NYCE.LOGIC

- Standard integrations

extendaretail

Table of contents

Standard API Integrations **2**

Purpose **3**

Fortnox **4**
- Overview 4
- Software requirements 4
- Information from Customer 4
- Security 4
- Configuration 4
  - Installation 4
  - Settings 4
  - Important configurations 5
- Defined messages 5
  - Item import 5
  - Customer order/return import 5
  - Customer order/return import (already invoiced) 6
  - Deliveries export 6
  - Balance export 6

WooCommerce **7**
- Overview 7
- Software requirements 7
- Information from Customer 7
- Security 7
- Configuration 7
  - Installation 7
  - Settings 7
  - Defined messages 8
  - WooCommerceItem - Items in WooCommerce 8
  - WooCommerceOrder - Customer orders in WooCommerce 8
  - WooCommerceBalance - Item balance in WooCommerce 8
  - WooCommerceDelivery - Customer orders in WooCommerce 8

Shopify **9**
- Overview 9

Software requirements 9
Information from Customer 9
Security 9
Configuration 9
Installation 9
Settings 9
Defined messages 10
GetItems - Items in Shopify 10
GetOrders - Customer orders in Shopify 10
UpdateBalance - Inventory in Shopify 10
UpdateOrders - Customer orders in Shopify 10
CreateRefunds - Refunds in Shopify 10

Specter **11**
Overview 11
Software requirements 11
Information from Customer 11
Configuration 11
Installation 11
Settings 11
Important configurations 12
Defined messages 12
Item import 12
Customer import 12
Customer order import 12
Deliveries export 12
Purchase order import 13
Purchase order deliveries export 14
Transactions export 14

Standard API Integrations

NYCE.LOGIC provide a couple of standard integration modules which simplifies and makes the implementation process more cost efficient for the customer and it's clients if they use the associated system.

Currently integrations modules exists for Shopify, WooCommerce, Fortnox and Specter.

The standard integration modules consist of integration settings for import and export and different message types (which message types are included in each module depends on which system we integrate with).

In general the standard integration handles Items, Balance, Customer Orders, Customer Order Delivery and track & trace information.

NYCE.LOGIC uses the API details you provide from the associated system, and then we connect and integrate with your prefered system.

Fortnox

Overview

Fortnox is a web based ERP for small businesses.
(https://www.fortnox.se/)

Software requirements

1. NYCE.LOGIC 3.6 or later
2. Work Server 3.6 or later
3. Valid license for module Fortnox3 (3.7+)
4. Fortnox and optional warehouse module

Information from Customer

To set up the integration between a Fortnox account and NYCE.LOGIC WMS we need information about Access Token, Client Secret and Base URL.

Security

TLS1.2, SSL3

Configuration

Installation

To be able to integrate with NYCE.LOGIC WMS with Fortnox you need to apply the latest NYCE.LOGIC WMS version including Fortnox3DatabaseVersion; this will enable the settings and message types needed for the integration. You also need to add licensekey including module Fortnox.

Settings

The following settings are a prerequisite to use the Specter integration. All of the information below is supplied from Fortnox for the specific system that should be integrated. NYCE.LOGIC WMS requires that integration has been connected to a user account in Fortnox:

- Fortnox.BaseUrl - Contains the base URL for the Fortnox API
- Fortnox.AccessToken - Token for accessing the customer's Fortnox environment
- Fortnox.WarehouseModule - Flag to indicate if WarehouseModule is enabled in Fortnox.
- Fortnox.CreateInvoice - Flag to indicate if invoice should be created in Fortnox.
- Fortnox.PrefixCustomerOrderNumber - Applies a prefix to customerordernumber during import and removes during export
- Fortnox.StockPointCode - StockPointCode in Fortnox which should receive stock adjustments **NOTE!** Only StockPointCode per Warehouse in NYCE.LOGIC can be used. If multiple StockPoints is needed then multiple warehouses must be set up in NYCE.LOGIC.
- Fortnox.SkipInvoicedForLastModified - Only used when import method OrdersByLastModified is used. If value is true all invoiced orders are skipped during import. If false then invoiced orders will be imported **unless** they already exist in NYCE.LOGIC.

