFlow Magento extension sends the message in the native OrderFlow API format (as described in https://www.orderflow-wms.co.uk/resources/documentation).

If the product existed in OrderFlow before the message arrived, the product information will have been merged into the existing record. If it did not exist, the new product definition will have been created.

The next step is to find the product in using the OrderFlow product search.

Take a note of the product externalReference, then go to **Inventory > Products > Search**, and enter the externalReference in the *Product Code* field. Set the *Active* dropdown to blank and click on the 'Search' button. You should see the product that has just been imported in the search results list. Click on it to see more information. Selecting the magnifier next to *Created* will take you to the import record last used to update the import definition.

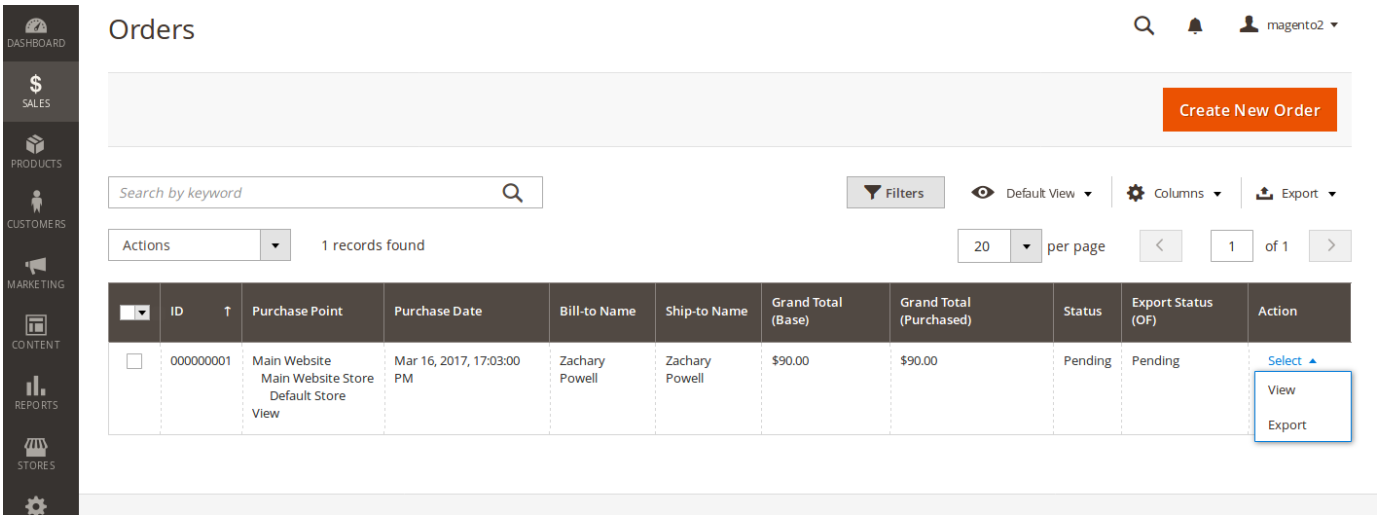
Note: a product imported into OrderFlow will not necessarily be set to activated by default, it will be inactive until all the conditions defined using the system property **product.activation.required.fields** are met.

Order Export

The OrderFlow Magento extension supports the export of orders, allowing them to be packed and despatched in the OrderFlow environment.

As with product export, orders are typically configured to be exported to OrderFlow automatically once they become eligible for export. However, orders can also be exported manually, which is particularly useful while still testing the integration.

An example of this in operation is shown below.



To manually export orders, select orders which are in the state 'Pending'.

Select the 'Export' *Action*. You should see a message such as the following:

```
Exported 2 orders(s) to OrderFlow with 0 failures.
```

Note that only open invoiced orders can be exported in this way.

Orders which are 'Pending' or 'Cancelled' will not be eligible for export, so attempting to export these will return a message such as:

```
No exportable (invoiced) orders selected.
```

Magento Workflow

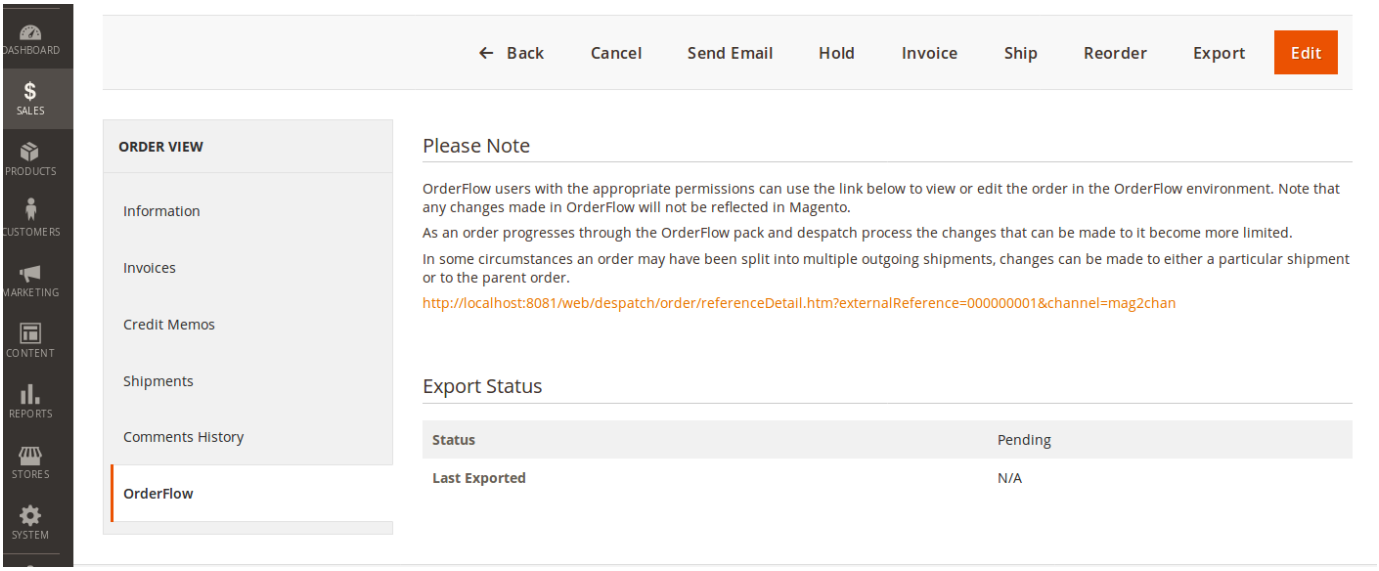
On Magento 2, the process of exporting a order to OrderFlow uses the following state flow:

- An order is created, or modified.
- The order is placed into the 'Pending' status.
- Magento contacts OrderFlow at regular intervals to let it know that there is a new or updated order available to be retrieved.
- The order is placed into the 'Queued' status.
- OrderFlow makes a request to Magento via the native APIs to retrieve the order.
- The entity is placed in the 'Exported' status.

