CAVALLO

**SalesPad**

# Shopify Integration

## Overview

Cavallo's integration with Shopify handles automatic syncing of inventory and sales information between SalesPad/GP and a Shopify website. Product and inventory level information is pushed from SalesPad 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 so that they can be processed and fulfilled. Payment information can also be imported from the website for visibility in SalesPad. After fulfillment and tracking information is updated for each sales order in SalesPad, this information is pushed to the website for customer visibility. If sales orders are voided in SalesPad or GP, those updates are communicated back to the website.

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

## General Information On Expressions

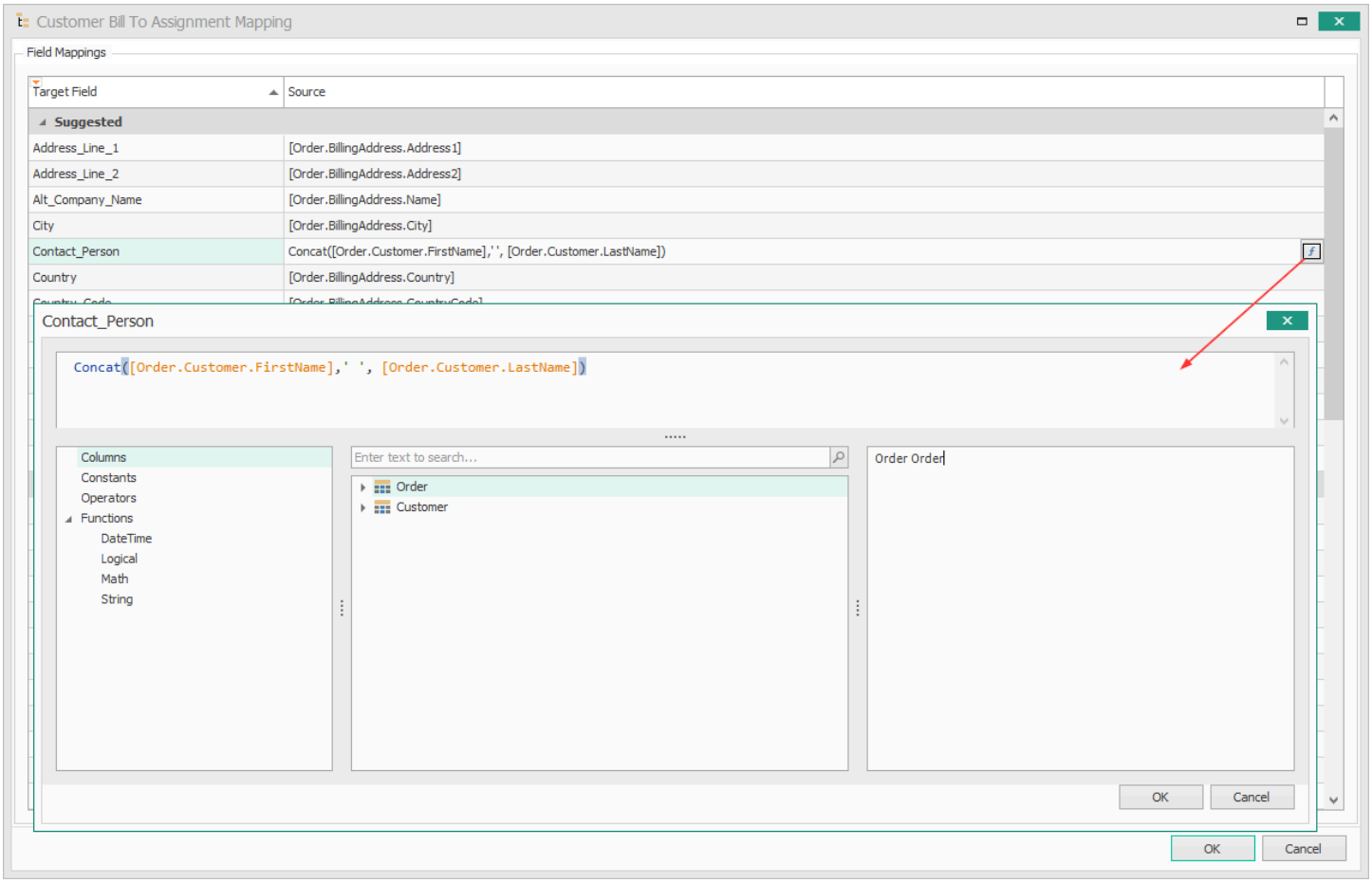
### Overview

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.

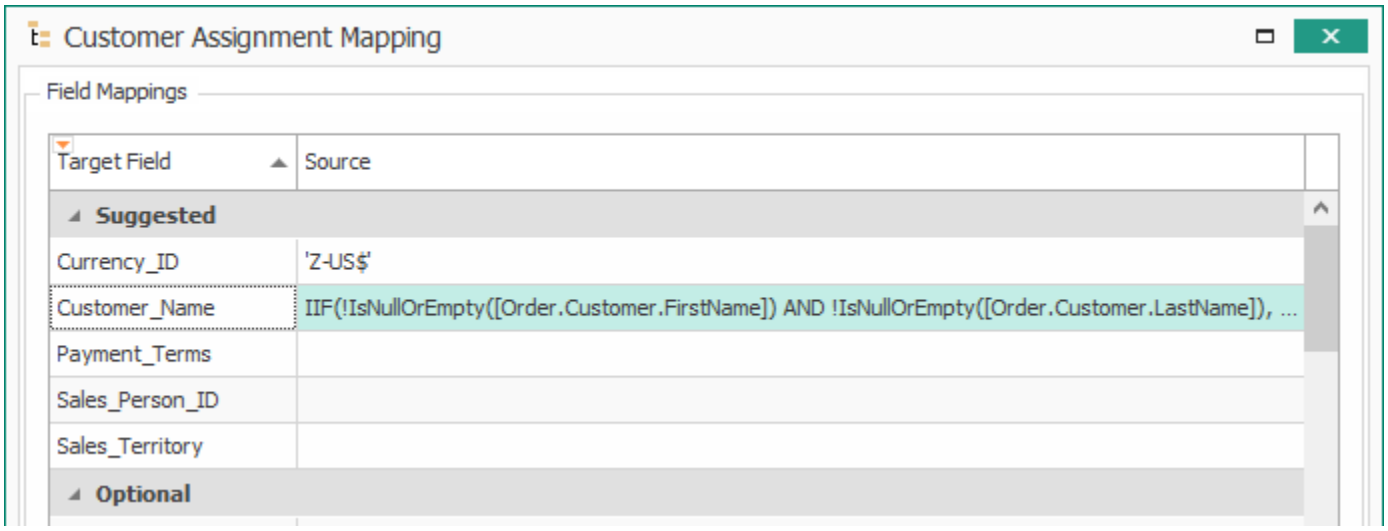
As an example, below is the default expression for *Customer Bill To Assignment Mapping - Contact_Person.* In this case, the Shopify order is the source object, and a SalesPad CustomerAddr is