The system specific settings for NYCE.LOGIC WMS is the following:

- Fortnox.Setting.CustomerOrderTemplate = Template to use when creating Customer order
- Fortnox.Setting.CustomerReturnTemplate = Template to use when creating Customer return
- Fortnox.Setting.CustomerTemplate = Template to use when creating customer
- Fortnox.Setting.UseInvoiceAddressAsDeliveryAddress = If true invoice address is applied also as delivery address

Important configurations
Based on the decision of allowing partial delivery or not you have to at the same time set restorderhandling setting or parameter on OrderType/CustomerOrder

Defined messages

Item import
Items import will add and update all items.

Customer order/return import
Customer orders/returns import will add and update all orders where an invoice has not been created in Fortnox and that has not yet been processed in the picking proposal.

Parameters used during import:
- filter = invoicenotcreated

Return orders will be imported as customer orders with order type RETURN.

You can use the settings “Fortnox.Setting.CustomerOrderTemplate” or “Fortnox.Setting.CustomerReturnTemplate” and “Fortnox.Setting.CustomerTemplate” to set the code of the templates to use to set default values.

If setting “Fortnox.PrefixCustomerOrderNumber” is set then the prefix will be added to the customer order number during import.

Customer order/return import (already invoiced)
Orders that have been invoiced are normally not imported. But in some scenarios, especially when Fortnox is connected to an e-commerce site, the order might be paid/invoiced when created and is therefore filtered out using the ordinary order import method.

To import orders that are already invoiced, add a new integration message source (or change the existing) and change the Mask from “Orders” to “OrdersByLastModified”.
This will fetch all orders since the date set in the setting Fortnox.LastImportDate (this is automatically updated every time the import method runs).
In combination with this there is also the setting Fortnox.SkipInvoicedForLastModified. If the setting is true (default) the importmethod work in the same way as the ordinary order import.
If the setting is false it will import orders that are invoiced **unless** they already exist in NYCE.LOGIC.

Deliveries export
Export of deliveries will be made for CustomerOrderDeliveries with status Shipped, Completed. The system will export shipment sequence numbers as well as delivered quantities. If WarehouseModule is enabled the exported delivery is marked as Warehouse Ready in Fortnox.

If setting “Fortnox.PrefixCustomerOrderNumber” is set then the prefix will be removed from the customer order number during export if it has been applied earlier.

Balance export

Export of items in stock will export all items to Fortnox. This is not a relative update but a fixed update of the current balance in the warehouse. Please be careful to first export all deliveries so both systems have knowledge of fulfilled orders/deliveries.

Stock adjustment

Exports stockadjustments to Fortnox for the specified StockPointCode. The export works by importing the current balance from Fortnox and calculating the differences against the balance in NYCE.LOGIC and then exporting a Warehouse Inbound Document to Fortnox with the stock adjustments.

NOTE! This export requires both that Warehouse Module is activated and that StockPointCode is set. If either is false the export will not run.

WooCommerce

Overview

WooCommerce is a web based ERP for small businesses.
(https://woocommerce.com/)

Software requirements

1. NYCE.LOGIC 3.6 or later
2. Work Server 3.6 or later
3. Valid license for module WooCommerce (3.7+)

Information from Customer

To set up the integration between a WooCommerce store account and NYCE.LOGIC WMS we need the information about URL and Rest API Key Details, such as Consumer Key and Consumer Secret.

Security

TLS1.0, TLS1.1, TLS1.2

Configuration

Installation

To be able to integrate with NYCE.LOGIC WMS with WooCommerce you need to apply the latest NYCE.LOGIC WMS version including WoocommerceDatabaseVersion; this will enable the settings and message types needed for the integration. You also need to add licensekey including module Woocommerce.