A similar workflow applies for exporting products to OrderFlow.

Order Embellishments

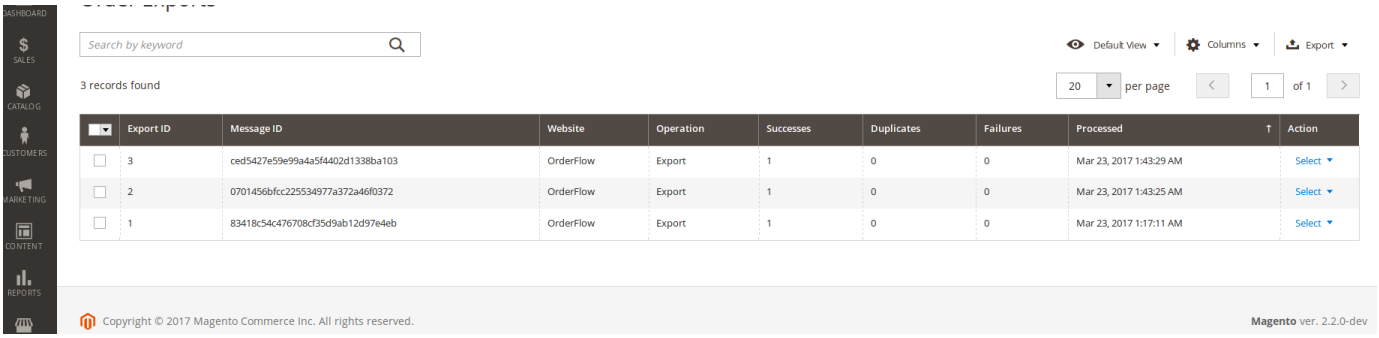
The OrderFlow Magento extension includes an additional section in the order summary, as shown below



The section includes information on how OrderFlow may be used, and provides a link to the current order in OrderFlow (assuming the order has been exported).

Order Export History - Magento

The history of order exports, including those exported manually and automatically is available from the **System -> OrderFlow Exports -> Orders** menu.

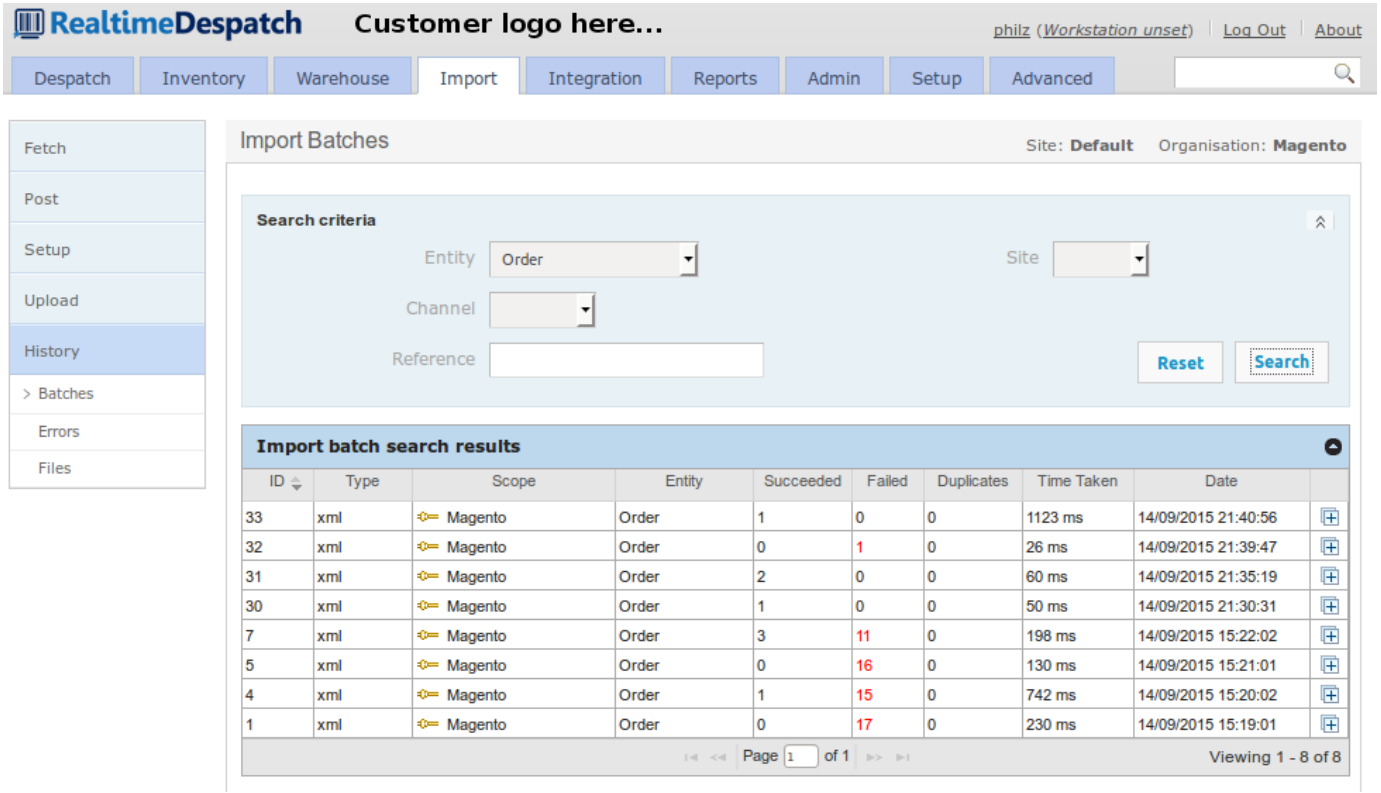


As with the product export screen, the **summary** order export admin grid includes Export ID, Message ID, the returned responses (Successes, Duplicates, Failures) and the ability to view requests in detail.

The **detail** order export form repeats the summary information, lists the exported order ID's along with response, any message and a timestamp. Depending on log cleaning settings within the configuration the raw requests and response XML may be detailed.

Order Import History - OrderFlow

The equivalent history of order imports into OrderFlow can also be seen in OrderFlow in the **Import -> History -> Batches** menu, filtered by the *Entity* 'Order', as shown below:



The most recently imported order will also normally appear as the first item in the Order search, reached from the **Despatch -> Orders -> Search** menu.

Inventory Import

The inventory import operation is a very important part of the OrderFlow Magento integration, particularly as the imported stock quantity is generally used to determine whether or not the product concerned appears as 'in stock' and therefore sellable to the public.

The inventory import uses a queueing architecture through which many (any number from one to several hundred, for example) product stock levels may be sent from OrderFlow to Magento via a single message. This message is queued for processing.