the destination object. When a new SalesPad CustomerAddr is created in the process of creating a customer for an imported order, this expression is used to populate the Contact_Person field.

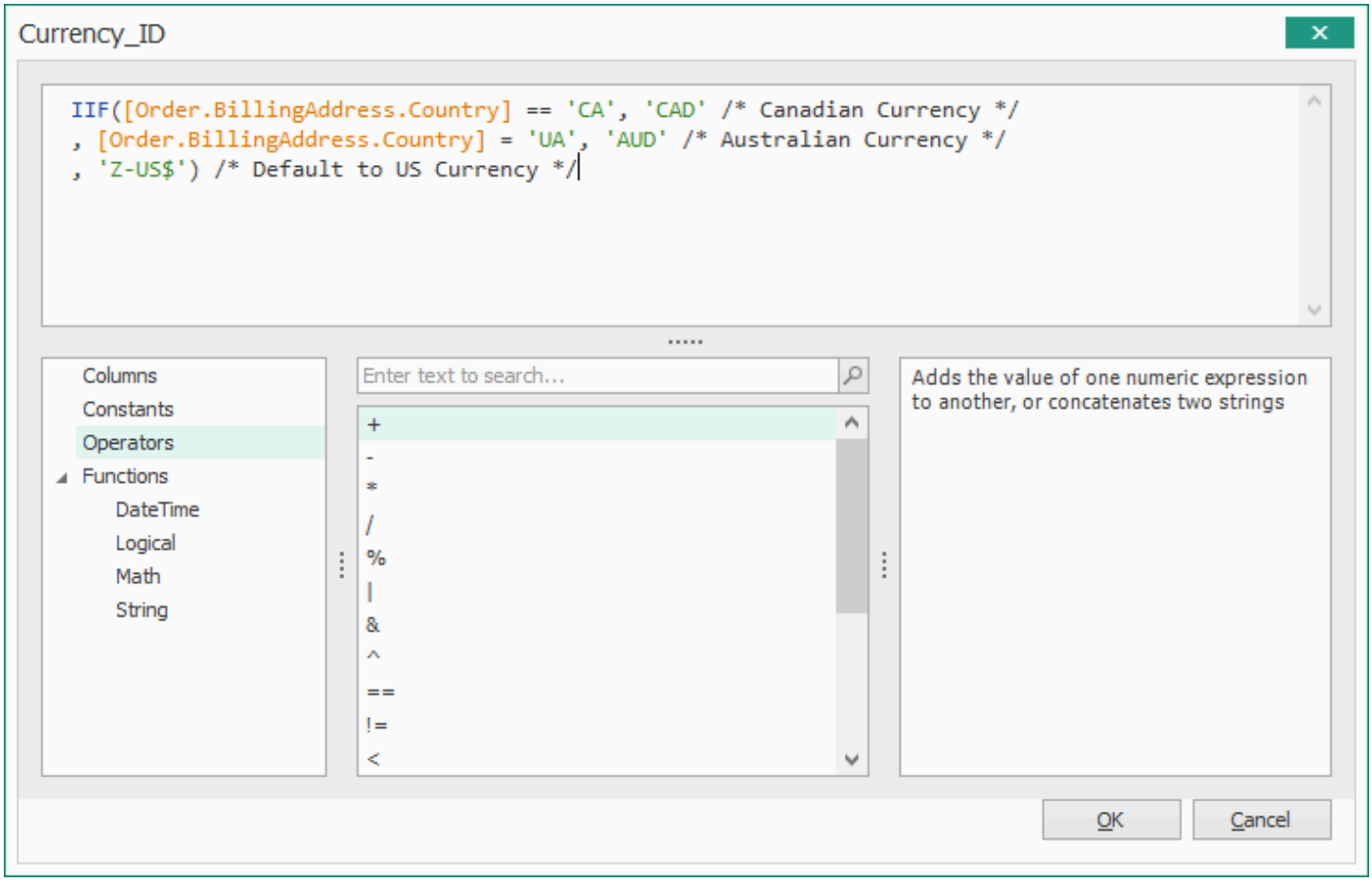



Rather than a single customer name field with the customer's full name, Shopify has separate first and last name fields, so the Contact_Person field cannot be populated by simply mapping one field to another. The Concat function is used to append the Shopify Customer last name to the first name, with 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 hard-coded 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.




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.