Settings

- Woo.BaseUrl - Url till WooCom
- Woo.ConsumerKey - Användarnyckel
- Woo.ConsumerSecret - Användarlösen/secret
- Woo.AuthorizedHeader - true/false. Anger om Basic Authorization header ska användas eller consumer_key/consumer_secret headers
- Woo.PrefixCustomerOrderNumber - Applies a prefix to customerordernumber during import and removes during export
- Woo.AutomaticallyPaybackRefunds - Defines if an automatic payback of a refund should be done to the customer. Default is true

Defined messages

WooCommerceItem - Items in WooCommerce
We import all items with WooCommerceID from WooCommerce, in NYCE.LOGIC WMS WooCommerceID value is stored in Barcode type WID.

Parameters used during import:

- per_page = 100 # Fetch in batches of 100 items

WooCommerceOrder - Customer orders in WooCommerce
We import customer orders from WooCommerce for those orders that meet the condition Customer Order status = Processing.

Parameters used during import:

- per_page = 100 # Fetch in batches of 100 items
- status = processing # Only fetch orders in status processing

If setting “Woo.PrefixCustomerOrderNumber” is set then the prefix will be added to the customer order number during import.

WooCommerceBalance - Item balance in WooCommerce
We export item balance to WooCommerce for all items with WooCommerceID (value stored in Barcode type WID, with WooCommerceID as value).

WooCommerceDelivery - Customer orders in WooCommerce
We export information about customer orders we delivered.

If setting “Woo.PrefixCustomerOrderNumber” is set then the prefix will be removed from the customer order number during export if it has been applied earlier.

Shopify

Overview

Shopify is a web based ERP for small businesses.
(https://www.shopify.com/)

Software requirements

1. NYCE.LOGIC 3.6 or later
2. Work Server 3.6 or later
3. Valid license for module Shopify (3.7+)

Information from Customer

To set up the integration between a Shopify store account and NYCE.LOGIC WMS we need the information about URL, both BaseURL and LocationID, and API details such as ApiKey and ApiPassword.

Security

TLS1.2, SSL3

Configuration

Installation

To be able to integrate with NYCE.LOGIC WMS with Shopify you need to apply the latest NYCE.LOGIC WMS version including ShopifyDatabaseVersion; this will enable the settings and message types needed for the integration. You also need to add licensekey including module Shopify.

Settings

- Shopify.ApiKey
- Shopify.ApiPassword
- Shopify.BaseUrl
- Shopify.LocationId
- Shopify.PrefixCustomerOrderNumber
- Shopify.SendRefundNotification

Defined messages

GetItems - Items in Shopify
We import items from Shopify without any conditions.

GetOrders - Customer orders in Shopify
We import customer orders from Shopify. The condition fulfillment_status = unshipped must be met in order to let us import the order. We import a maximum of 50 orders each time.

Parameters used during import:

- fulfillment_status = unshipped # Only fetch orders with fulfillment status unshipped
- limit = 50 # Only fetch 50 orders per call

If setting “Shopify.PrefixCustomerOrderNumber” is set then the prefix will be added to the customer order number during import.

if setting “Shopify.OnlyImportPaidOrders” is set then only orders in status Paid and Partially_refunded will be imported.

If the CustomerOrderTemplate setting is used then the AllowPartialDelivery flag for CustomerOrder must be set in the template. If a template is not used, AllowPartialDelivery will default to false.

UpdateBalance - Inventory in Shopify
We export item balance for all items with the custom property “SHOPIFY_INVENTORY_ID”.

UpdateOrders - Customer orders in Shopify
We export information about delivered customer order if the condition CustomerOrderDelivery.Status = Delivered with substatus Completed is met.