The interface is robust in allowing for these messages to be delivered out of sequence, while still being able to determine whether a particular update is the most recent available, or whether it should be discarded as a stale update.

As mentioned earlier in this document, the stock updates are initiated by OrderFlow.

The product inventory quantity is calculated as the following:

- the total stock across all locations
- *less* stock in non-usable locations (damaged, quarantined, etc.)
- *less* the stock quantity associated with open (unfulfilled) orders received by OrderFlow

Initiating Inventory Import

There are three ways that inventory import into Magento can be triggered. In each case, the action is initiated in OrderFlow.

Single product push: the stock level for an individual product can be 'pushed' or exported from OrderFlow to Magento. This option is not typically used in day to day operations, but is very useful in testing.

Full stock push: the stock levels for all products in OrderFlow can be pushed from OrderFlow to Magento. This option is typically used to reset all of the Magento stock levels, for example, after a stock take.

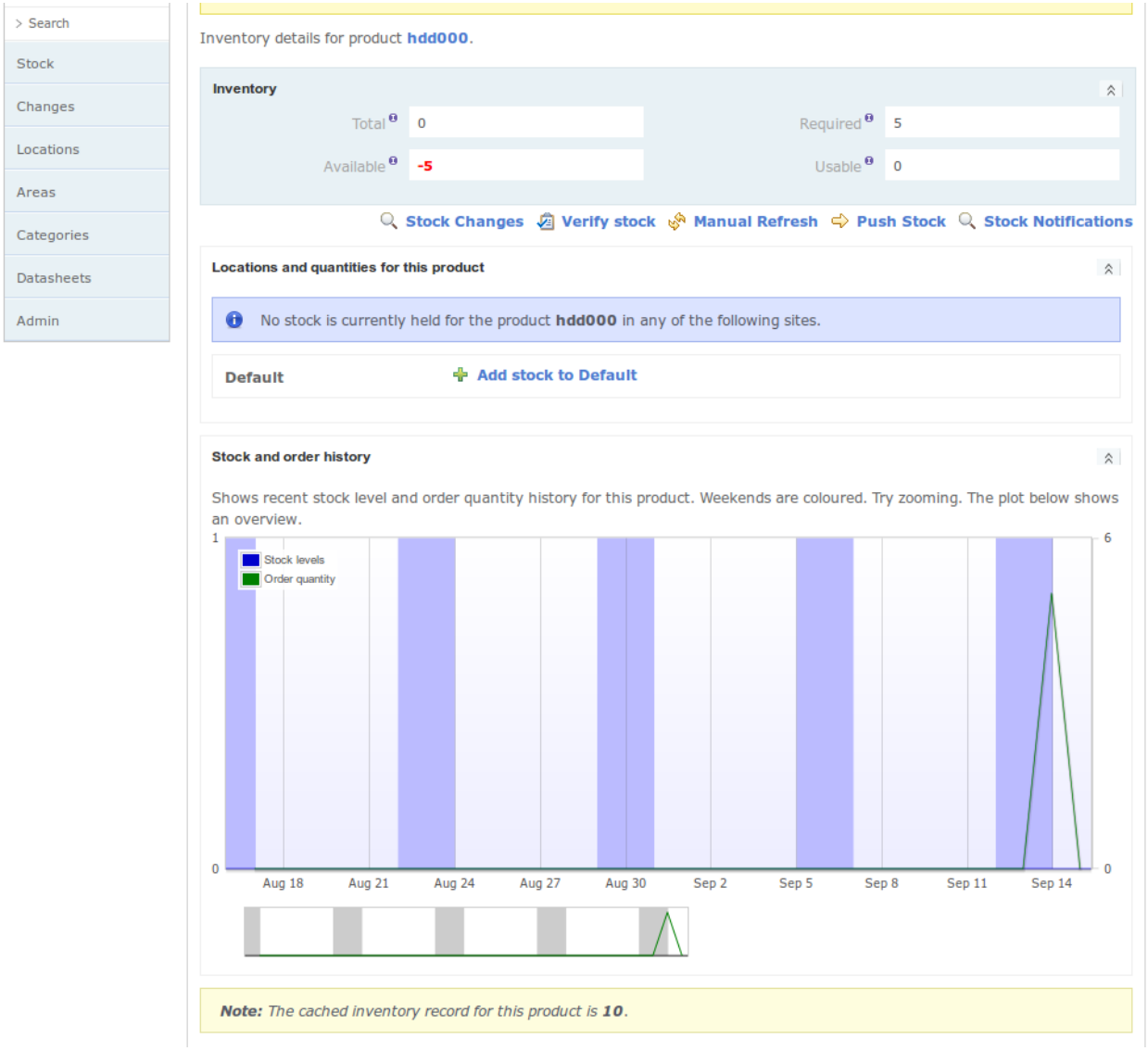
Incremental stock push: this mechanism involves a stock push for products whose stock level may have changed since the last export took place, so is very efficient in its use of system resources. In almost all environments this process will be automated.

Single Stock Push

The best way to understand how OrderFlow and the Magento extension work together to do stock updates is to work through a single stock push example.

Start by navigating to a test product, which we can do through the **Inventory -> Inventory -> Search** menu.

The following screenshot shows the listing for one of the out of stock products.