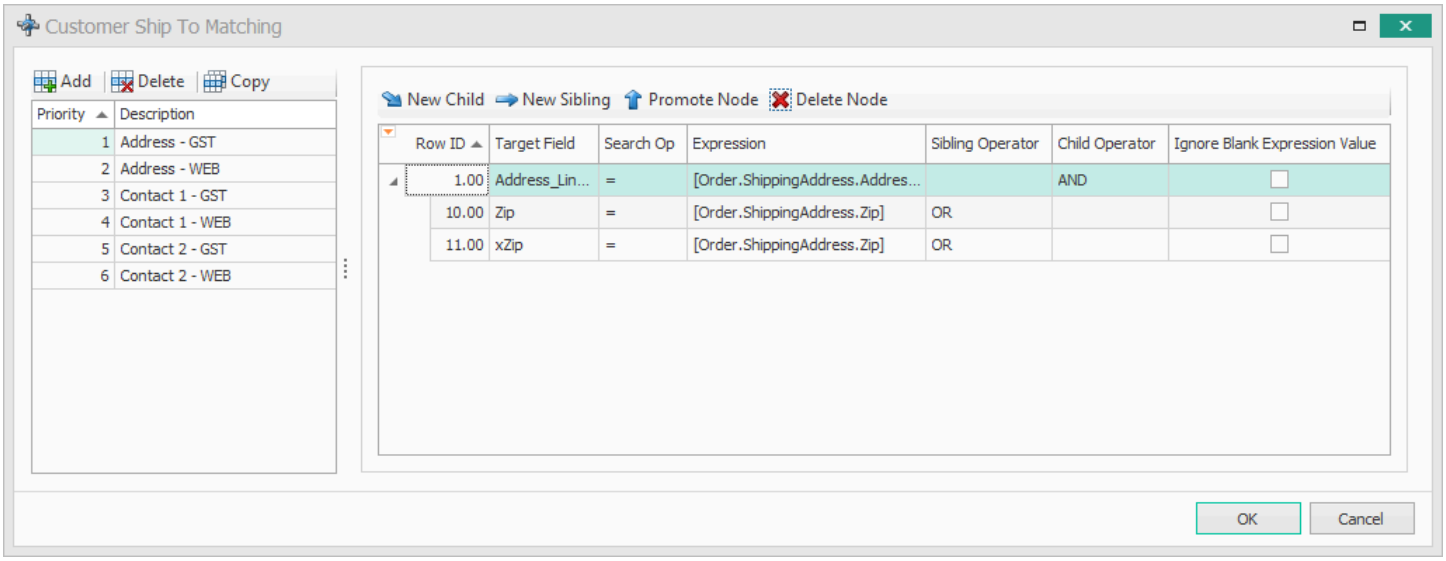



The complete expression language documentation can be found here.

## 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 a customer address on 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:




# Product Export

## Overview

The Product Export updates products in Shopify based on new and changed item masters in SalesPad. Products which exist in both systems are updated in Shopify, and products which do not yet exist in Shopify are automatically created as part of this process. This component provides flexibility to specify which items and associated values are pushed to Shopify.

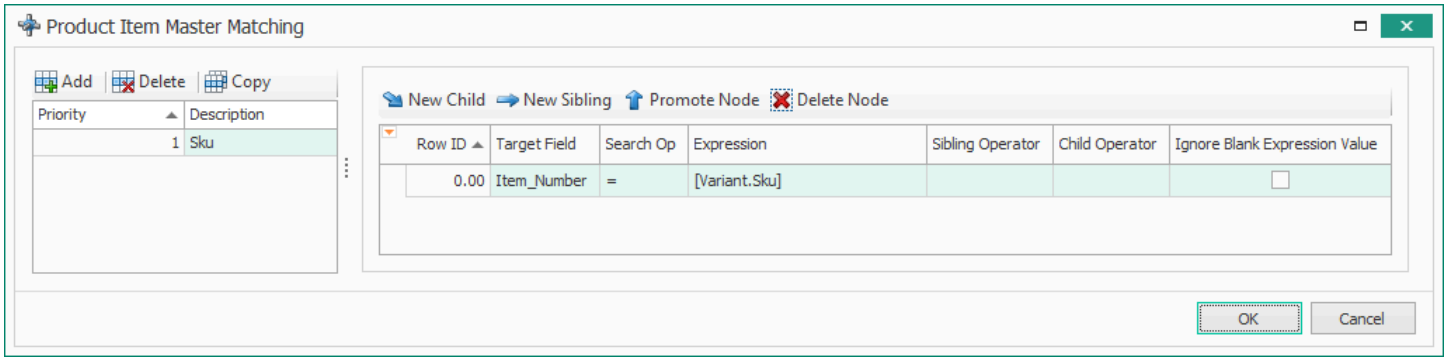
## General Settings

**Forward Item Master After Initial Export** - *If set to True, the product export will forward item masters when they're first exported to Shopify.*

**Product Export Page Size** - *Determines the maximum number of Shopify products that SalesPad will export in a single API call. Default: 2500.*

## Matching Settings

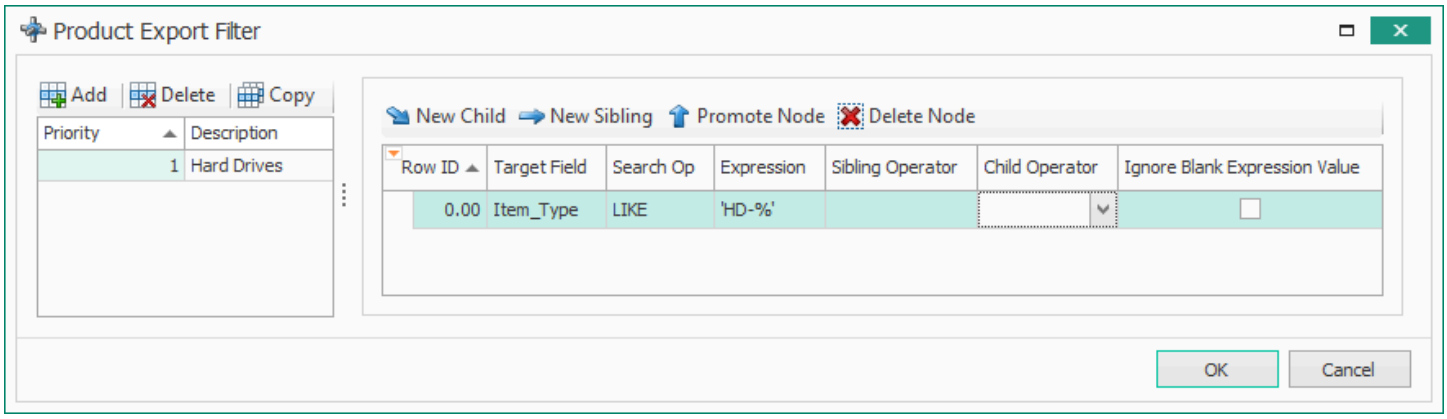
**Product Item Master Matching** - *Define the mappings for looking up the Item Master for the Product Export.*