CreateRefunds - Refunds in Shopify
We create a refund in Shopify based on a CustomerReturnDelivery. If setting SendRefundNotification is active a notification is sent from Shopify to the customer

Specter

Overview

Specter is a web based ERP for small businesses.
(https://www.specter.se)

Software requirements

1. NYCE.LOGIC 3.6.
2. Work Server 3.6.
3. Valid license for module Specter (3.7+)
4. Specter 3.x with enabled integration 3.0

Information from Customer

Configuration

Installation

To be able to integrate with NYCE.LOGIC WMS with Shopify you need to apply the latest NYCE.LOGIC WMS version including SpecterDatabaseVersion; this will enable the settings and message types needed for the integration. You also need to add licensekey including module Specter

Settings

The following settings are a prerequisite to use the Specter integration. All of the information below is supplied from Specter for the specific system that should be integrated. NYCE.LOGIC WMS requires the API - “Lagersystem” permissions in Specter:

- Specter.ApiUserId
- Specter.ApiUserKey
- Specter.sbmid
- Specter.md5Key1
- Specter.md5Key2
- Specter.PrefixCustomerOrderNumber
- Specter.PrefixPurchaseOrderNumber

The system specific settings for NYCE.LOGIC WMS is the following:

- Specter.Setting.CustomerOrderTemplate = Template to use when creating Customer order
- Specter.Setting.CustomerTemplate = Template to use when creating customer
- Specter.CreateInvoice = Defines if invoice should be created when delivery is exported

- Specter.SendInvoiceMail = Defines if mail should be sent to customer when invoice is created
- Specter.SendDeliveryEmail = Defines if mail should be sent to customer when delivery has been confirmed

Important configurations
Based on the decision of allowing partial delivery or not you have to at the same time set restorderhandling setting or parameter on OrderType/CustomerOrder

If you allow partial delivery you also must enable restorder handling to be able to fully deliver the orders to Specter.

Defined messages

Item import
Items import will add and update changed items since the last import.
The import should be marked as “Repeat” as it can contain more than 500 items.

Customer import
Customer import will add and update changed customers since the last import. The import should be marked as “Repeat” as it can contain more than 500 customers.

You can use the settings “Specter.Setting.CustomerTemplate” to set the code of the template to use to set default values.

Customer order import
Customer orders import will add and update changed orders since the last import. The import should be marked as “Repeat” as it can contain more than 500 orders.

You can use the settings “Specter.Setting.CustomerOrderTemplate” and “Specter.Setting.CustomerTemplate” to set the code of the templates to use to set default values.

If setting “Specter.PrefixCustomerOrderNumber” is set then the prefix will be added to the customer order number during import.

Deliveries export
Export of deliveries will be made for CustomerOrderDeliveries with status Shipped, Completed. The system will export shipment sequence numbers as well as delivered quantities.
The export should be marked as “Repeat” because it has a fetch size of 100 set as default.

If setting “Specter.PrefixCustomerOrderNumber” is set then the prefix will be removed from the customer order number during export if it has been applied earlier.

Purchase order import

Import of purchase orders will be made for added or updated purchase orders since last import. The import should be marked as “Repeat” as it can contain more than 500 purchases.

If setting “Specter.PrefixPurchaseOrderNumber” is set then the prefix will be added to the customer order number during import.

Purchase order deliveries export
Export of purchase deliveries will default be made for deliveries with status Closed, Completed. So be careful to close notifications when receival is done.

If setting “Specter.PrefixPurchaseOrderNumber” is set then the prefix will be removed from the customer order number during export if it has been applied earlier.

Transactions export
Export of transactions is by default setup to export the following transaction types:

- INVENTORY_COUNTING
- ZERO_INVENTORY_COUNTING
- All transaction codes marked as AllowManualProcessing

The transactions will be exported as “relative” values with positive and negative adjustments.
Balance export

Export of items in stock will export all items to Specter. This is not a relative update but a fixed update of the current balance in the warehouse. Please be careful to first export all deliveries (both orders and purchase) so both systems have knowledge of fulfilled orders/deliveries.