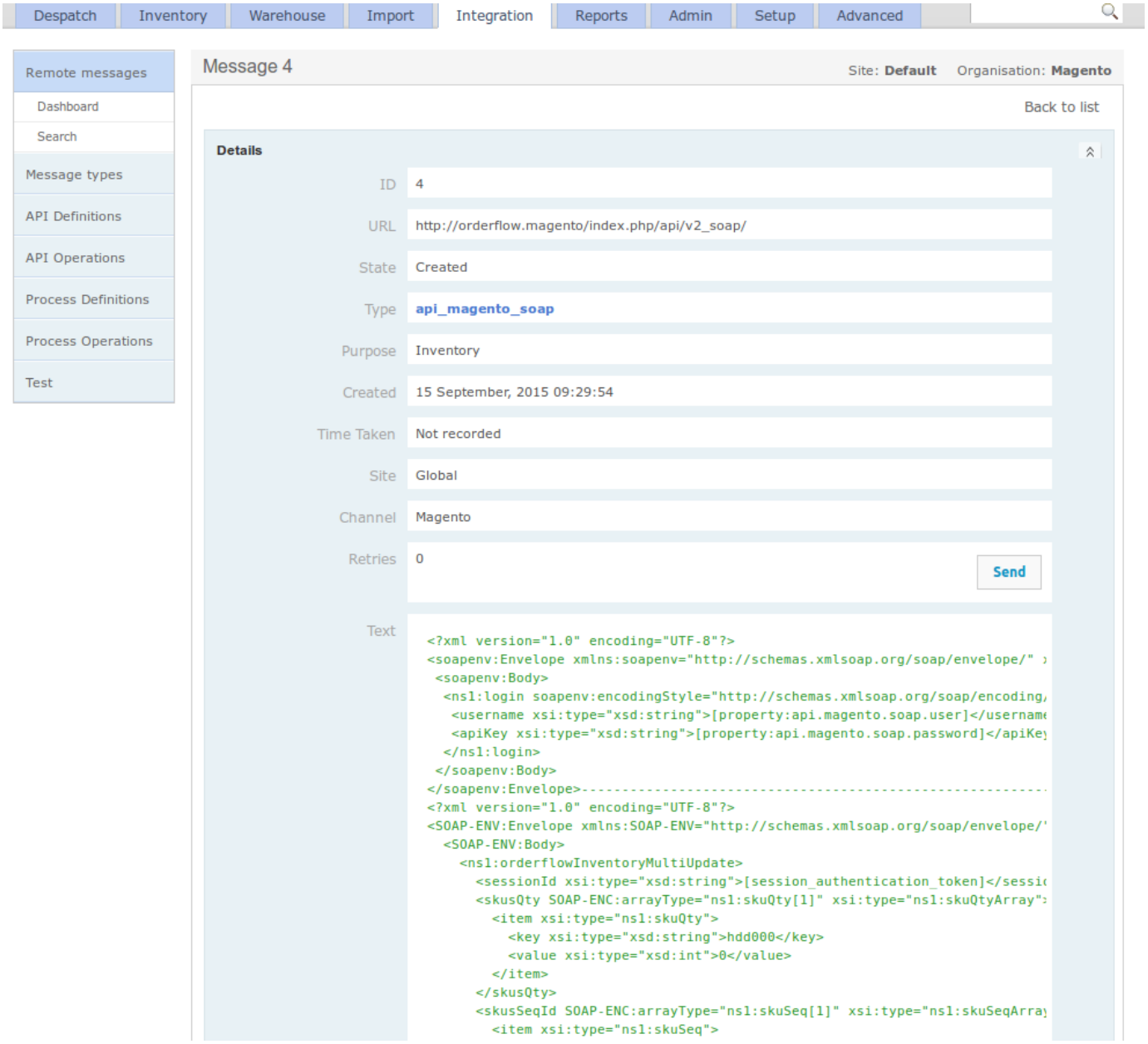


Note the *cached inventory record* for this product, which here shows as *10*.

A single stock push can be invoked using the *Push Stock* link, followed by the *Confirm* button.

On the OrderFlow side, this operation results in an outgoing message being queued, which can be found using the **Integration -> Remote Messages -> Search** menu.

Drilling into the detail of the most recently created record, the following queued message displays:



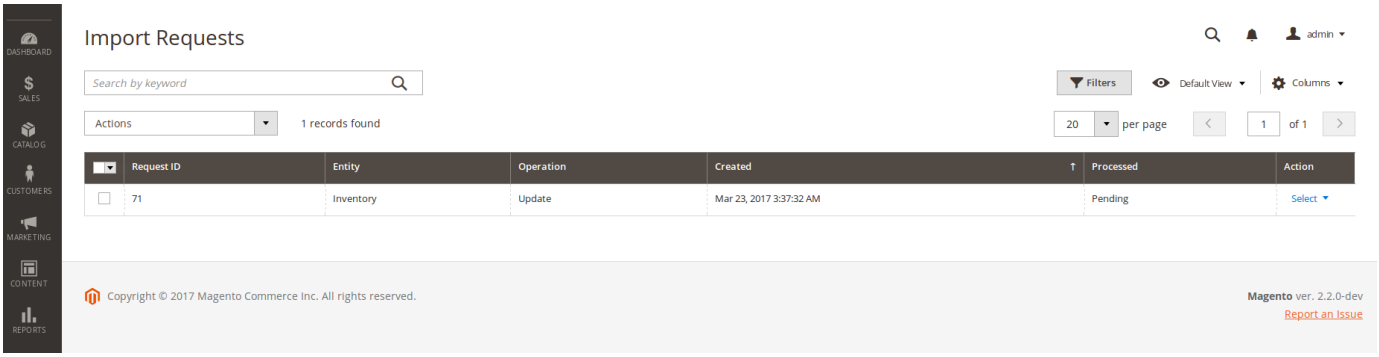
Note that OrderFlow is now configured to automatically send negative availability figures as zero.

Normally, sending of these messages will be automated. Additionally, the *Send* button can be used to manually push through the message to Magento.

Clicking on the *Send* button should result in the *Response* field being populated with text such as the following:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<ns1:realtimeDespatchOrderFlowInventoryRequestManagementV1UpdateResponse xmlns:ns1="http://local.magento2/
soap/default?services=realtimeDespatchOrderFlowInventoryRequestManagementV1">
    <result>
    Success - Message 71 Received
    </result>
</ns1:realtimeDespatchOrderFlowInventoryRequestManagementV1UpdateResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that the above response has indicated that message *71* has been received. This message can be found immediately on the Magento Admin Panel using the **Settings -> OrderFlow Imports -> Requests** menu, as shown below.



Each row in the grid shows the following:

Request ID: a numerical counter, simply an internal identifier within the extension for that record.

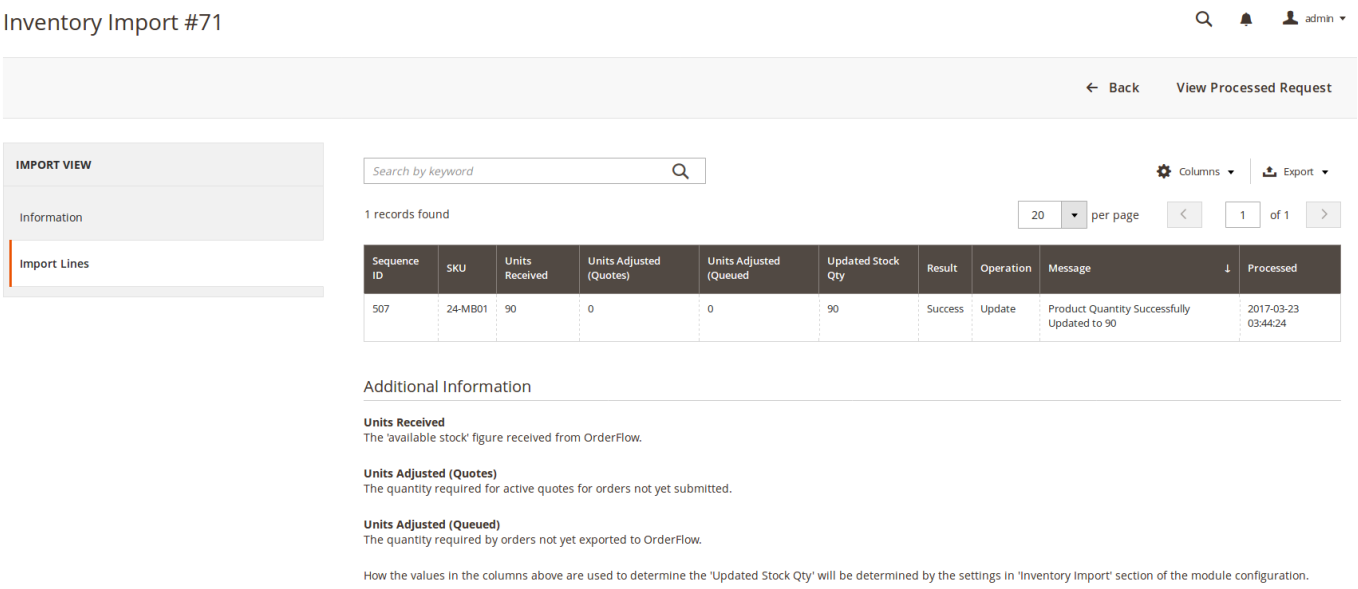
Entity: this is the type of import, in this case Inventory