## Lookup Settings

**Product Export Filter** - *Define the criteria for determining which items export to Shopify.*

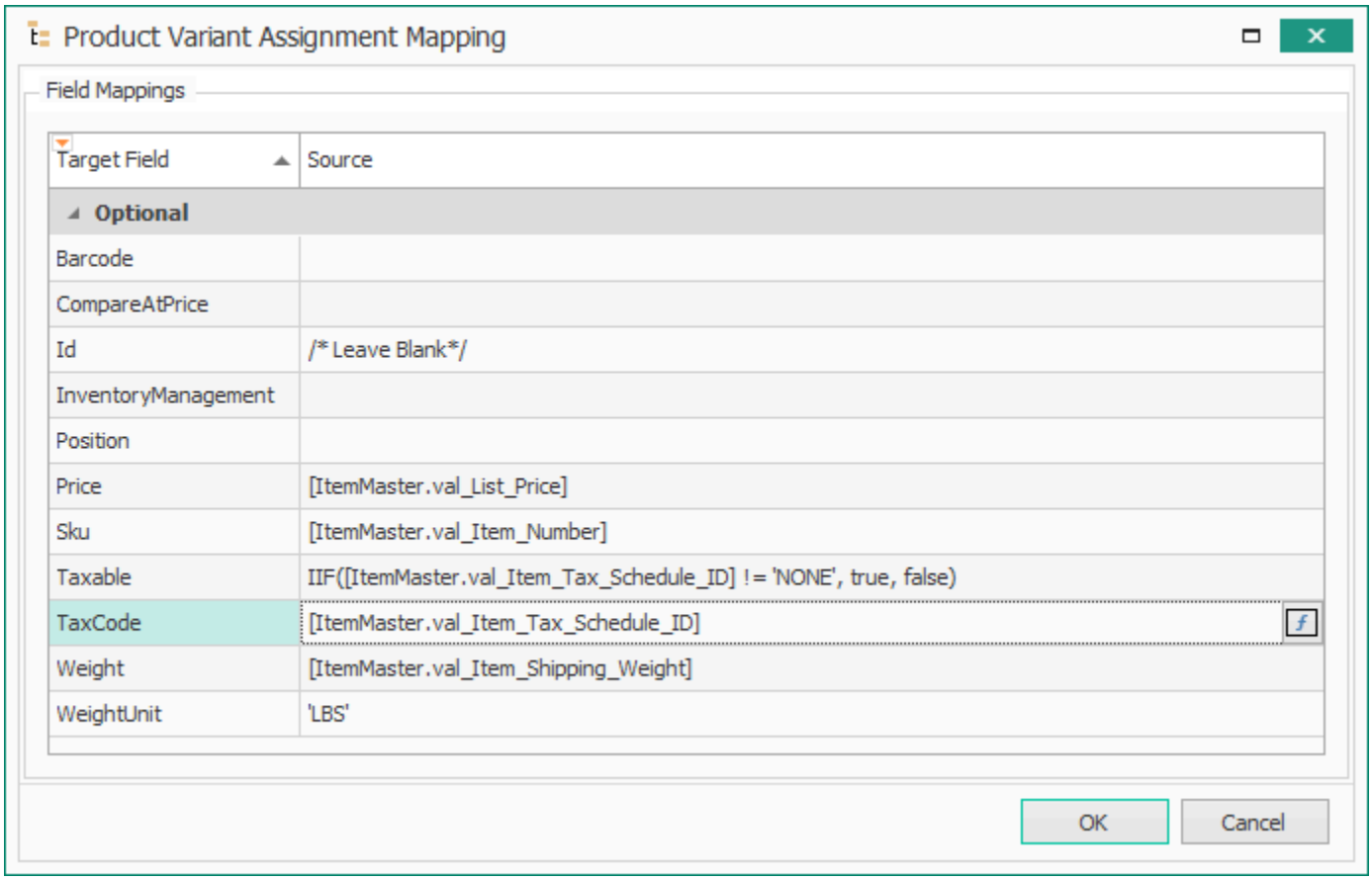
Only one priority row should be added to the left grid. The conditions on the right will be used to determine which Item Masters will be exported to Shopify. In the example below, the system will export all items that start with *HD-*.




## Assignment Settings

**Product Assignment Mapping** - *Define the mappings to be used when exporting a Product.*

**Product Variant Assignment Mapping** - *Define the mappings to be used when exporting a Product Variant.*




## Processing

The Product Export bulk loads products from Shopify and matches them to item masters in SalesPad based on the *Product Item Master Matching* setting. The *Product Export Filter* is used to filter out any items which should not be exported to Shopify. Then products that already exist in Shopify are updated if their corresponding SalesPad item has changed, and products that do not yet exist in Shopify are automatically created.

Both creation and updates use the same *Product Assignment Mapping* so that fields are updated the same way. Product variant information for both creation and updates is updated based on the *Product Variant Assignment Mapping*.

## Endpoints

### Get Products

At the beginning of Product Export, the integration bulk loads various data from Shopify. The products themselves are pulled, as well as their product variant information.

