WooCommerce Integration

Product Development - 2024-12-06 - Extended Modules

Overview

Cavallo's integration with WooCommerce handles automatic syncing of inventory and sales information between SalesPad/GP and a WooCommerce website. Product and inventory level information is pushed from SalesPad/GP to the website so that customers have visibility of which products are available. Sales orders created by customers on the website are pulled down to SalesPad/GP so that they can be processed and fulfilled. Tracking information and notes for each sales order are pushed back to the website for customer visibility. Payment information can be imported from the website, and PayFabric authorizations can be imported for capturing within SalesPad. When sales orders are voided, those updates are communicated back to the website.

This document covers the configuration for all components of the WooCommerce integration.

Table Of Contents

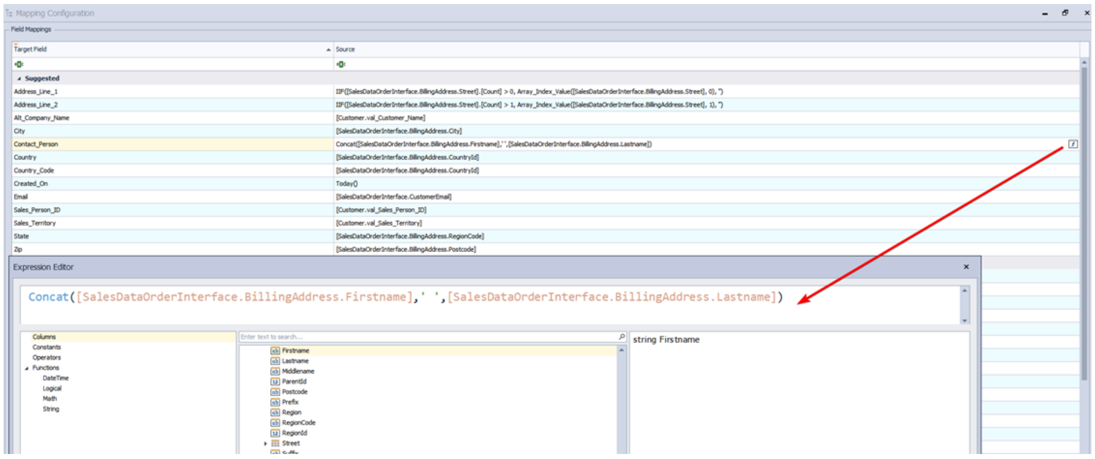
General Information On Expressions

Overview

Many integration settings allow the use of expressions to configure how internal and external entities should be matched or to designate which values are assigned from external to internal entities.

Each expression editor will have a set of source objects to match or map from, and a set of destination objects to be matched or populated. These editors allow additional complexity beyond simply mapping source object fields to destination object fields.

As an example, below is the default expression for *Customer Bill To Address Assignment Mapping - Contact_Person.* In this case, the WooCommerce order is the source object, and a SalesPad/GP CustomerAddr is the destination object. When a new SalesPad/GP CustomerAddr is created in the process of creating a customer for an imported order, this expression is used to populate the Contact_Person field.

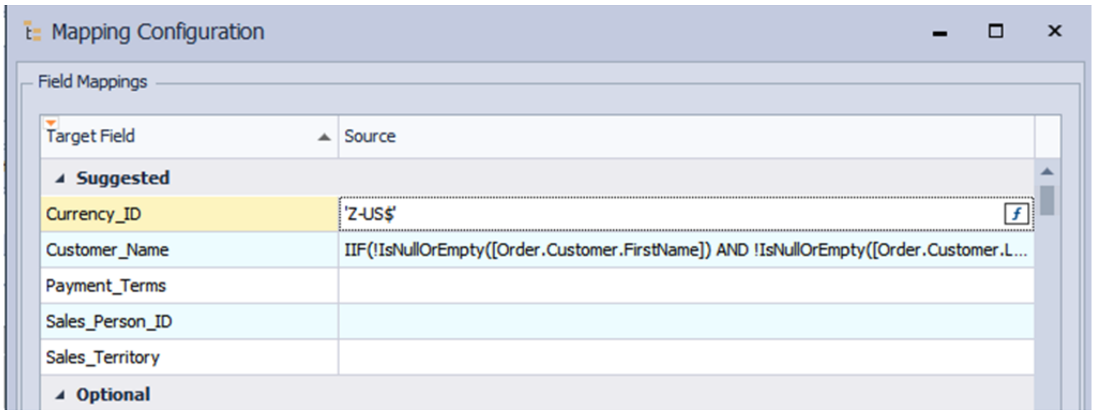


Target Field	Source
Suggested	
Address_Line_1	IIF([SalesDataOrderInterface.BillingAddress.Street].[Count] > 0, Array_Index_Value([SalesDataOrderInterface.BillingAddress.Street], 0), '')
Address_Line_2	IIF([SalesDataOrderInterface.BillingAddress.Street].[Count] > 1, Array_Index_Value([SalesDataOrderInterface.BillingAddress.Street], 1), '')
Alt_Company_Name	[Customer.val_Customer_Name]
City	[SalesDataOrderInterface.BillingAddress.City]
Contact_Person	Concat([SalesDataOrderInterface.BillingAddress.Firstname],' ',[SalesDataOrderInterface.BillingAddress.Lastname])
Country	[SalesDataOrderInterface.BillingAddress.CountryId]
Country_Code	[SalesDataOrderInterface.BillingAddress.CountryId]
Created_On	Today()
Email	[SalesDataOrderInterface.CustomerEmail]
Sales_Person_ID	[Customer.val_Sales_Person_ID]
Sales_Territory	[Customer.val_Sales_Territory]
State	[SalesDataOrderInterface.BillingAddress.RegionCode]
Zip	[SalesDataOrderInterface.BillingAddress.Postcode]

```
Concat([SalesDataOrderInterface.BillingAddress.Firstname],' ',[SalesDataOrderInterface.BillingAddress.Lastname])
```

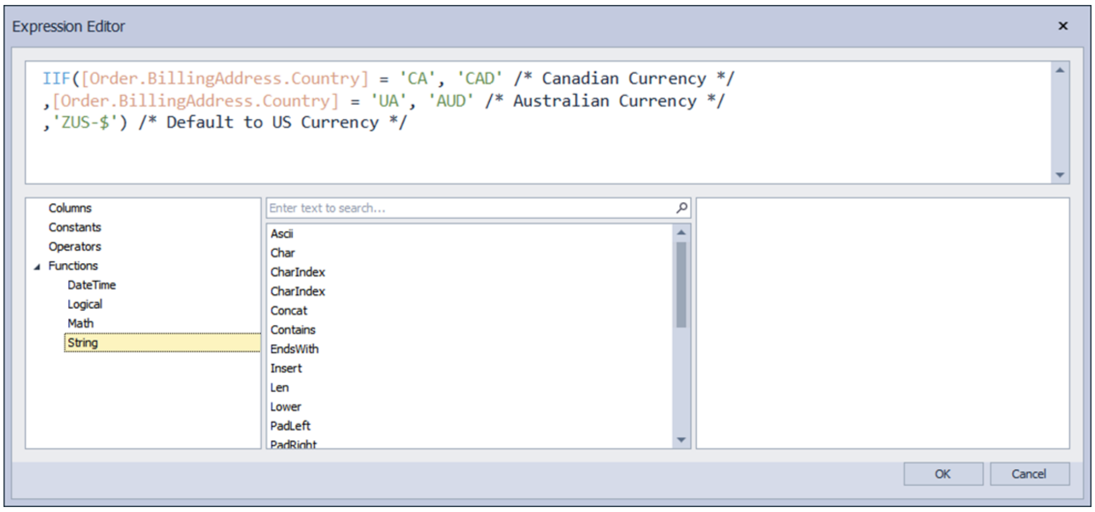
Rather than a single customer name field with the customer’s full name, WooCommerce has separate first and last name fields, so the Contact_Person field cannot be populated by simply mapping one field to another. The Concat function is used to append the WooCommerce Customer last name to the first name, with a space in between. If the customer’s first name is “Jane” and last name is “Doe”, the Contact_Person field will be “Jane Doe”.

Expression editors can also be used to assign hardcoded values when populating fields. In the below example configuration of the *Customer Assignment Mapping* setting, all customers will be created with value “Z-US$” in the Currency_ID field.



Target Field	Source
Suggested	
Currency_ID	'Z-US$'
Customer_Name	IIF(!IsNullOrEmpty([Order.Customer.FirstName]) AND !IsNullOrEmpty([Order.Customer.L...
Payment_Terms	
Sales_Person_ID	
Sales_Territory	
Optional	

For an example of a more complex use case, the below expression includes an IIF (if statement) conditional. Comments may be included to provide notes for future reference.


Expression Editor

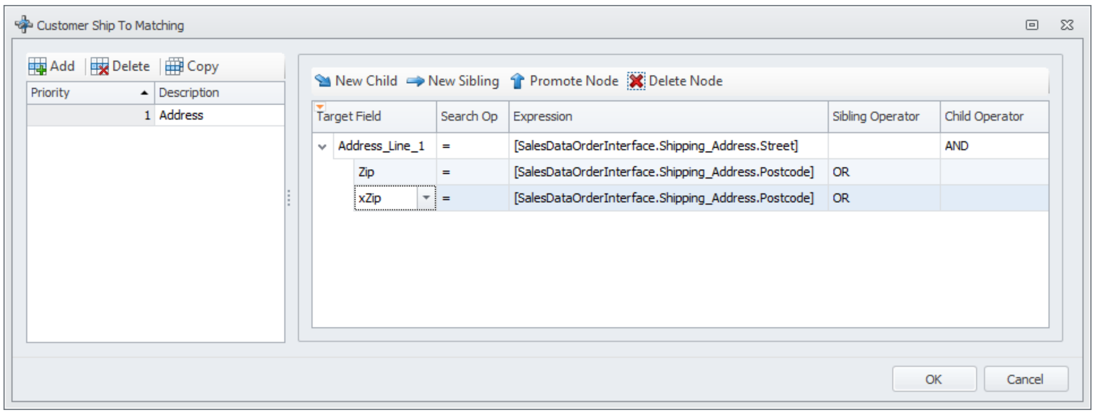
```
IIF([Order.BillingAddress.Country] = 'CA', 'CAD' /* Canadian Currency */
,[Order.BillingAddress.Country] = 'UA', 'AUD' /* Australian Currency */
,'ZUS-$') /* Default to US Currency */
```

The complete expression language documentation can be found here.

Sibling and Child Relationship

A combination of sibling and child expressions can be used during matching. The expressions can be joined using logical operators such as AND, OR, etc.