Operation: what is being carried out on that entity, in this case we are updating the inventory

Note that the import record above is not created instantaneously on receipt of the message. Instead it is queued for creation through a scheduled job. Before the request has been processed, no import record will have been created, and the value for the column will be 'Pending'.

The *detail* of the request shows the same information as above, but also shows the XML received from OrderFlow.

It is also possible to follow the 'View Import Report' link once the message has been processed. This leads to a screen such as the following.



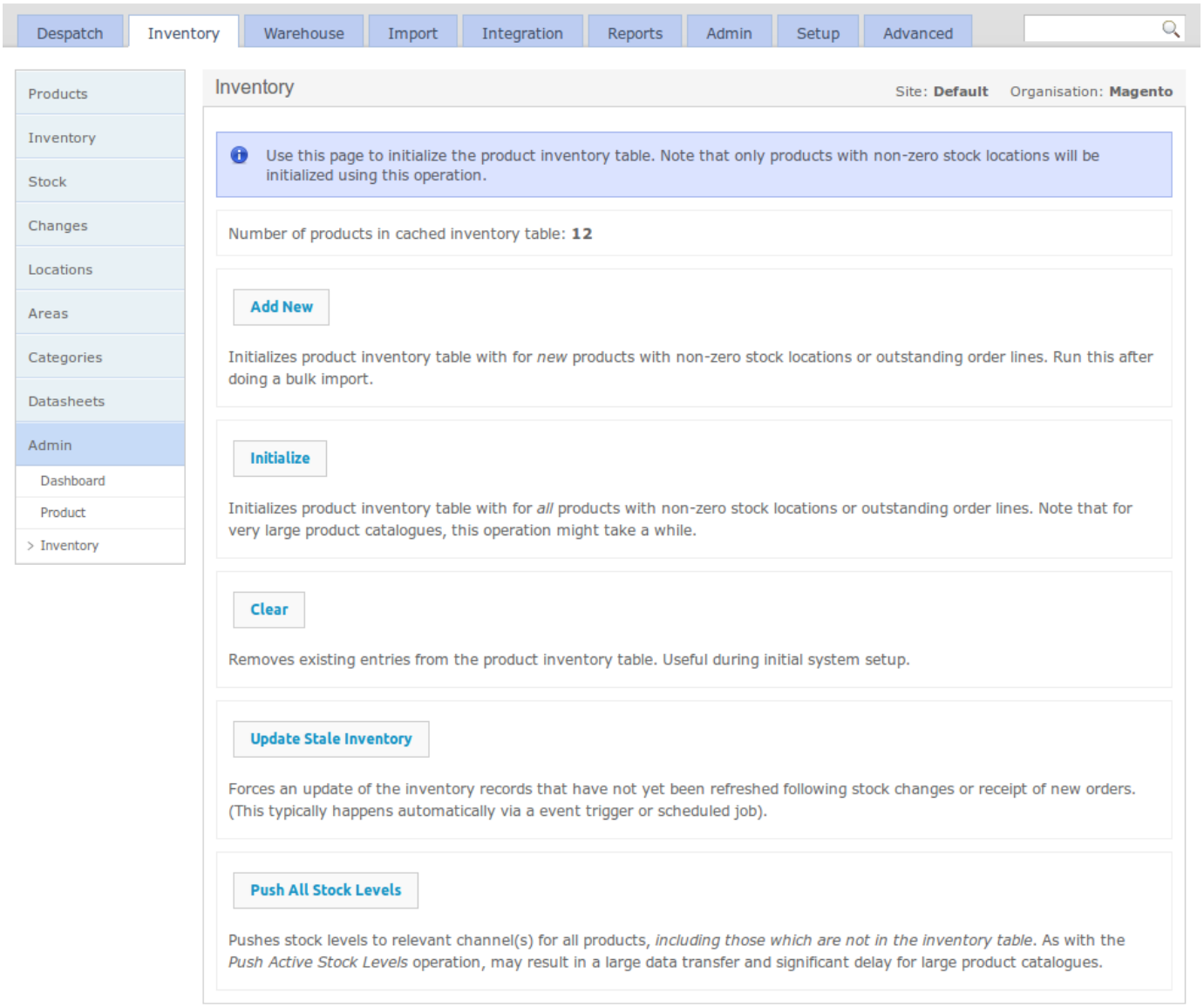
On this screen, we can see the outcome of import processing. Specifically, the screen informs us that the stock quantity of the SKU `24-MB01` has been set to 90.

Note the *Sequence ID* column. This displays the value of OrderFlow product inventory record ID at the point when the inventory notification is created. This field is used to ensure that stale updates are not applied, as described in the Handling Inventory Duplicates section.

Full Stock Push

A full stock push can be initiated to send similar stock updates to the ones applied above for all of the products known to OrderFlow.

To initiate a full stock push, navigate in OrderFlow to **Inventory -> Admin -> Inventory**, as shown below:



The *Push All Stock Levels* button can be used to initiate a product inventory export for all products. The stock push will be divided into multiple separate messages, with the maximum number of products sent per message set according to the system property *inventory.notification.chunk.size*. A sensible value for this property would be about 100 to 200.

Once invoked, the messages created would be queued in OrderFlow, then sent automatically to Magento. On Magento, the messages will be queued for processing as described in the previous section, and processed automatically. The *Batch Size* property in the Magento extension's Inventory Import configuration will control how many messages can be processed each time the inventory import job runs.