GraphQL Query:

```
Unset
{
  products {
    edges {
      node {
        id
        descriptionHtml
        handle
        productType
        status
        templateSuffix
        title
        vendor
        variants {
          edges {
            node {
              id
              barcode
              compareAtPrice
              position
              price
              sku
              taxCode
              taxable
              weight
              weightUnit
              inventoryManagement
}}}}}}}
```

This query is wrapped in a Bulk Operation Run Query which returns each product.

Sample response:

```
Unset
{"id":"gid:\/\/shopify\/Product\/8763188936963","descriptionHtml":"","handle":"the-draft-snowboard","prod
uctType":"snowboard","status":"DRAFT","templateSuffix":null,"title":"The Draft
Snowboard","vendor":"Snowboard Vendor"}
```

```
{"id":"gid:\/\/shopify\/ProductVariant\/45970737824003","barcode":null,"compareAtPrice":null,"position":1
,"price":"2629.95","sku":"","taxCode":"","taxable":true,"weight":0.0,"weightUnit":"POUNDS","inventoryMana
gement":"SHOPIFY","__parentId":"gid:\/\/shopify\/Product\/8763188936963"}
{"id":"gid:\/\/shopify\/Product\/8763188969731","descriptionHtml":"","handle":"the-collection-snowboard-h
ydrogen","productType":"snowboard","status":"ACTIVE","templateSuffix":null,"title":"The Collection
Snowboard: Hydrogen","vendor":"Hydrogen Vendor"}
{"id":"gid:\/\/shopify\/ProductVariant\/45970737856771","barcode":null,"compareAtPrice":null,"position":1
,"price":"600.00","sku":"","taxCode":"","taxable":true,"weight":0.0,"weightUnit":"POUNDS","inventoryManag
ement":"SHOPIFY","__parentId":"gid:\/\/shopify\/Product\/8763188969731"}
{"id":"gid:\/\/shopify\/Product\/8763189002499","descriptionHtml":"","handle":"the-archived-snowboard","p
roductType":"snowboard","status":"ARCHIVED","templateSuffix":null,"title":"The Archived
Snowboard","vendor":"Snowboard Vendor"}
```


The bulk operation separates the products and variants in the response into separate JSON objects which are then used to match to existing item masters in GP.

### Update Product

The Product Export then updates products which already exist in both systems.

GraphQL Query:

```
Unset
mutation call($input: ProductInput!) {
  productUpdate(input: $input) {
    product {
      id
      descriptionHtml
      handle
      productType
      status
      tags
      templateSuffix
      title
      vendor
      variants {
        edges {
          node {
            id
            barcode
```

```
            compareAtPrice
            position
            price
            sku
            taxCode
            taxable
            weight
            weightUnit
            inventoryManagement
          }
        }
      }
    }
    userErrors {
      field
      message
    }
  }
}
```

This query is wrapped in a Bulk Operation Run Query which returns a succeeded value and a message along with any error codes or exceptions thrown.

### Create Product

Then it creates Shopify products for any SalesPad items that did not have a Shopify product match.

GraphQL Query:

```
Unset
mutation call($input: ProductInput!) {
  productCreate(input: $input) {
    product {
      id
      descriptionHtml
      handle
      productType
      status
      tags
      templateSuffix
```

```
      title
      vendor
      variants {
        edges {
          node {
            id
            barcode
            compareAtPrice
            position
            price
            sku
            taxCode
            taxable
            weight
            weightUnit
          }
        }
      }
    }
    userErrors {
      field
      message
    }
  }
}
```

This query is wrapped in a Bulk Operation Run Query which returns a succeeded value and a message along with any error codes or exceptions thrown.