For example, to match the customer address on the Address Line 1 field and Zip field or xZip user field, set the Address Line 1 child operator to AND and the Zip and xZip sibling operator to OR:


Customer Ship To Matching

Priority	Description
1	Address

Target Field	Search Op	Expression	Sibling Operator	Child Operator
Address_Line_1	=	[SalesDataOrderInterface.Shipping_Address.Street]		AND
Zip	=	[SalesDataOrderInterface.Shipping_Address.Postcode]	OR	
xZip	=	[SalesDataOrderInterface.Shipping_Address.Postcode]	OR	

Product Export

Overview

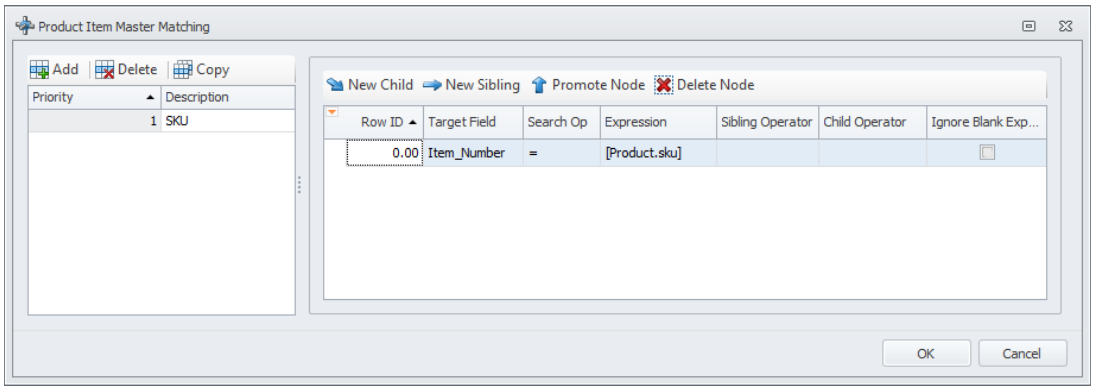
Export Item Masters from SalesPad/GP to WooCommerce. This component provides flexibility to specify which items and associated values are pushed to WooCommerce.

General Settings

Number Of Products Per Export Page - *Specify the number of products in each page of the Product Export. (Max: 100)*

Matching Settings

Product Item Master Matching - *This setting will be used to try and match every WooCommerce item to a GP Item. If a match is found, the GP Item will be used to update the WooCommerce product.*


Product Item Master Matching

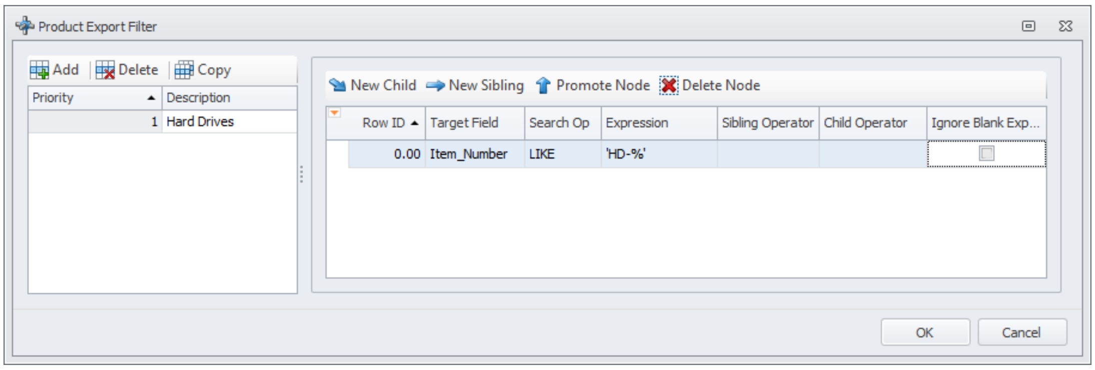
Priority	Description
1	SKU

Row ID	Target Field	Search Op	Expression	Sibling Operator	Child Operator	Ignore Blank Exp...
0.00	Item_Number	=	[Product.sku]			☐

Lookup Settings

Product Export Filter - *Define the criteria for which GP Item Numbers should be exported to WooCommerce. If left blank, all active items will be exported.*

Only one priority row should be added to the left grid. The conditions on the right will be used to determine which Item Masters will be exported to WooCommerce. In the example below, the system will export all items that start with *HD-*.

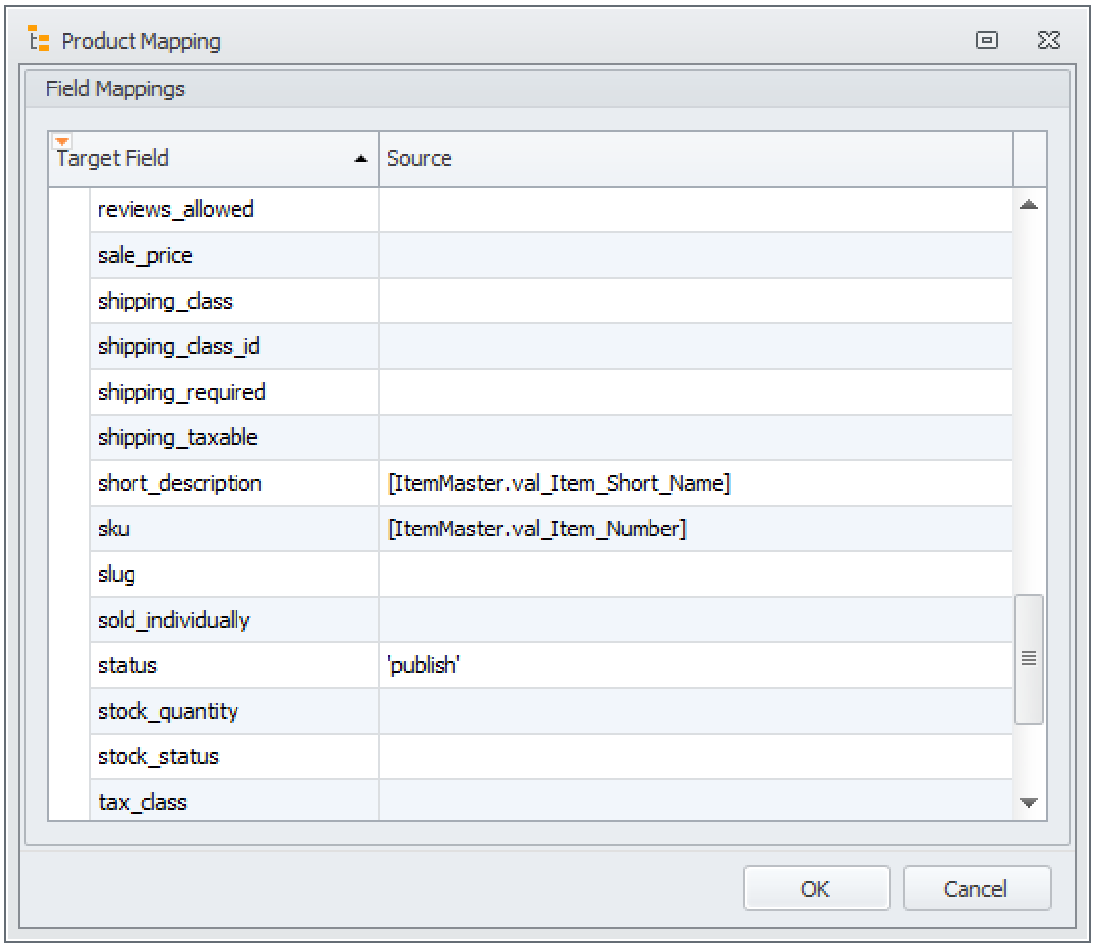

Product Export Filter

Priority	Description
1	Hard Drives

Row ID	Target Field	Search Op	Expression	Sibling Operator	Child Operator	Ignore Blank Exp...
0.00	Item_Number	LIKE	'HD-%'			☐

Assignment Settings

Product Mapping - *Assign the WooCommerce product fields from the GP item master.*



Target Field	Source
reviews_allowed	
sale_price	
shipping_class	
shipping_class_id	
shipping_required	
shipping_taxable	
short_description	[ItemMaster.val_Item_Short_Name]
sku	[ItemMaster.val_Item_Number]
slug	
sold_individually	
status	'publish'
stock_quantity	
stock_status	
tax_class	

Processing
The Product Export bulk loads products from WooCommerce, then it updates any products that already exist in WooCommerce, and it creates new products in WooCommerce for the rest.

Endpoints
At the beginning of the Product Export, the integration bulk loads various data from WooCommerce.

List All Products
Request method: GET

Endpoint: https://www.yourstore.com/wp-json/wc/v3/products

See sample response.

The Product Export will then update products that are already in both systems and create products that are not yet in WooCommerce.

Batch Update Products
Request method: POST

Endpoint: https://www.yourstore.com/wp-json/wc/v3/products/batch

See sample request and response.

Scripts

Product Item Master Matching Script - *A C# Script that can be used to match a GP item based on the WooCommerce product. This runs after an evaluation has been made with the Product Item Master Matching setting.*

Product Pre Export Script - *A C# Script that runs before a Product is exported.*

Product Post Export Script - *A C# script that runs after a product is successfully exported.*

Inventory Level Export

Overview

This automation component allows sending specific SalesPad/GP inventory quantities to the WooCommerce store. WooCommerce does not have a concept for multiple warehouses, so it is recommended to total up the quantities for each item across all applicable warehouses.

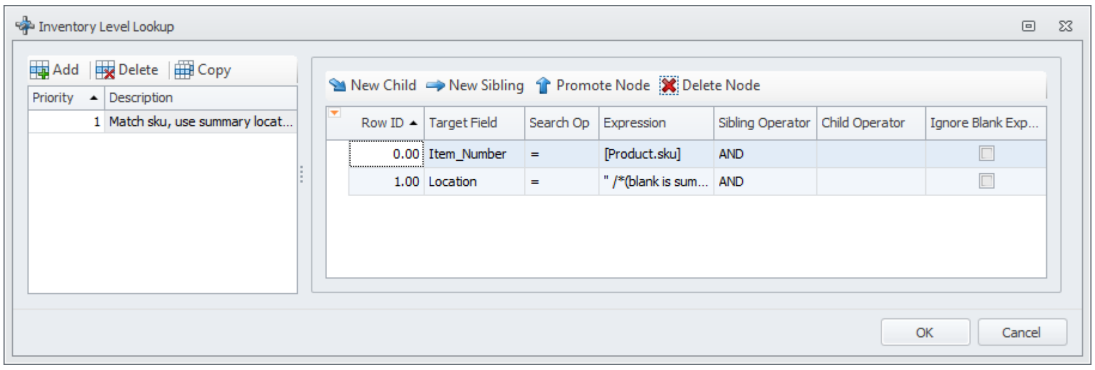
General Settings

Log WooCommerce Inventory Levels Not Updated - *The integration pulls all WooCommerce items, then attempts to find a GP item based on the Inventory Level Lookup setting. Set this value to true if you would like a log for each WooCommerce item that could not be matched. Defaults to false.*

Number Of Records Per Export Page - *Specify the number of records in each page of the Inventory Level Export. (Max: 100) Defaults to 10.*