> **Note**
>
> In general a full stock push should not be necessary. Unless the inventory is reinitialised using the *Initialize* button, most of the stock updates will be discarded as duplicates. How this happens will be explained in more detail in the Handling Inventory Duplicates section below. *Initialize* button will recalculate the inventory level for all of the products on the system.

Incremental Stock Push

The manual stock push operations are only useful in exceptional circumstances. Ordinarily, inventory notifications are automatically send through an **incremental export** process.

Each time a stock change is made, or a new order is received, or an order is cancelled, OrderFlow will automatically recalculate the inventory for each of the affected products, creating a new inventory record for each product.

Another process on OrderFlow checks for all inventory records created since the last inventory notification took place, and creates a new inventory notification for each of these.

This process results in a continual stream of near real time inventory updates for products whose stock positions may have changed.

Handling Inventory Duplicates

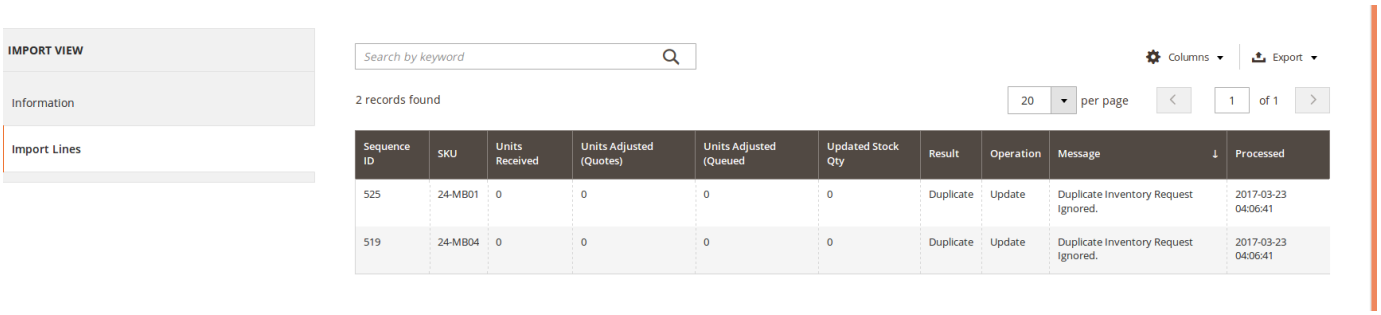
The OrderFlow Magento integration has been designed to deal with the consequences of real world situations that may occur in any enterprise computing environment; network outages may result in stock updates being received out of sequence, and even duplicates of the same message being received.

In the stock notification message, as well as sending a stock value, OrderFlow also sends the current inventory record ID as a product sequence number associated with the the update, as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="urn:Magento" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <ns1:orderflowInventoryMultiUpdate>
        <sessionId xsi:type="xsd:string">[session_authentication_token]</sessionId>
        <skusQty SOAP-ENC:arrayType="ns1:skuQty[1]" xsi:type="ns1:skuQtyArray">
            <item xsi:type="ns1:skuQty">
                <key xsi:type="xsd:string">hdd000</key>
                <value xsi:type="xsd:int">0</value>
            </item>
        </skusQty>
        <skusSeqId SOAP-ENC:arrayType="ns1:skuSeq[1]" xsi:type="ns1:skuSeqArray">
            <item xsi:type="ns1:skuSeq">
                <key xsi:type="xsd:string">hdd000</key>
                <value xsi:type="xsd:int">10</value>
            </item>
        </skusSeqId>
        <messageSeqId xsi:type="xsd:int">4</messageSeqId>
        </ns1:orderflowInventoryMultiUpdate>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The extension ensures that for each of the stock updates received, the system does not overwrite a more recently created stock record with one that is older, that is, a stale record.

If a stale update is received and processed, the extension will detect this and log an **Settings -> OrderFlow Imports -> Inventory** entry such as the following:



The extension will also detect messages that have been previously received and processed. In this case, the occurrence will be logged for auditing purposes, but no further processing will be attempted.

Unsent Orders and Active Quotes

The stock available figure sent to Magento from OrderFlow takes into account the orders that it has received at the point where the stock figure is calculated.

However, there are three sources of lag which may result in the reported figure that is updated in Magento *overstating* the actual available figure:

- there is a delay in OrderFlow between the time that the 'most recent' order is received in OrderFlow and the time that the available figure is calculated.
- there is a delay in OrderFlow between the time the available figure is calculated and the time that it is sent to Magento.
- there is a delay in Magento between the time the available figure is sent to Magento and the time it is processed on Magento.

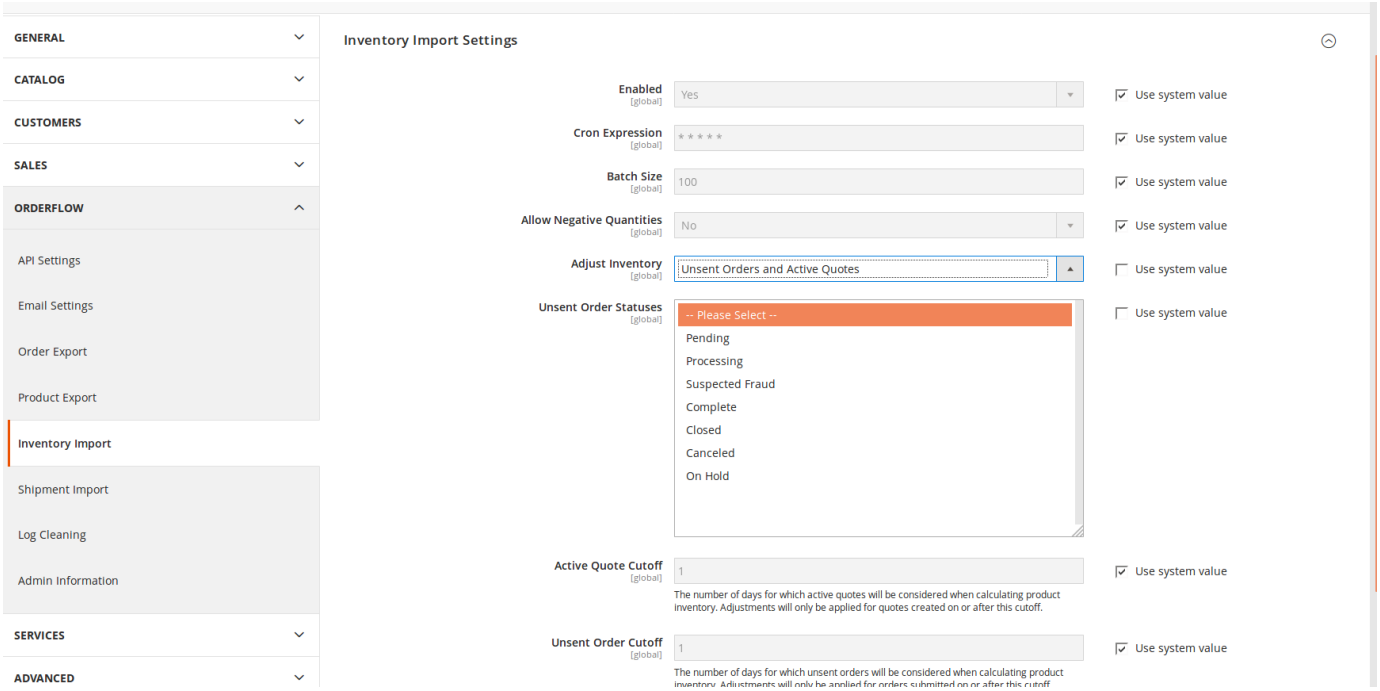
During the lag period, the following may have happened:

- new orders may have been taken for the sku concerned, but not yet sent to OrderFlow.
- new orders may have been added to cart, but not yet completed on Magento.