## Scripts

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

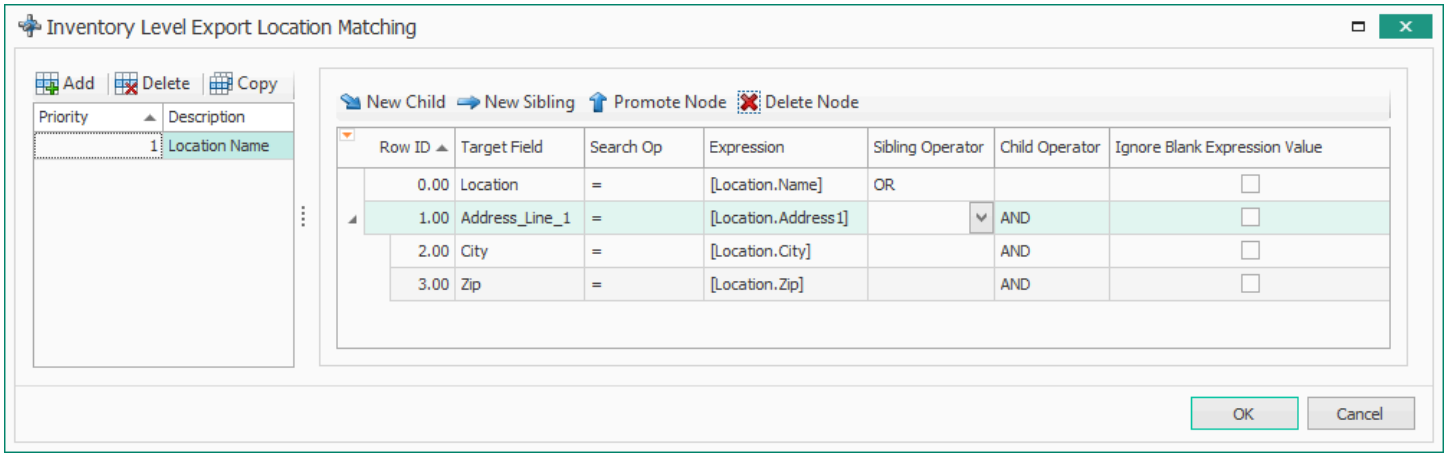
## Inventory Level Export

## Overview

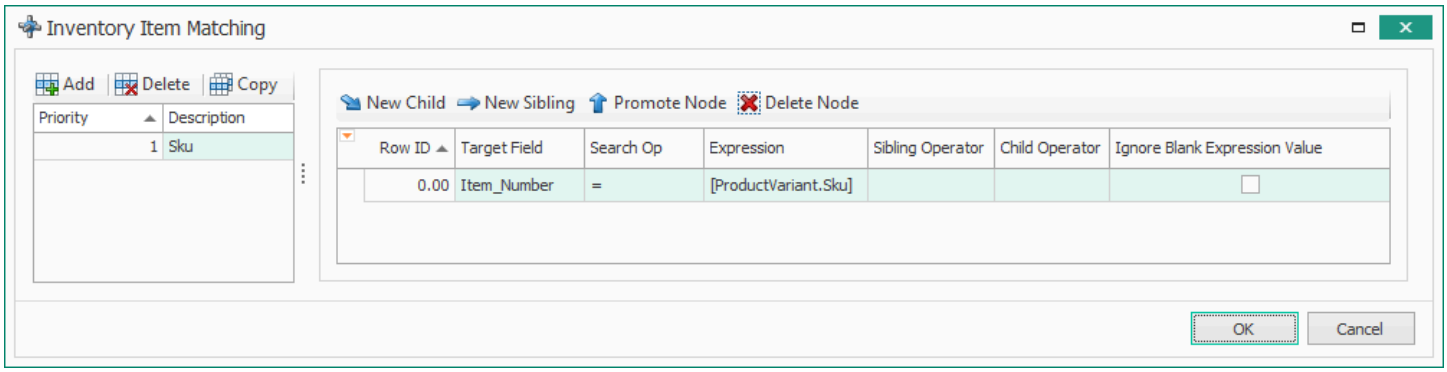
The Inventory Level Export updates item availability information in Shopify based on current inventory levels in SalesPad warehouses.  This ensures that customers placing orders on the website know how much they can order before they will have to wait for backorders.

## Matching Settings

**Inventory Level Export Location Matching** - *Define the criteria for matching a Shopify location to a GP Location in the Inventory Level Export.*

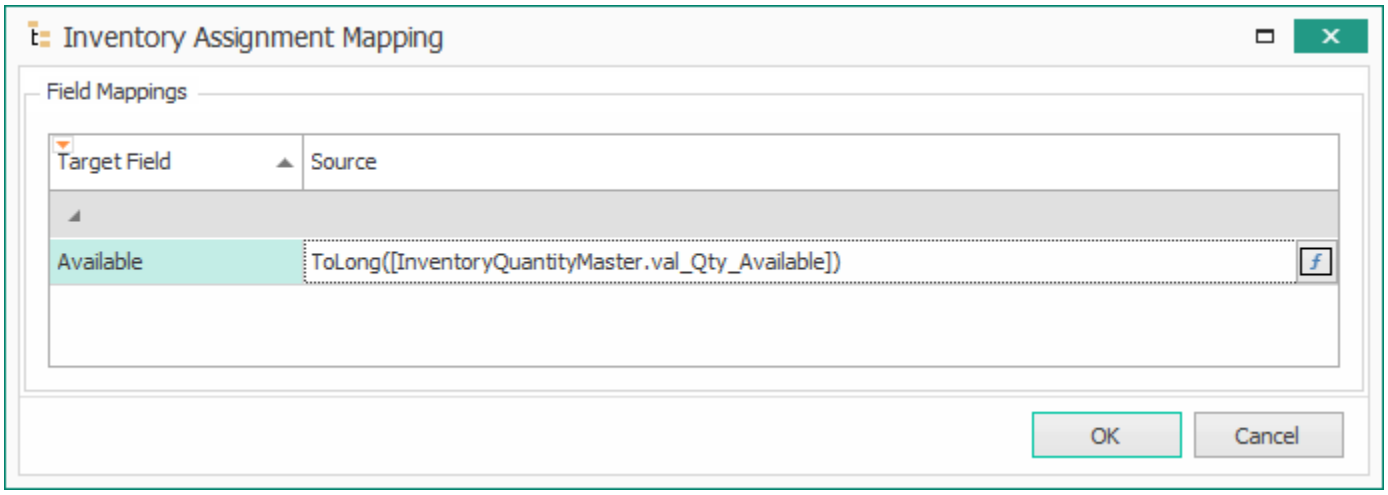



**Inventory Item Matching** - *Define the criteria for which GP inventory levels to export to Shopify.*




## Assignment Settings

**Inventory Assignment Mapping** - *Define the mappings to be used when exporting inventory quantities.*




## Processing

The Inventory Level Export bulk loads inventory levels and products from Shopify. The *Inventory Item Matching* setting is used to match Shopify products to SalesPad items, and the *Inventory Level Export Location Matching* setting is used to match Shopify locations to SalesPad warehouses. Then the *Inventory Assignment Mapping* setting is used to update the Shopify item availability based on its availability in its corresponding SalesPad item and warehouse.

## Endpoints

### Get Products

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

GraphQL Query

```
Unset
{
    productVariants {
        edges {
            node {
                id
                sku
                inventoryItem {
                    id
                    legacyResourceId
                    inventoryLevels {
                        edges {
```

```
                        node {
                            id
                            available
                            location {
                                id
                                legacyResourceId
                                name
                            }
                        }
                    }
                }
            }
        }
    }
}
```

This query is wrapped in a Bulk Operation Run Query which returns each product.