Matching Settings

Inventory Level Lookup - *Define which GP Inventory Quantity Master record should be used to update the Product Inventory Levels. This should be used to match both Item and Location. If left blank, no inventory level quantity will be exported. Only one GP Site will be used per WooCommerce product, regardless if the filter returns multiple GP locations.*

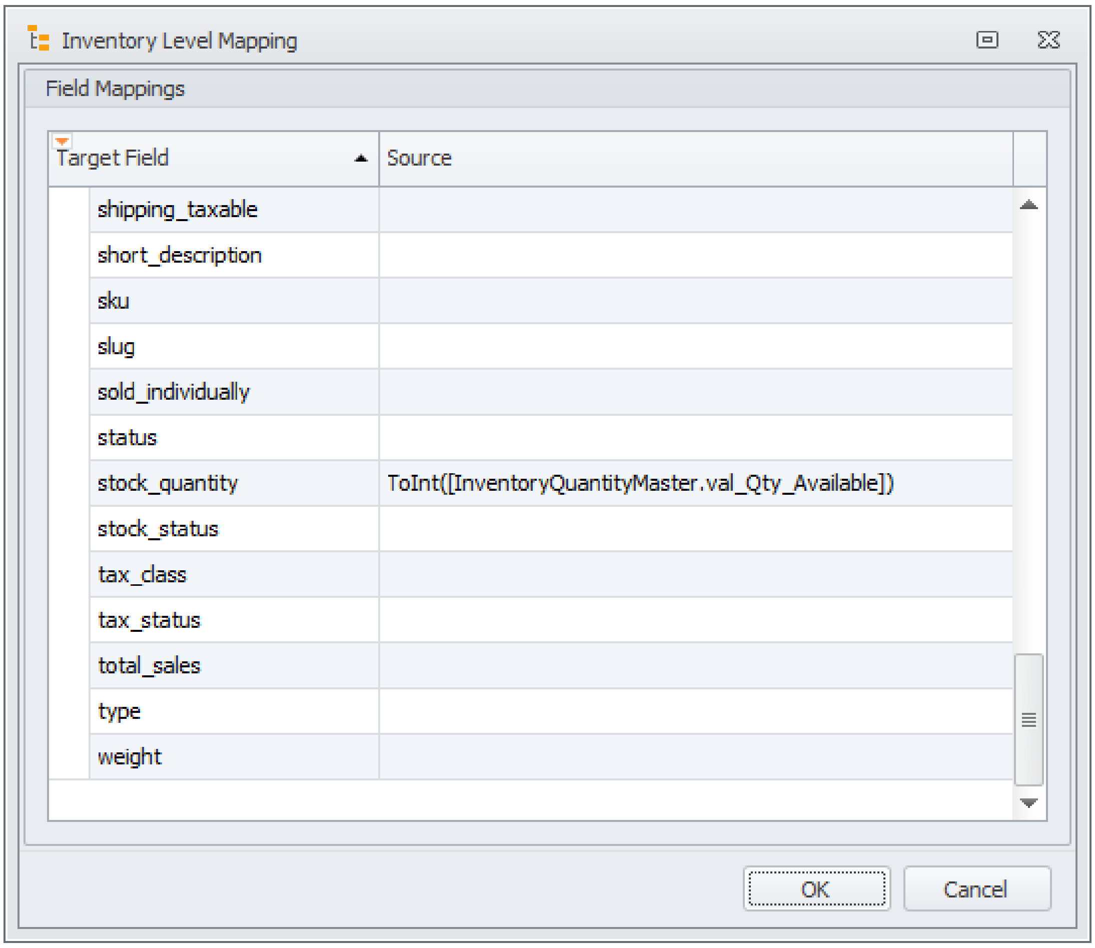


Priority	Description
1	Match sku, use summary locat...

Row ID	Target Field	Search Op	Expression	Sibling Operator	Child Operator	Ignore Blank Exp...
0.00	Item_Number	=	[Product.sku]	AND		☐
1.00	Location	=	" /*(blank is sum...	AND		☐

Assignment Settings

Inventory Level Mapping - *Assign the WooCommerce Stock quantity from the GP Inventory Location defined in Inventory Level Matching.*


Inventory Level Mapping

Field Mappings

Target Field	Source
shipping_taxable	
short_description	
sku	
slug	
sold_individually	
status	
stock_quantity	ToInt([InventoryQuantityMaster.val_Qty_Available])
stock_status	
tax_class	
tax_status	
total_sales	
type	
weight	

Processing

The Inventory Level Export bulk loads inventory levels and products from WooCommerce, and then it updates the inventory levels in WooCommerce.

Endpoints

At the beginning of the Inventory Level Export, the integration bulk loads data from WooCommerce.

List All Products

Request method: GET

Endpoint: https://www.yourstore.com/wp-json/wc/v3/products

See sample response.

The Inventory Level Export will then update inventory levels for products that are in both systems.

Batch Update Products

Request method: POST

Endpoint: https://www.yourstore.com/wp-json/wc/v3/products/batch

See sample request and response.

Scripts

Inventory Level Pre Export Script - *A C# Script that runs before an Inventory Level is exported.*

Inventory Level Post Export Script - *A C# Script that runs after a batch of Inventory Levels is successfully exported.*

Customer and Order Import

Overview

For each sales order on the website, the Order Import component will attempt to match a SalesPad/GP Customer to the WooCommerce Customer based on the *Customer Ship To Matching*, *Customer Bill To Matching*, and *Customer Matching settings*. If no match is found, a new customer will be created in SalesPad/GP based on the *Customer Assignment Matching setting*, and address codes for the new customer will be determined by the *Customer Ship To Assignment Mapping* and *Customer Bill To Assignment Mapping* settings.

Once the customer and addresses are matched or created, the sales order will be created based on the *Sales Document Assignment Mapping*, *Item Master Matching*, and *Sales Line Assignment Mapping* settings.

Payment information can be imported for sales orders as general or PayFabric payments. PayFabric transactions are created based on the *Sales Document PayFabric Mapping* setting.

General Settings

Enable Order Import Trace - *If enabled, customer and customer address matching information will be logged during order import. This setting should be enabled for troubleshooting purposes only. Defaults to False.*

Flag Comment For Imported Order - *Create a comment to send to a WooCommerce order when the document is imported. Defaults to “This order was imported into SalesPad.”*

Forward Document After Import - *If enabled, the imported order will be forwarded in workflow after being saved. Defaults to False.*

Import Orders In Status - *Select the Order Status that should be imported from WooCommerce. Defaults to “processing”.*

Multiple Potential Customers Scenario - Review Queue - *Queue that contains orders where a definitive customer match couldn't be found due to multiple possibilities being present. By default, the order will use the customer that has the earliest created date, then the order will be moved to the Workflow Queue designated by this setting to be reviewed.*

Named Notes Tab for WooCommerce Order Comments - *Specify which tab to import WooCommerce order comments to on the Sales Document. Defaults to “Internal Notes”.*

Number Of Days To Look Back - *Specify the number of days to look back from today to import orders. For example, set to 30 to import orders only from the last 30 days. Set to zero to import orders from any time. Defaults to 0.*

Number Of Orders Per Page - *Specify the number of orders to import at one time. Defaults to 50.*

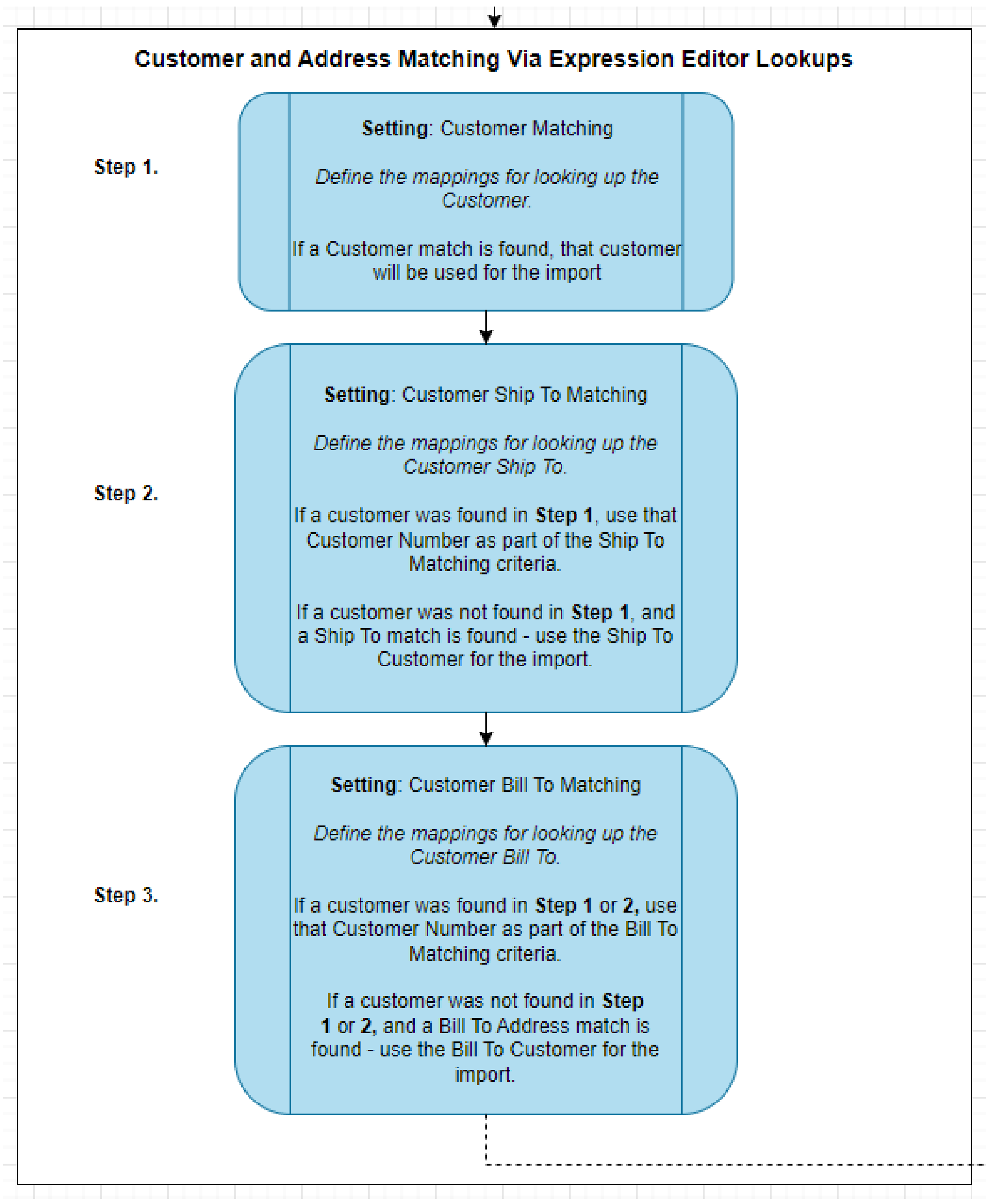
Roll Back Order Import Transaction On Error - *When enabled, the transaction encompassing the order import will be rolled back when an error occurs. This prevents data from a partially completed import from being saved to the database. Defaults to True.*

Update WooCommerce Order Status After Import - *After an order is imported into SalesPad, update the Order Status in WooCommerce. Defaults to “completed”.*

Matching Settings
These settings use Expression Editors that allow creating custom matching criteria to determine how a WooCommerce entity should match a SalesPad/GP entity.