The OrderFlow Magento integration includes a feature which allows for these scenarios to be taken into account.

Magento Configuration

The following screenshot shows the inventory import configuration in Magento.



The **Adjust Inventory** control allows you to determine whether to allow inventory to be adjusted based on *unsent orders*, *unsent orders and active quotes*, or not at all. The options for this are:

- **No**: no inventory adjustment is made based on unsent orders and/or active quotes. This is the backward compatible setting, and is the default option.
- **Unsent orders**: the Magento module takes into account orders that have been submitted since the date of the last order used in calculating the inventory level in OrderFlow.
- **Unsent orders and active quotes**: the Magento module takens into account not only unsent orders, but active quotes that have not yet been confirmed as orders. Note that a quote is created when a user adds items into their 'shopping cart'.

The remaining configuration options set on the screen above.

- **Unsent Order Statuses**: determines the status of orders that will be considered as unsent. For example, orders for which the payment was unable to complete may be excluded from the list of orders that would be used to adjust the inventory for a product.
- **Active Quote Cutoff**: the maximum age for active quotes that might be used to adjust the inventory. This is useful to ensure that old or stale quotes don't artificially depress the adjusted product inventory. This value is set to *1* by default.s
- **Unsent Order Cutoff**: the maximum age for unsent orders that might be used to adjust the inventory. This allows exclusion from the inventory adjustment old orders that for whatever reason have not have been passed through to OrderFlow.

Shipment Import

Once the job of fulfilling an order has been completed, then OrderFlow needs to inform Magento that this has happened. This serves two purposes.

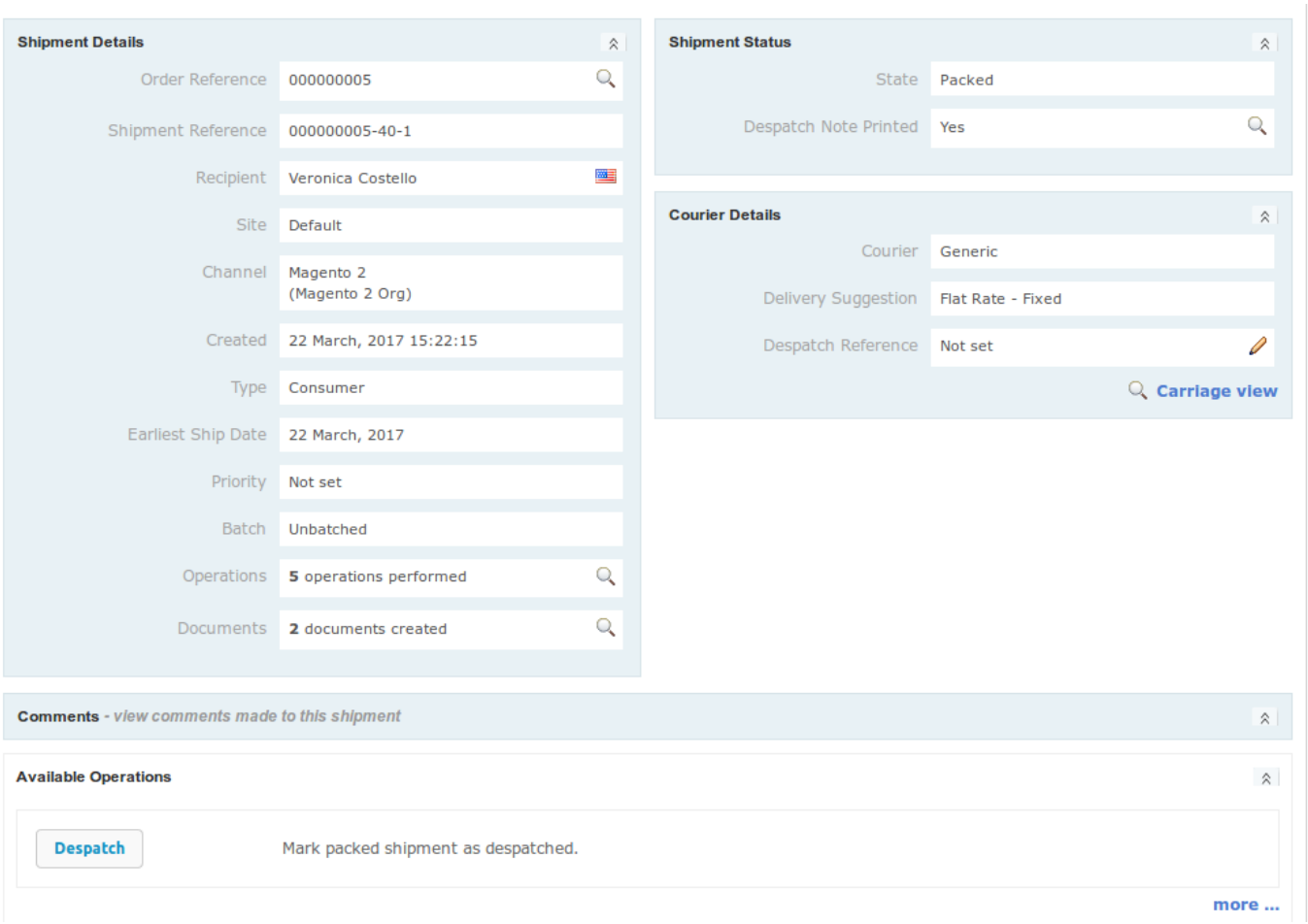
- to allow the order to be closed off, so that it no longer appears as requiring fulfilment.
- to pass back additional information that may be used for notifying customers of despatch, such as courier details and tracking references.

The shipment import on Magento is triggered at the point where the shipment is marked as picked and packed. Of course, the process of allocating stock, picking and packing will need to take place, but these operations are outside of the scope of this document.

There are a number of ways of marking a shipment as despatched on OrderFlow.