Sample Response:

```
Unset
{"id":"gid:\/\/shopify\/ProductVariant\/45970737824003","sku":"","inventoryItem":{"id":"gid:\/\/shopify\/
InventoryItem\/48049714659587","legacyResourceId":"48049714659587"}}
{"id":"gid:\/\/shopify\/InventoryLevel\/114950734083?inventory_item_id=48049714659587","available":20,"lo
cation":{"id":"gid:\/\/shopify\/Location\/78633042179","legacyResourceId":"78633042179","name":"WAREHOUSE
"},"__parentId":"gid:\/\/shopify\/ProductVariant\/45970737824003"}
{"id":"gid:\/\/shopify\/ProductVariant\/45970737856771","sku":"","inventoryItem":{"id":"gid:\/\/shopify\/
InventoryItem\/48049714692355","legacyResourceId":"48049714692355"}}
{"id":"gid:\/\/shopify\/InventoryLevel\/114950734083?inventory_item_id=48049714692355","available":50,"lo
cation":{"id":"gid:\/\/shopify\/Location\/78633042179","legacyResourceId":"78633042179","name":"WAREHOUSE
"},"__parentId":"gid:\/\/shopify\/ProductVariant\/45970737856771"}
{"id":"gid:\/\/shopify\/ProductVariant\/45970737889539","sku":"","inventoryItem":{"id":"gid:\/\/shopify\/
InventoryItem\/48049714725123","legacyResourceId":"48049714725123"}}
```

### Update Inventory Levels

The Inventory Level Export then updates inventory levels for products which exist in both systems.

GraphQL Query:

```
Unset
mutation inventoryBulkAdjustQuantityAtLocation($inventoryItemAdjustments: [InventoryAdjustItemInput!]!,
$locationId: ID!) {
    inventoryBulkAdjustQuantityAtLocation(inventoryItemAdjustments: $inventoryItemAdjustments,
locationId: $locationId) {
        inventoryLevels {
            id
            available
        }
        userErrors {
            field
            message
        }
    }
}
```

This query is wrapped in a Bulk Operation Run Query which returns the adjusted InventoryLevels and any UserErrors.

## Customer and Order Import

### Overview

The Order Import retrieves all orders in Shopify that are ready to be imported. It matches customer and contact information to existing customers and contacts in SalesPad, or it creates customers and addresses as needed, before creating the sales orders. SalesPad orders and lines are created based on configurable mapping settings, then they are linked back to their corresponding Shopify orders, and finally their Shopify orders are flagged as imported.

### 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.*

**Financial Status Filter** - *Specify one or more financial statuses which a Shopify order must have in order to be imported.*

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

**Fulfillment Status Filter** - *Specify the fulfillment status which orders must have for the order import to import them.*

**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 Shopify Order Comments** - *Specify which tab to import Shopify 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. (Maximum of 100) Defaults to 50.*

**Payment Financial Status Filter** - *Specify one or more financial statuses for which a payment will be created after a successful import.*

**Processed Order Tag** - *After an order is imported to SalesPad, this tag will be written to the Shopify order to prevent subsequent imports. Defaults to "EXPORTED_TO_SALESPAD".*

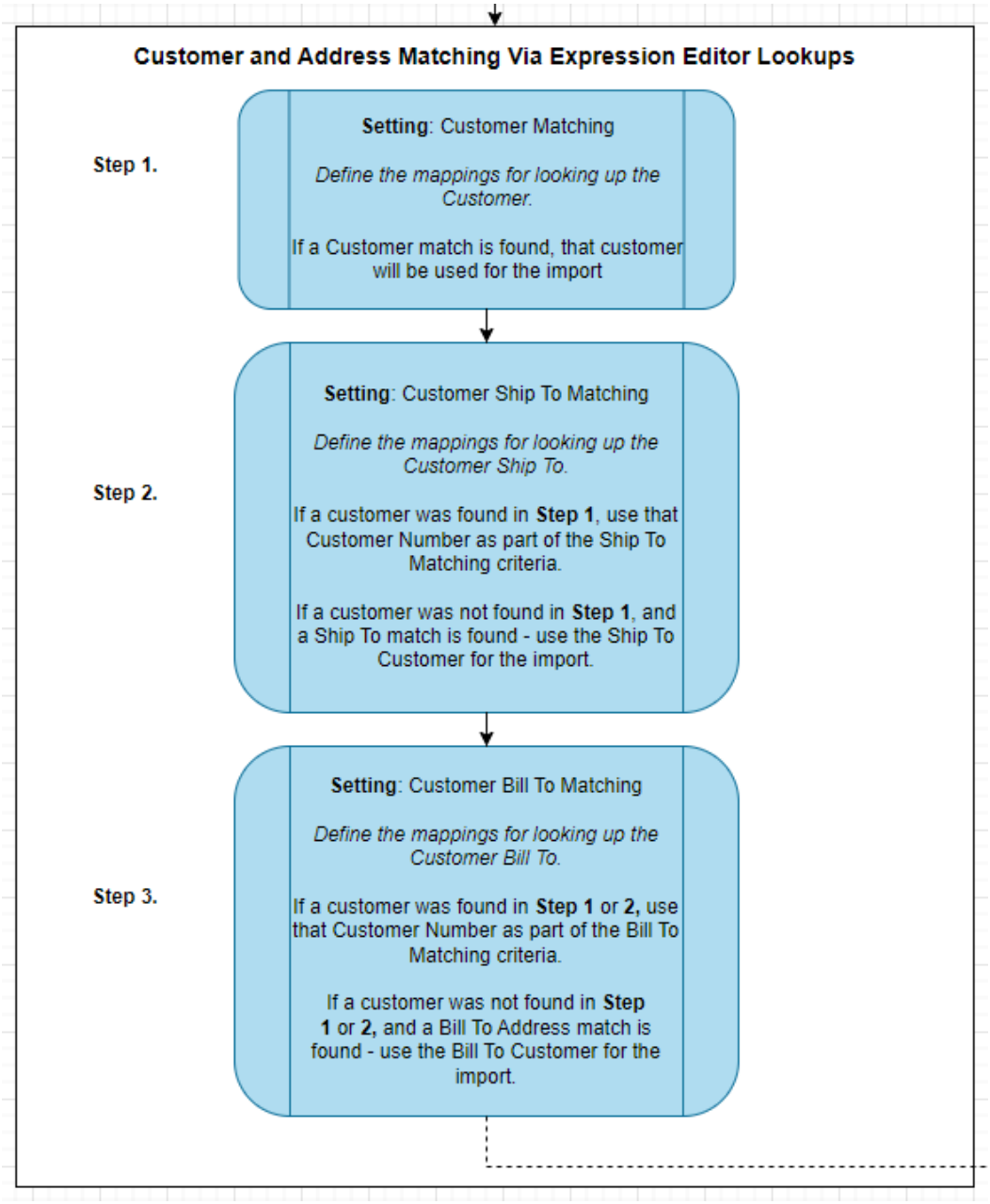
**Roll Back Order Import Transaction On Error** - *When enabled, the transaction encompassing the order import will be rolled back when an error occurs. This prevents data from a partially completed import from being saved to the database. Defaults to True.*

## Matching Settings

These settings use Expression Editors that allow creating custom matching criteria to determine how a Shopify entity should match to a SalesPad entity.