Each Matching setting will build one or more SQL queries to attempt to find a match in the SalesPad/GP database. These queries are executed one at a time in order of Priority (lowest to highest). If an earlier priority query returns a result, that result will be returned to be used in the import, and the subsequent priorities will not execute.

The diagram below illustrates the sequence in which SalesPad attempts to match a SalesPad/GP Customer, Ship To Address, and Bill To Address each time a WooCommerce Order is imported.


Customer and Address Matching Via Expression Editor Lookups
Step 1.
Setting: Customer Matching
Define the mappings for looking up the Customer.
If a Customer match is found, that customer will be used for the import
Step 2.
Setting: Customer Ship To Matching
Define the mappings for looking up the Customer Ship To.
If a customer was found in **Step 1**, use that Customer Number as part of the Ship To Matching criteria.
If a customer was not found in **Step 1**, and a Ship To match is found - use the Ship To Customer for the import.
Step 3.
Setting: Customer Bill To Matching
Define the mappings for looking up the Customer Bill To.
If a customer was found in **Step 1** or **2**, use that Customer Number as part of the Bill To Matching criteria.
If a customer was not found in **Step 1** or **2**, and a Bill To Address match is found - use the Bill To Customer for the import.

Customer Matching - *Define the mappings for looking up the Customer.*

This setting is used as the first attempt to match a SalesPad/GP Customer to a WooCommerce Customer. If a customer is found in this step, that customer's Customer Number will be added to the search criteria when looking up the Ship To and Bill To Addresses.

Note that this setting does not need to be populated, since the customer can also be matched indirectly via Customer Ship To Matching or Customer Bill To Matching settings. If WooCommerce Orders should be matched by the shipping address instead, then the *Customer Matching* setting can be cleared, and the *Customer Ship To Matching* and/or *Customer Bill To Matching* settings will be used to load the Ship To / Bill To and the corresponding customer.

Customer Ship To Matching - *Define the mappings for looking up the Customer Ship To Address.*

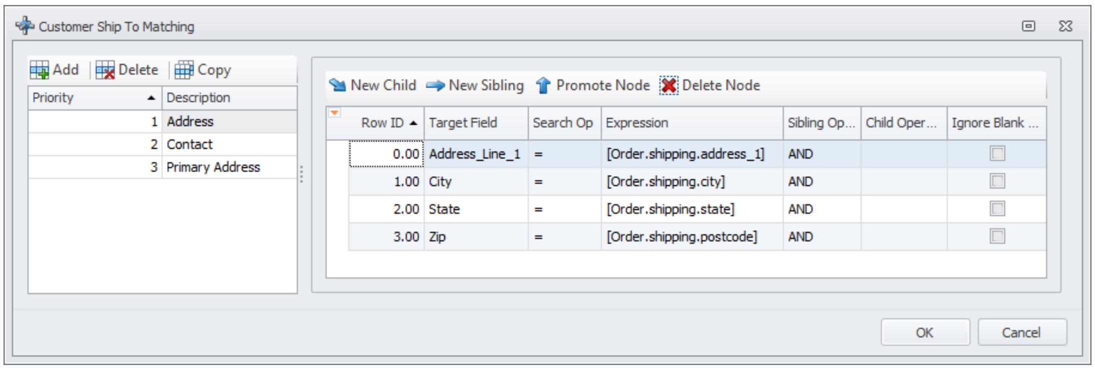
This setting determines how SalesPad will attempt to look up the Ship To Address for an incoming WooCommerce order. If a SalesPad/GP Customer was matched via the Customer Matching setting, the Ship To Address must also belong to that customer. Otherwise, the search will consider address codes across all customers.

For example, if the customer ‘AARONFIT0001’ has already been matched, only address codes having a Customer_Num of ‘AARONFIT0001’ will be considered, even if they otherwise match the criteria in the Ship To Address Matching setting. If a customer has not yet been matched when a Shipping Address is found, the order will be imported under the customer for that shipping address.

The default settings attempt to match a SalesPad/GP Shipping Address in three different steps.

Priority 1

Address: Attempt to find a match based on Address Line 1, City, State, and Zip. All four of these fields must be matched.

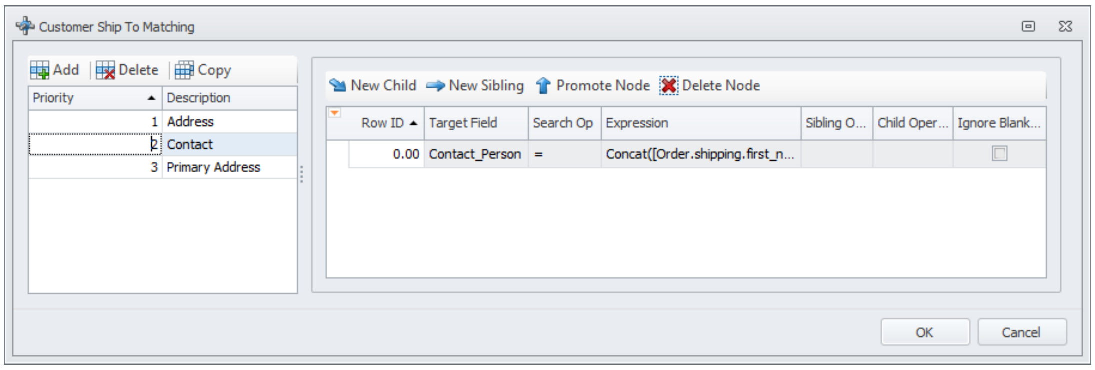


Priority	Description
1	Address
2	Contact
3	Primary Address

Row ID	Target Field	Search Op	Expression	Sibling Op...	Child Oper...	Ignore Blank ...
0.00	Address_Line_1	=	[Order.shipping.address_1]	AND		☐
1.00	City	=	[Order.shipping.city]	AND		☐
2.00	State	=	[Order.shipping.state]	AND		☐
3.00	Zip	=	[Order.shipping.postcode]	AND		☐

Priority 2

Contact: Attempt to find a match based on the First and Last Name of the WooCommerce Customer or WooCommerce Shipping Address.

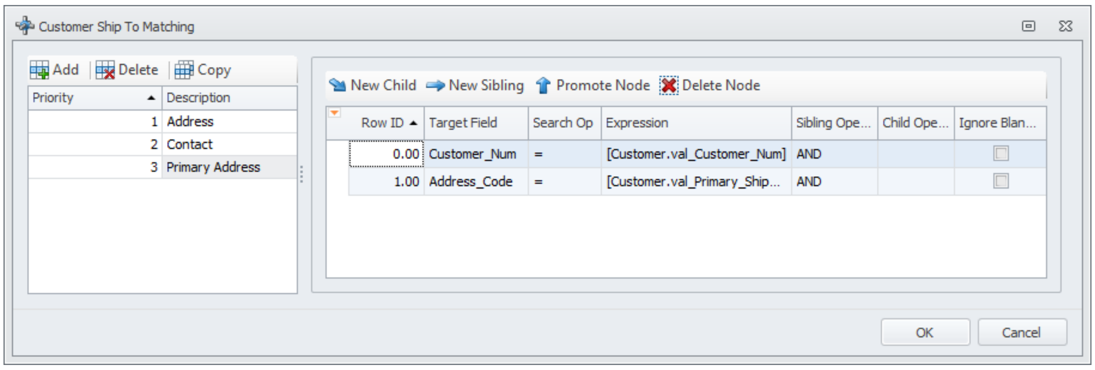


Priority	Description
1	Address
2	Contact
3	Primary Address

Row ID	Target Field	Search Op	Expression	Sibling O...	Child Oper...	Ignore Blank...
0.00	Contact_Person	=	Concat([Order.shipping.first_n...			☐

Priority 3

Primary Address: Attempt to find a match based on the Primary Address for the Customer. This assumes that the Customer was matched via the *Customer Matching* setting.


Customer Ship To Matching

Priority	Description
1	Address
2	Contact
3	Primary Address

Row ID	Target Field	Search Op	Expression	Sibling Ope...	Child Ope...	Ignore Blan...
0.00	Customer_Num	=	[Customer.val_Customer_Num]	AND		☐
1.00	Address_Code	=	[Customer.val_Primary_Ship...	AND		☐

Customer Bill To Matching - *Define the mappings for looking up the Customer Bill To Address.*

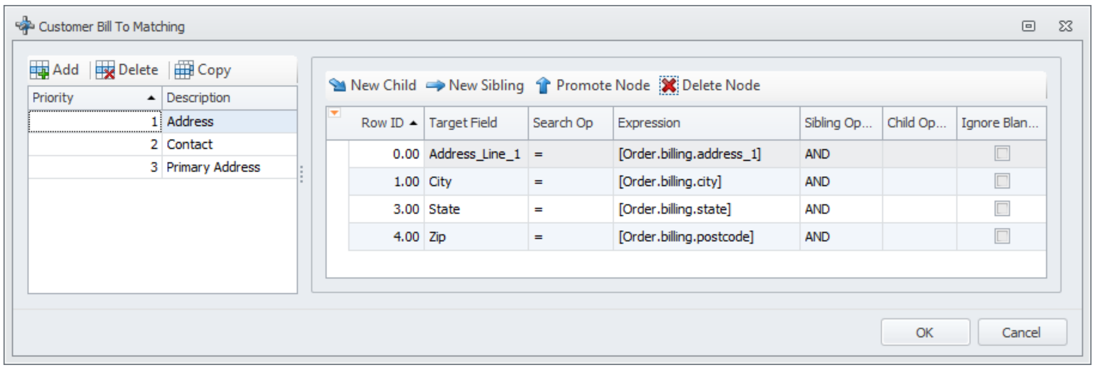
This setting determines how SalesPad Desktop will attempt to look up the Bill To Address for an incoming WooCommerce order. If a SalesPad/GP Customer was matched via the Customer Matching setting, the Bill To Address must also belong to that customer. Otherwise, the search will consider address codes across all customers.

For example, if the customer ‘AARONFIT0001’ has already been matched, only address codes having a Customer_Num of ‘AARONFIT0001’ will be considered, even if they otherwise match the criteria in the Bill To Address Matching setting. If a customer has not yet been matched when a Bill To Address is found, the order will be imported under the customer for that shipping address.

The default settings attempt to match a SalesPad/GP Billing Address in three different steps.

Priority 1

Address: Attempt to find a match based on Address Line 1, City, State, and Zip. All four of these fields must be matched.