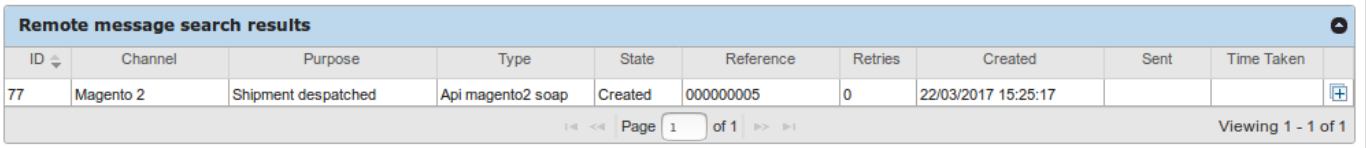
- from the shipment detail screen, using the *Despatch* button. This allows a single shipment to be marked as despatched.
- from the shipment search screen. This is only possible if configured correctly, and if the search has been limited to 'Packed' shipments. This option allows multiple shipments to be despatched manually.
- manually, using the **Despatch -> Bulk Operations -> Despatch** menu. This allows all packed shipments to be despatched by courier via single operation.
- automatically via a scheduled job, either at specific time intervals or after a configurable delay.

In the example below, we will navigate to a shipment that has been packed, and despatch it using the *Despatch* button below:



Once invoked, the outcome is similar to that of the Inventory Import, in that a new message is queued for sending, and will appear listed in the **Integration -> Remote Messages -> Search** menu.

The specific records will be easily found by applying the additional filters for *Category* ('API Integration') and *Purpose* ('Shipment despatched'). Once sent, the shipment will appear on the listing as shown below:



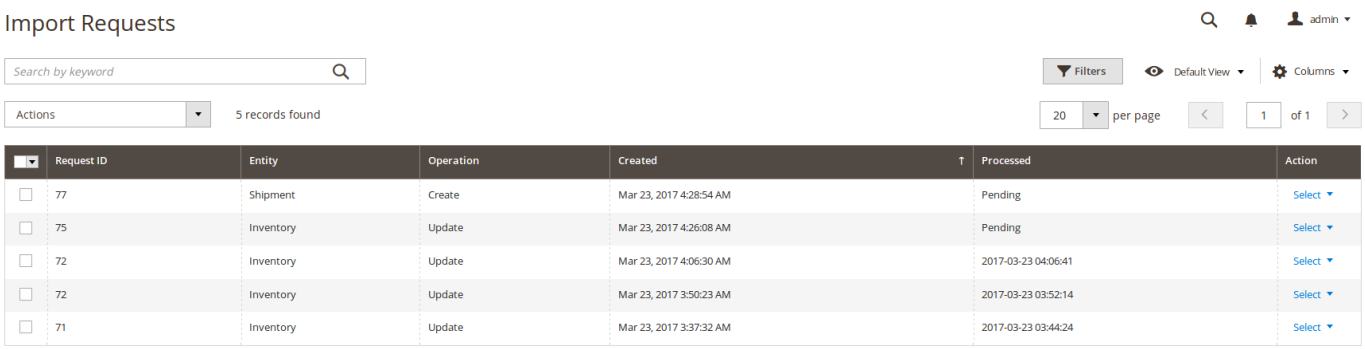
The body of the shipment despatch message contains details of the shipment despatch, the courier used, and the tracking number, if appppropriate.

It also contains a listing of the individual product and quantity combinations that have been included in the shipment.

This allows the extension to support partial fulfilments. If only a part of the initial order has been fulfilled, a shipment will be created for the order in Magento but the order will be kept open for further fulfilment updates.

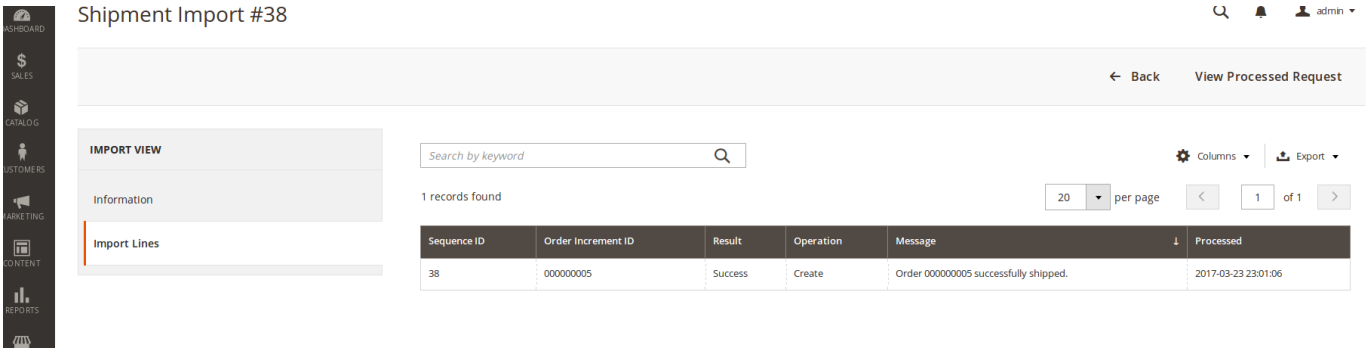
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ns1="[property:api.magento2.soap.base.url]?
services=realtimeDespatchOrderFlowShipmentRequestManagementV1">
    <SOAP-ENV:Body>
    <ns1:realtimeDespatchOrderFlowShipmentRequestManagementV1CreateRequest>
        <sessionId>[session_authentication_token]</sessionId>
        <orderIncrementId>000000005</orderIncrementId>
        <skuQty>
        <item>
            <sku>24-MB01</sku>
            <qty>1</qty>
        </item>
        </skuQty>
        <comment>Shipment Created</comment>
        <email>1</email>
        <includeComment>1</includeComment>
        <courierName>Generic</courierName>
        <serviceName></serviceName>
        <trackingNumber></trackingNumber>
        <dateShipped>2017-03-22 15:25:16</dateShipped>
        <messageSeqId>[message_id]</messageSeqId>
    </ns1:realtimeDespatchOrderFlowShipmentRequestManagementV1CreateRequest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

As soon as the notification request has been sent to Magento, it will be visible in the Magento Admin Panel under the *Settings -> OrderFlow Imports -> Requests* menu, as shown below.



Again, as with the inventory import, the Processed field is not populated immediately, but is only set once the request has been processed. The timing of this will depend on the configuration settings for the Shipment Import, as described in the Magento Configuration section.

Once Processed is set, the import report log can be viewed. Navigate to *Settings -> OrderFlow Imports -> Shipments* to this will show the details of the import, which will confirm, as shown below that the shipment has been created.



Navigating to the order within Magento will show the shipment against the order. If no more items need to be fulfilled, the expected behaviour at this point is for the order to be marked as complete.