Customer Bill To Matching

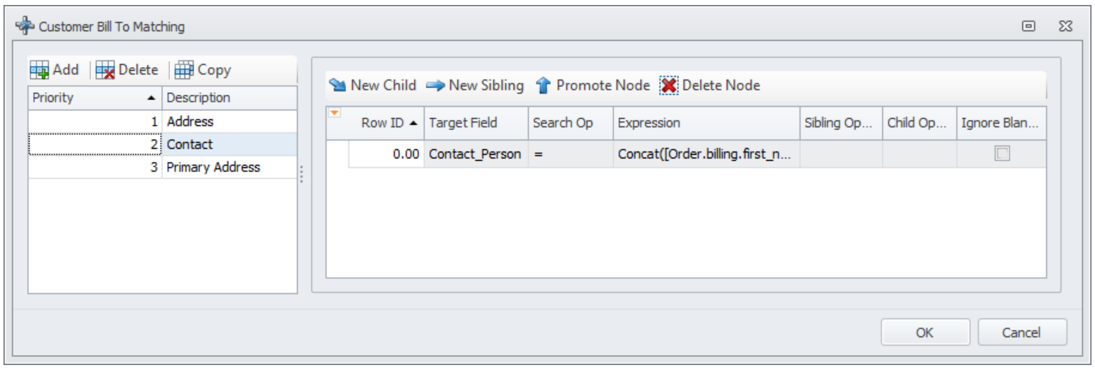
Priority	Description
1	Address
2	Contact
3	Primary Address

Row ID	Target Field	Search Op	Expression	Sibling Op...	Child Op...	Ignore Blan...
0.00	Address_Line_1	=	[Order.billing.address_1]	AND		☐
1.00	City	=	[Order.billing.city]	AND		☐
3.00	State	=	[Order.billing.state]	AND		☐
4.00	Zip	=	[Order.billing.postcode]	AND		☐

Priority 2

Contact Person: Attempt to find a match based on the First and Last Name of the

WooCommerce Customer or WooCommerce Shipping Address.

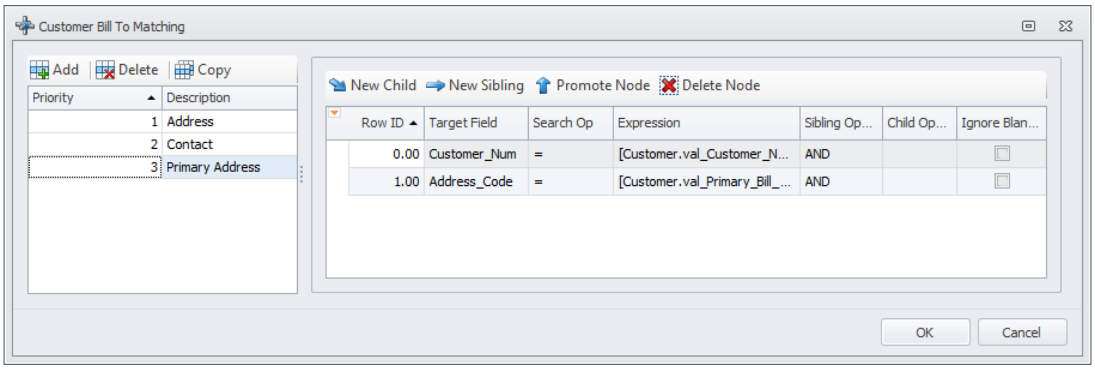


Priority	Description
1	Address
2	Contact
3	Primary Address

Row ID	Target Field	Search Op	Expression	Sibling Op...	Child Op...	Ignore Blan...
0.00	Contact_Person	=	Concat([Order.billing.first_n...			☐

Priority 3

Primary Address: Attempt to find a match based on the Primary Address for the Customer. This assumes that the Customer was matched via the *Customer Matching* setting.



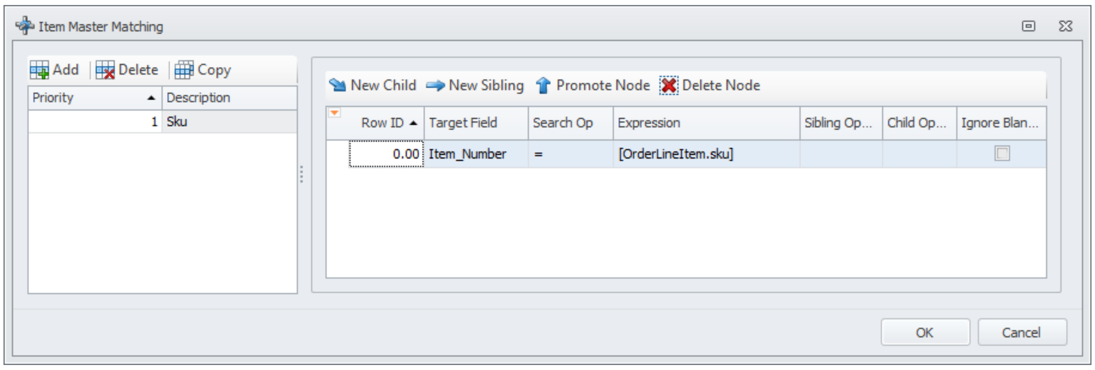
Priority	Description
1	Address
2	Contact
3	Primary Address

Row ID	Target Field	Search Op	Expression	Sibling Op...	Child Op...	Ignore Blan...
0.00	Customer_Num	=	[Customer.val_Customer_N...	AND		☐
1.00	Address_Code	=	[Customer.val_Primary_Bill_...	AND		☐

Item Master Matching - *Define the mappings for matching WooCommerce items to a GP Item Master.*

Each WooCommerce Line Item will use this setting to find the corresponding SalesPad/GP Item.

The default settings attempt to match a WooCommerce SKU directly to a SalesPad/GP Item Number.



Priority	Description
1	Sku

Row ID	Target Field	Search Op	Expression	Sibling Op...	Child Op...	Ignore Blan...
0.00	Item_Number	=	[OrderLineItem.sku]			☐

Assignment Settings

Assignment settings use Expression Editors to define how newly created Customers, Orders, and Sales Lines will be populated when created during the order import process.

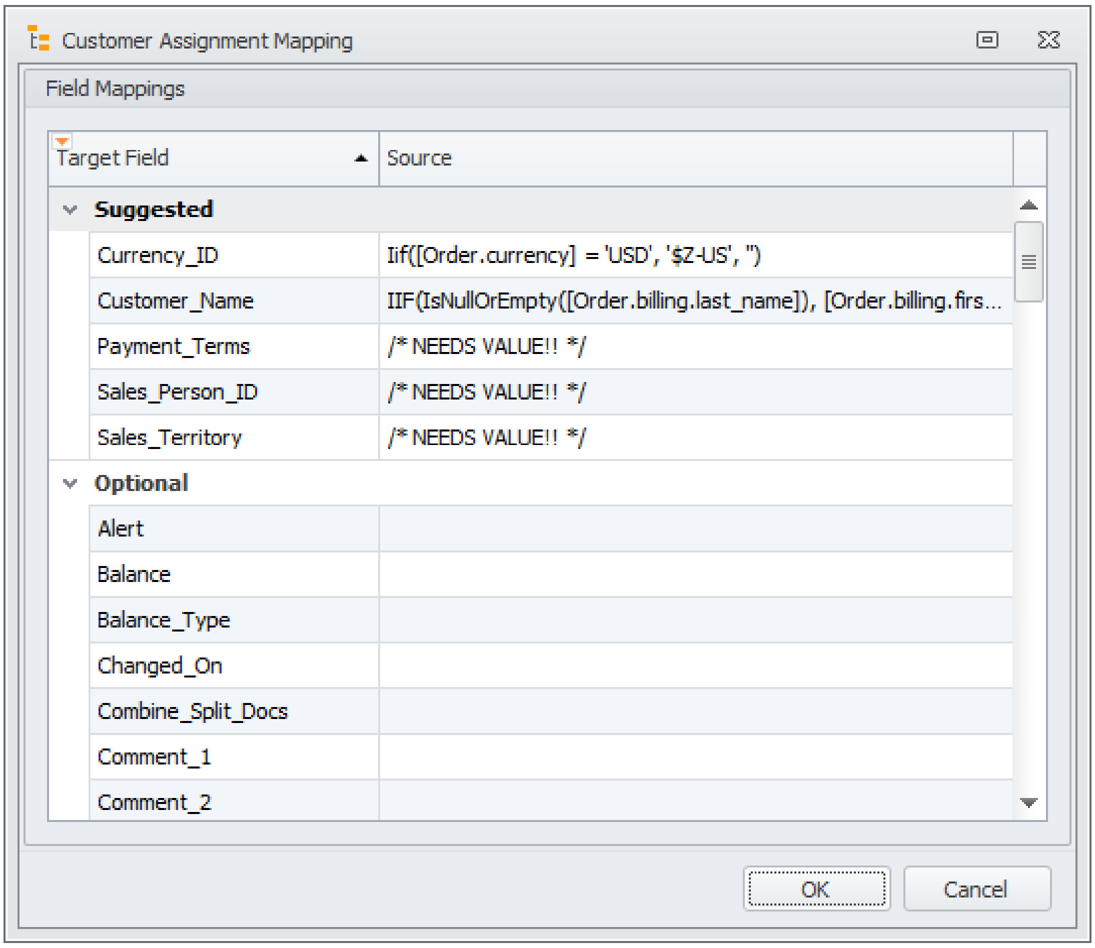
The target objects will be SalesPad/GP Customers, Contacts, Sales Documents, and Sales Line Items. The source objects will be WooCommerce Orders and Order Line Items.

Complete WooCommerce API documentation can be found here.

Each Mapping contains a grid of all the target fields and an expression that will be used to populate each field. Required fields are grouped under the Suggested category. If a Suggested category field is not populated, an error may occur.

Customer Assignment Mapping - *Define the mappings to be used when creating a new Customer.*

If an existing SalesPad/GP Customer was not found using the matching settings, then a new Customer will be created using the values in the *Customer Assignment Mapping* setting.

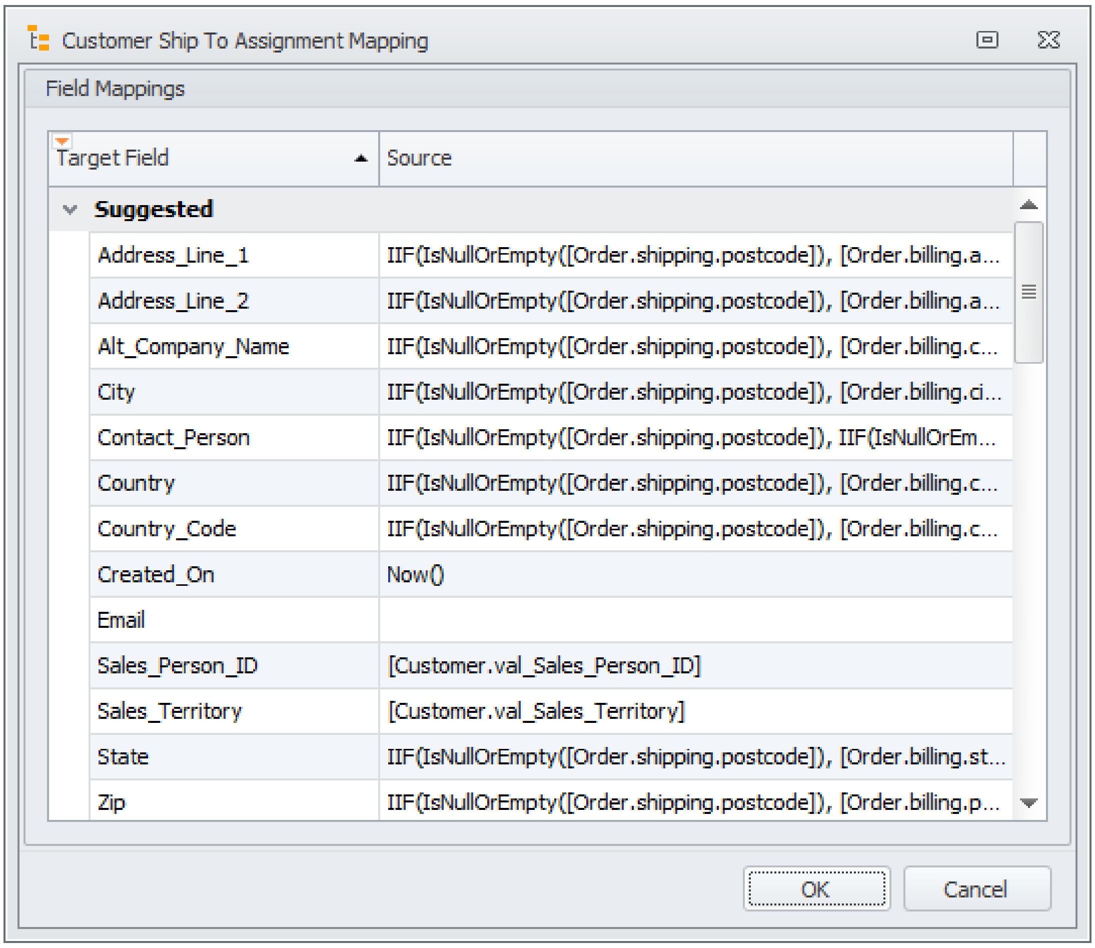


Target Field	Source
Suggested	
Currency_ID	Iif([Order.currency] = 'USD', '$Z-US', '')
Customer_Name	IIF(IsNullOrEmpty([Order.billing.last_name]), [Order.billing.firs...
Payment_Terms	/* NEEDS VALUE!! */
Sales_Person_ID	/* NEEDS VALUE!! */
Sales_Territory	/* NEEDS VALUE!! */
Optional	
Alert	
Balance	
Balance_Type	
Changed_On	
Combine_Split_Docs	
Comment_1	
Comment_2	

Customer Ship To Assignment Mapping - *Define the mappings to be used when creating a new Customer Ship To Address.*

If an existing SalesPad/GP Customer Address was not found using the *Customer Ship To*

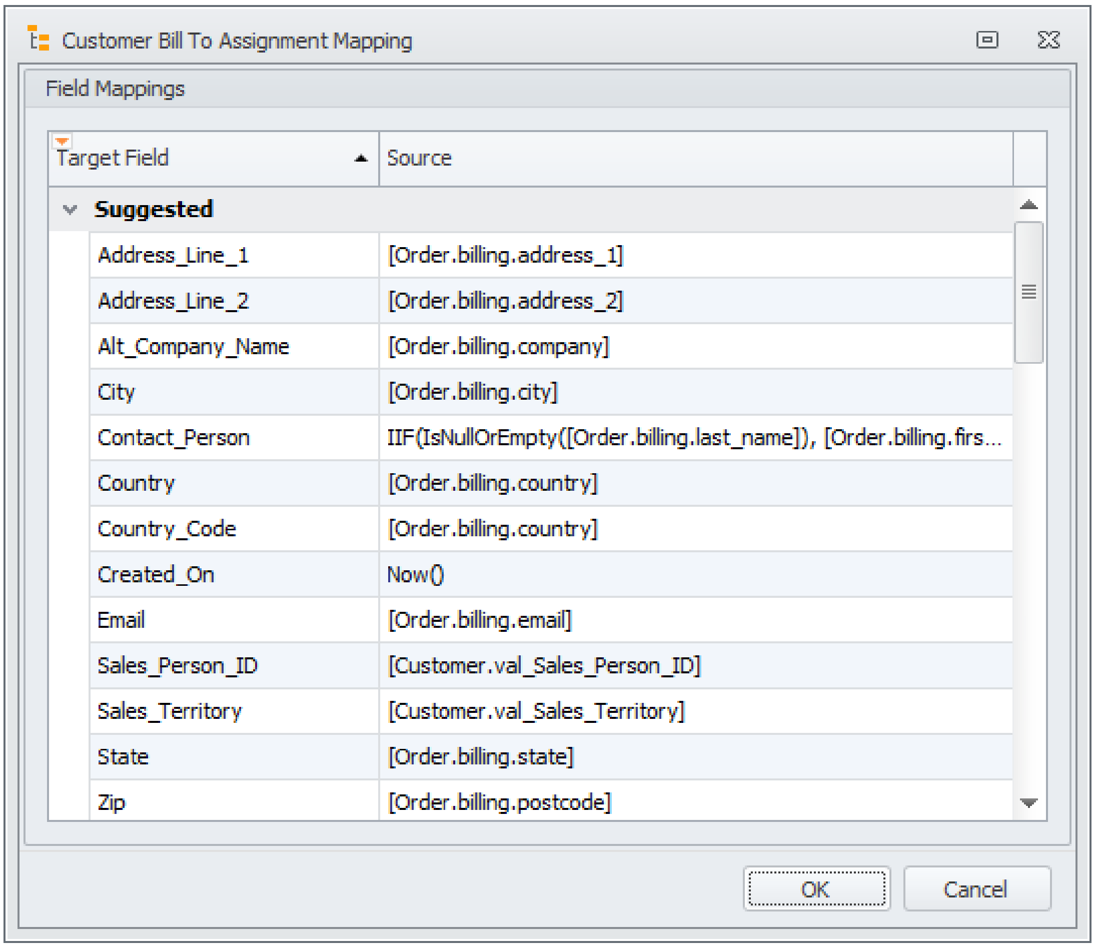
Matching setting, then a new Customer Address will be created using the values in the *Customer Ship To Assignment Mapping* setting.



Target Field	Source
Suggested	
Address_Line_1	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.a...
Address_Line_2	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.a...
Alt_Company_Name	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.c...
City	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.ci...
Contact_Person	IIF(IsNullOrEmpty([Order.shipping.postcode]), IIF(IsNullOrEm...
Country	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.c...
Country_Code	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.c...
Created_On	Now()
Email	
Sales_Person_ID	[Customer.val_Sales_Person_ID]
Sales_Territory	[Customer.val_Sales_Territory]
State	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.st...
Zip	IIF(IsNullOrEmpty([Order.shipping.postcode]), [Order.billing.p...

Customer Bill To Assignment Mapping - *Define the mappings to be used when creating a new Customer Bill To Address.*

If an existing SalesPad/GP Customer Address was not found using the *Customer Bill To Matching* setting, then a new Customer Address will be created using the values in the *Customer Bill To Assignment Mapping* setting.

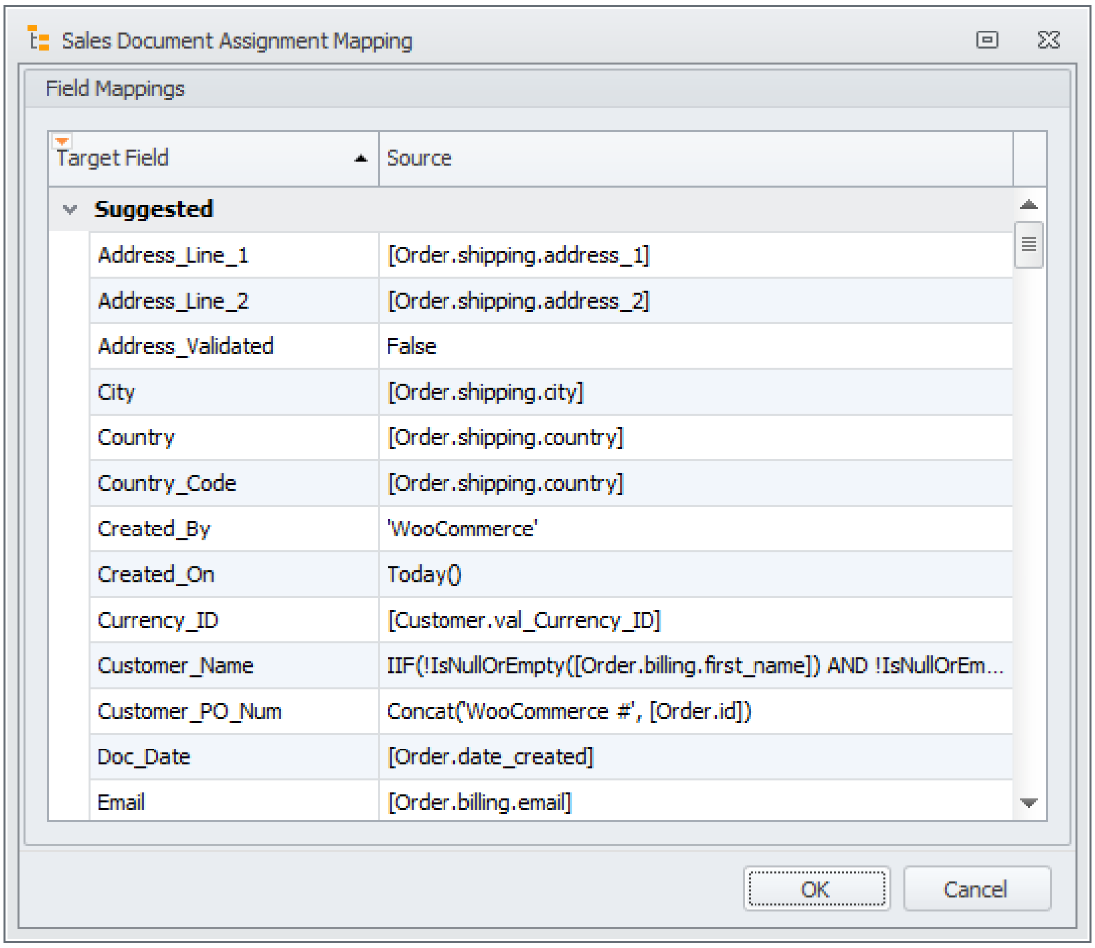

Customer Bill To Assignment Mapping

Field Mappings

Target Field	Source
Suggested	
Address_Line_1	[Order.billing.address_1]
Address_Line_2	[Order.billing.address_2]
Alt_Company_Name	[Order.billing.company]
City	[Order.billing.city]
Contact_Person	IIF(IsNullOrEmpty([Order.billing.last_name]), [Order.billing.firs...
Country	[Order.billing.country]
Country_Code	[Order.billing.country]
Created_On	Now()
Email	[Order.billing.email]
Sales_Person_ID	[Customer.val_Sales_Person_ID]
Sales_Territory	[Customer.val_Sales_Territory]
State	[Order.billing.state]
Zip	[Order.billing.postcode]

Sales Document Assignment Mapping - *Define the mappings to be used when creating a new Sales Document.*

When the WooCommerce Order is imported, this setting will be used to populate the header level fields on the SalesPad/GP Sales Document.

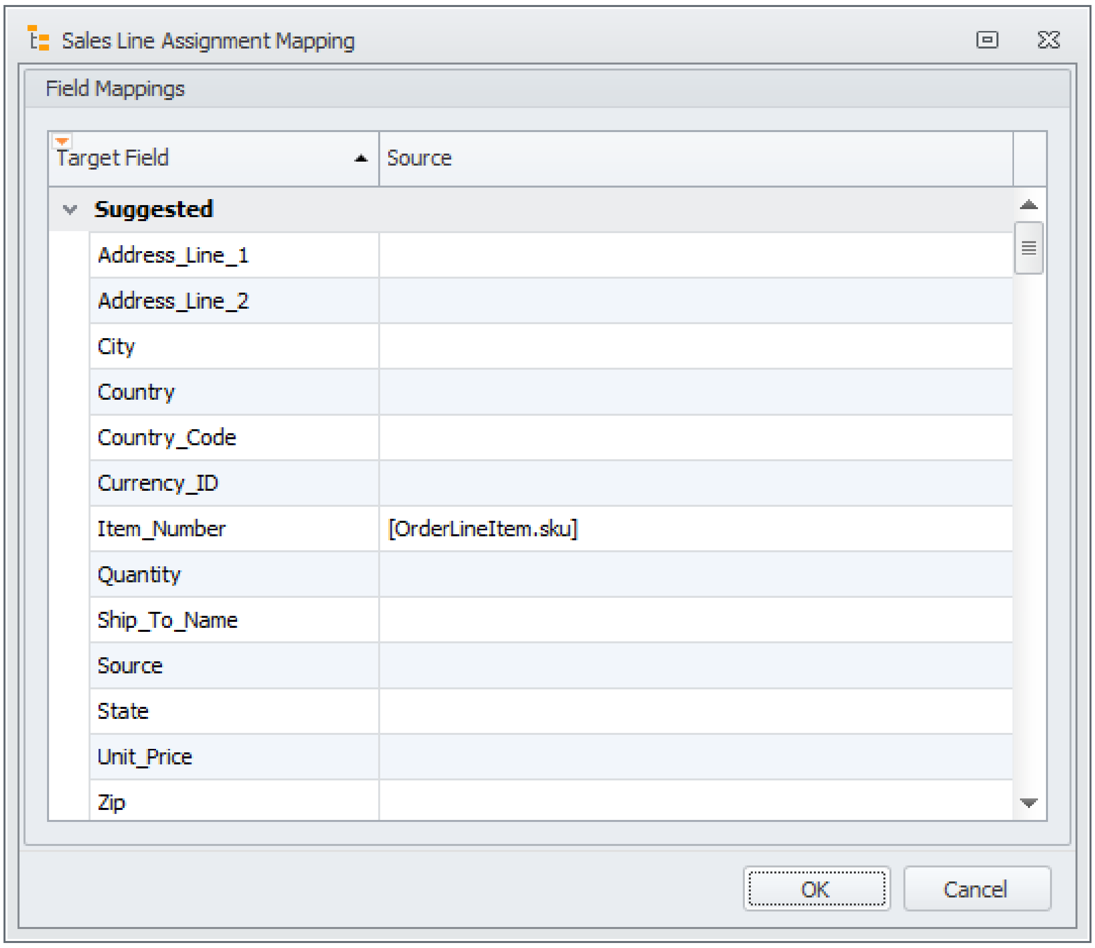

Sales Document Assignment Mapping

Field Mappings

Target Field	Source
Suggested	
Address_Line_1	[Order.shipping.address_1]
Address_Line_2	[Order.shipping.address_2]
Address_Validated	False
City	[Order.shipping.city]
Country	[Order.shipping.country]
Country_Code	[Order.shipping.country]
Created_By	'WooCommerce'
Created_On	Today()
Currency_ID	[Customer.val_Currency_ID]
Customer_Name	IIF(!IsNullOrEmpty([Order.billing.first_name]) AND !IsNullOrEm...
Customer_PO_Num	Concat('WooCommerce #', [Order.id])
Doc_Date	[Order.date_created]
Email	[Order.billing.email]

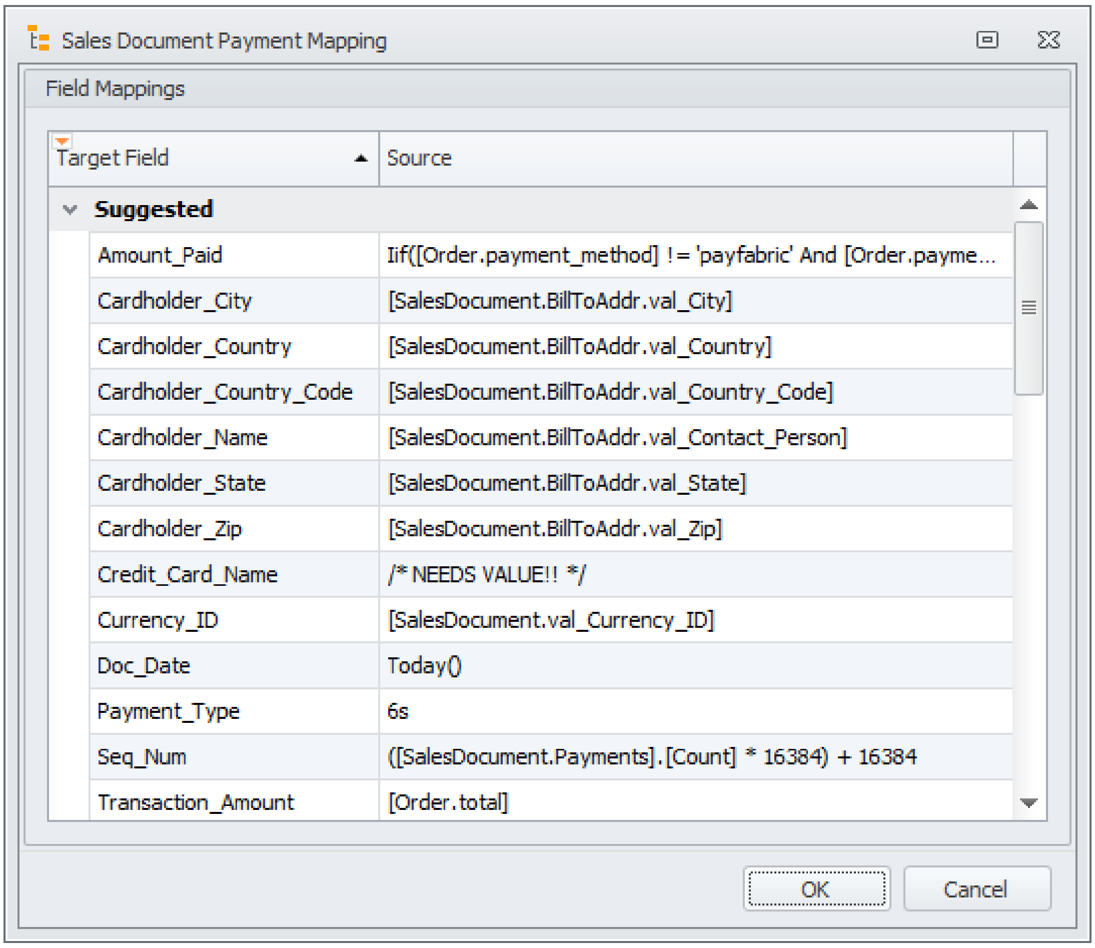
Sales Line Assignment Mapping - *Define the mappings to be used when creating a new Sales Line Item.*

When the WooCommerce Order is imported, each line on that order will be created as a sales line on the SalesPad/GP Sales Document. Each sales line's fields are set based on the *Sales Line Assignment Mapping* setting.



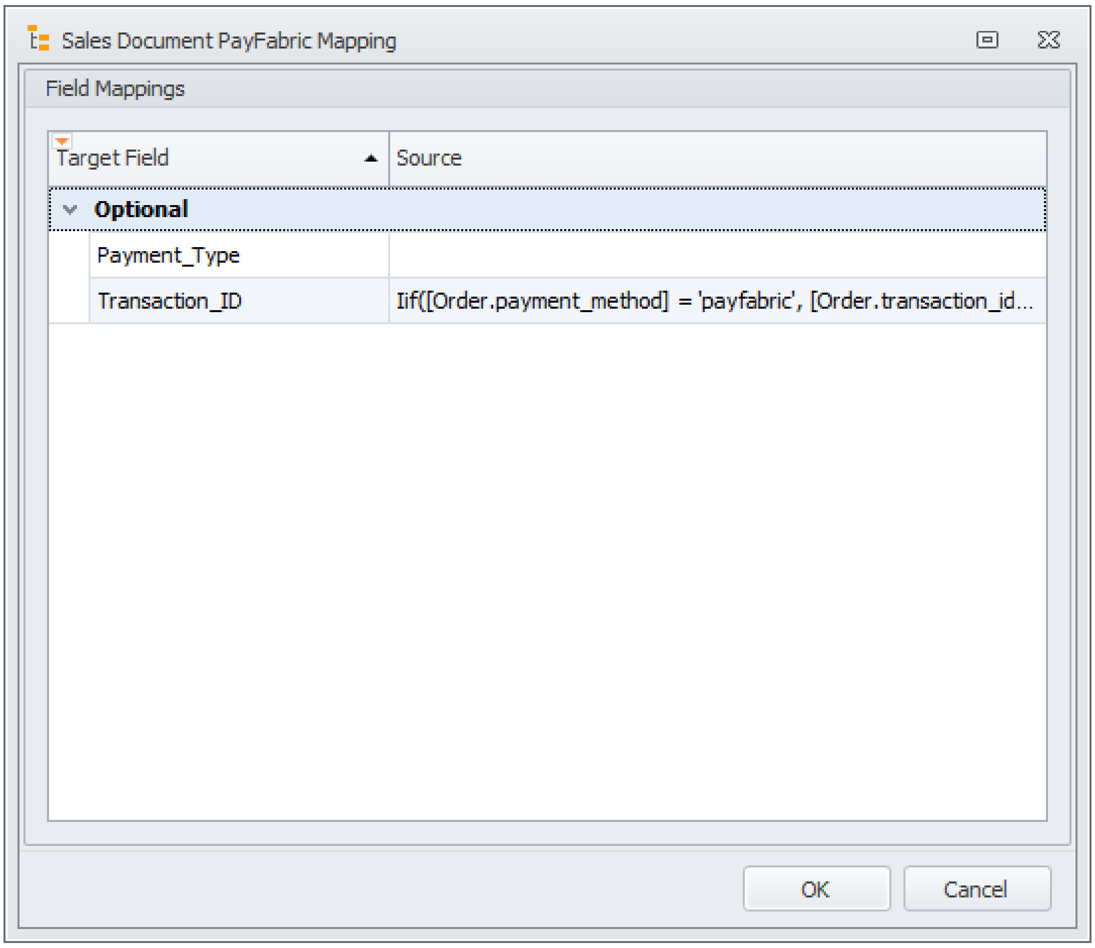
Target Field	Source
Suggested	
Address_Line_1	
Address_Line_2	
City	
Country	
Country_Code	
Currency_ID	
Item_Number	[OrderLineItem.sku]
Quantity	
Ship_To_Name	
Source	
State	
Unit_Price	
Zip	

Sales Document Payment Mapping - *Define the mappings to be used when creating a Sales Document Payment.*



Target Field	Source
Suggested	
Amount_Paid	Iif([Order.payment_method] != 'payfabric' And [Order.payme…
Cardholder_City	[SalesDocument.BillToAddr.val_City]
Cardholder_Country	[SalesDocument.BillToAddr.val_Country]
Cardholder_Country_Code	[SalesDocument.BillToAddr.val_Country_Code]
Cardholder_Name	[SalesDocument.BillToAddr.val_Contact_Person]
Cardholder_State	[SalesDocument.BillToAddr.val_State]
Cardholder_Zip	[SalesDocument.BillToAddr.val_Zip]
Credit_Card_Name	/* NEEDS VALUE!! */
Currency_ID	[SalesDocument.val_Currency_ID]
Doc_Date	Today()
Payment_Type	6s
Seq_Num	([SalesDocument.Payments].[Count] * 16384) + 16384
Transaction_Amount	[Order.total]

Sales Document PayFabric Mapping - *Define the mappings to be used when creating a PayFabric transaction.*



Target Field	Source
Optional	
Payment_Type	
Transaction_ID	Iif([Order.payment_method] = 'payfabric', [Order.transaction_id...

Processing
The Order Import retrieves all orders in WooCommerce that are ready to be imported, and then it creates customers and addresses as needed before creating the sales orders. It adds a comment and updates the status of the order in WooCommerce to indicate that importing was completed.

Endpoints
List All Orders
Request method: GET

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders

See sample response.

Update Order Status
Request method: PUT

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders/<id>/<status>

See sample request and response.

Create Order Note
Request method: POST

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders/<id>/notes

See sample request and response.

Scripts
Sales Document Pre Import Script - *A C# Script that runs before a Sales Document is imported.*

Parameters: System.ComponentModel.CancelEventArgs ce, Object sourceDoc, SalesPad.Bus.SalesDocument sd

When importing a WooCommerce Order, this script runs before any assignments have been executed. This script can be used to cancel the import early by setting ce.Cancel = true. Note that Sales Document fields populated by this script may be overridden by assignment settings.

New Customer Creation Script - *A C# Script that runs after a new customer is created and before it is saved.*

Parameters: System.ComponentModel.CancelEventArgs ce, Object sourceDoc, SalesPad.Bus.SalesDocument sd, SalesPad.Bus.Customer customer

This script will run before a new customer is saved for the first time. The script runs after the assignments from the *Customer Assignment Mapping*, but before the assignments from the *Customer Ship To Assignment Mapping* and the *Customer Bill To Assignment Mapping*.

The script can be used to do more complex assignments, and it can cancel the import by setting ce.Cancel = true.

Customer And Address Matching Script - *A C# Script that runs after the customer and addresses have been matched, and can be used to load a different customer, ship to address, or bill to address.*

Parameters: System.ComponentModel.CancelEventArgs ce, Object sourceDoc, SalesPad.Bus.Customer customer, SalesPad.Bus.CustomerAddr shipToAddr, SalesPad.Bus.CustomerAddr billToAddr, bool multipleCustomersFound

This script will run after the customer and addresses have been matched. It can be used to do more complex assignments or additional validation, and it can cancel the import by setting ce.Cancel = true.

Sales Document Pre Save - *A C# Script that runs just before a Sales Document is saved.*

Parameters: System.ComponentModel.CancelEventArgs ce, Object sourceDoc, SalesPad.Bus.SalesDocument sd

After this script runs, the new sales document is saved. This script is the last opportunity to adjust values, or cancel the import by setting ce.Cancel = true.

Sales Document Payment Script - *A C# Script that can be used to override the default payment mappings. (Setting: Sales Document Payment Mapping)*

Parameters: System.ComponentModel.CancelEventArgs ce, Object sourceDoc, SalesPad.Bus.SalesDocument sd

This script can override the default payment mappings, and it can cancel the import by setting ce.Cancel = true.

Item Master Matching Script - *A C# Script that can be used to load SalesPad.Bus.ItemMaster item.*

Parameters: System.ComponentModel.CancelEventArgs ce, Object sourceDoc, Object sourceLine, SalesPad.Bus.SalesDocument sd, SalesPad.Bus.ItemMaster item

This script can override the default matched Item Master, and it can cancel the import by setting ce.Cancel = true.

Customer and Customer Address Tracing
Depending on the complexity of the matching configuration, it may be difficult to tell how each customer and customer address is getting matched during order import. Order Import Tracing functionality will log every possible matching result to the spAAIntegrationTrace table in the database, regardless if the matching was successful or not. Tracing may be enabled by enabling the *Enable Order Import Trace* setting under Order Import. Note that enabling this setting may cause database bloat, so it is intended only for troubleshooting purposes.

The following quick report can be used to query the spAAIntegrationTrace table in SalesPad:

```
<report name="AA Integration Trace" AutoLinks="true" GroupFooterShowMode="Expanded" bestFitAll="true" AutoFit="false">

  <description />

  <query addWhere="true">SELECT *

FROM (

SELECT Automation_Name = ai.Instance_Name

,Automation_Description = ai.Instance_Description

,Component_Name = aic.Component_Name

,Trace_Name = ait.Trace_Name

,Group_ID = ait.Group_ID

,External_Object_Key = ait.External_Object_Key

,External_Object = ait.External_Object

,Mapping_Info = ait.Mapping_Info

,Business_Object_Name = ait.Business_Object_Name
,Search_Clause = ait.Search_Clause

,Results = ait.Results

,Created_On = ait.Created_On

,Created_By = ait.Created_By

FROM spAAIntegrationTrace AS ait WITH (NOLOCK)

LEFT JOIN spAAInstance AS ai WITH (NOLOCK) ON ai.AA_Instance_ID = ait.AA_Instance_ID

LEFT JOIN spAAInstanceComponent AS aic WITH (NOLOCK) ON aic.AA_Instance_ID =
ait.AA_Instance_ID

AND aic.Component_ID = ait.AA_Component_ID

) AS a</query>

  <search name="Automation Name" column="Automation_Name" searchOp="LIKE" Type="Text"
/>

  <search name="Trace Name" column="Trace_Name" searchOp="LIKE" Type="Text" />

  <search name="External Object Key" column="External_Object_Key" searchOp="LIKE"
Type="Text" />

  <search name="Business Object Name" column="Business_Object_Name" searchOp="LIKE"
Type="Text" />

  <search name="Created On" column="Created_On" searchOp="=" Type="DateTime" />

  <OnRunScript />

</report>
```

The embedded SQL query can be used directly from SSMS.

Order Update Export

Overview

The Order Update Export component is used to export tracking numbers and notes back to WooCommerce for orders that were originally created by the Order Import component.

This component will target a designated workflow queue. Each document in that queue will use the link that was created during the order import to call the WooCommerce API and update the appropriate document. For Sales Documents, the External ID designates the linked WooCommerce Order ID. The Sales Line Item also includes an External ID column that stores the ID of the corresponding WooCommerce line item.

General Settings

Export Queue - *Queue that contains orders ready to be exported to WooCommerce. This component will ignore orders which do not have an External ID, but will assume that any orders which have an External ID belong to the linked WooCommerce website even if the* *order originated from a different automation instance.*

Orders that were imported from WooCommerce will need to be directed into this queue so that they can export tracking information back to WooCommerce. They will wait to be processed, and afterward, they will be forwarded in the workflow.

Export Failure Queue - *In the event of an unsuccessful export, the document will be placed into this queue.*

If there is an exception during the order export process, the document will be moved to this queue and the error will be logged to the Automation Agent Action Center.

Number Of Orders Per Export Page - *Specify the number of orders in each page of the order export. Defaults to 50.*

Use this setting to minimize the number of orders that SalesPad attempts to load and process at the same time.

Roll Back Order Update Export Transaction On Error - *When enabled, the transaction encompassing the Order Update Export will be rolled back when an error occurs. This prevents data from a partially completed export from being saved to the database. Defaults to True.*

Tracking Update Notify - *Notify customer of tracking update. Defaults to True.*

Processing
The Order Update Export gets each order from WooCommerce, and then it adds a note with the item and tracking information.

Order note example:

The following items have shipped via tracking numbers 0123456789, 5555555555, 9898989898, 3336669999, 4444488888:

Item Number 'A100' (Qty: 1)

Item Number 'CAP100' (Qty: 5)

Item Number '100XLG' (Qty: 3)

Item Number 'HD-40' (Qty: 1)

Endpoints

List All Orders

Request method: GET

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders

See sample response.

Create Order Note

Request method: POST

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders/<id>/notes

See sample request and response.

Scripts

Order Update Export Pre Submission Script - *A C# Script that will execute prior to sending tracking info to WooCommerce.*

Order Voided Export

Overview

The Order Voided Export component is used to cancel WooCommerce orders.

General Settings

Cancellation Notify - *Notify Customer of order cancellation. Defaults to True.*

Number Of Days To Look Back - *Specify the number of days to look back from today to export voided orders. For example, set to 30 to export voided orders only from the last 30 days. Set to zero to export voided orders from any time. Defaults to 30.*

Number Of Voided Orders Per Export Page - *Specify the number of voided orders in each page of the Order Voided Export. Defaults to 50.*

Processing

The Order Voided Export component loads sales orders from SalesPad/GP and then voids corresponding orders in WooCommerce by updating the order status and note.

Order status is set to cancelled:

```
order.status = WooCommerce.Utils.OrderStatus.cancelled;
```

Order note example:

```
This order was voided in SalesPad.
```

Endpoints

Retrieve Order

Request method: GET

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders/<id>

See sample response.

Update Order

Request method: PUT

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders/<id>

See sample request and response.

Create Order Note

Request method: POST

Endpoint: https://www.yourstore.com/wp-json/wc/v3/orders/<id>/notes

See sample request and response.