Each Matching setting will build one or more SQL queries to attempt to find a match in the SalesPad 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.

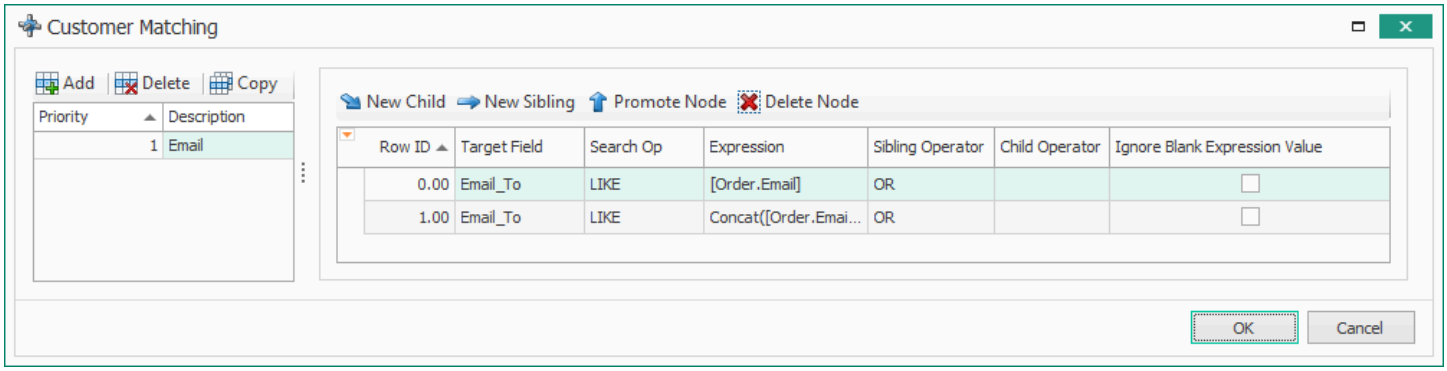
The diagram below illustrates the sequence in which SalesPad attempts to match a SalesPad Customer, Ship To Address, and Bill To Address each time a Shopify Order is imported.




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

This setting is used as the first attempt to match a SalesPad Customer to the Shopify 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.

The default settings attempt to match a SalesPad Customer based on the email of the Shopify Order. The conditions include a LIKE operator combined with an OR sibling operator. The LIKE operator indicates that the Email_To field must contain, but not exactly match, the expression. The OR sibling operator indicates that at least one of the conditions must match in order for a customer record to be considered a match. In the example below, the customer's Email_To field must match either the Order.Email field, or the Order.Email field with a ';' character appended. For example, if the Shopify customer's email is "aaronfitz@gmail.com", a SalesPad customer with "aaronfitz@gmail.com;aaron@fitzelectrical.com" would be a match. The approximate SQL lookup executed for this condition would be:
"WHERE (customer.Email_To LIKE '%aaronfitz@gmail.com%') or customer.Email_To LIKE '%aaronfitz@gmail.com;%'




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 Shopify 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.*

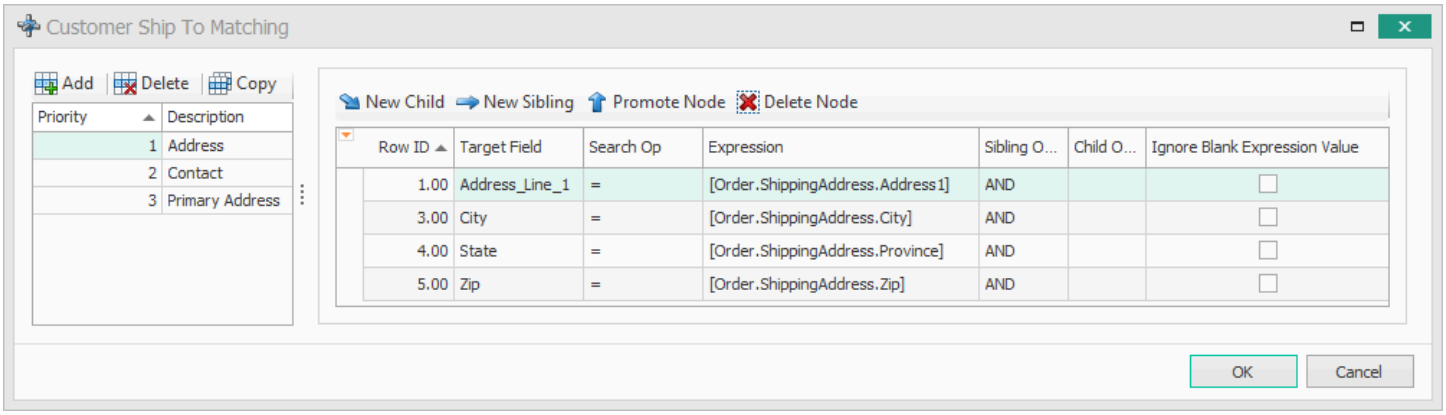
This setting determines how SalesPad will attempt to look up the Ship To Address for an incoming Shopify order. If a SalesPad Customer was matched via the **Customer Matching** setting, the Ship To Address must also belong to that customer. Otherwise, the search will consider address codes across all 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 Shipping Address in three different steps.

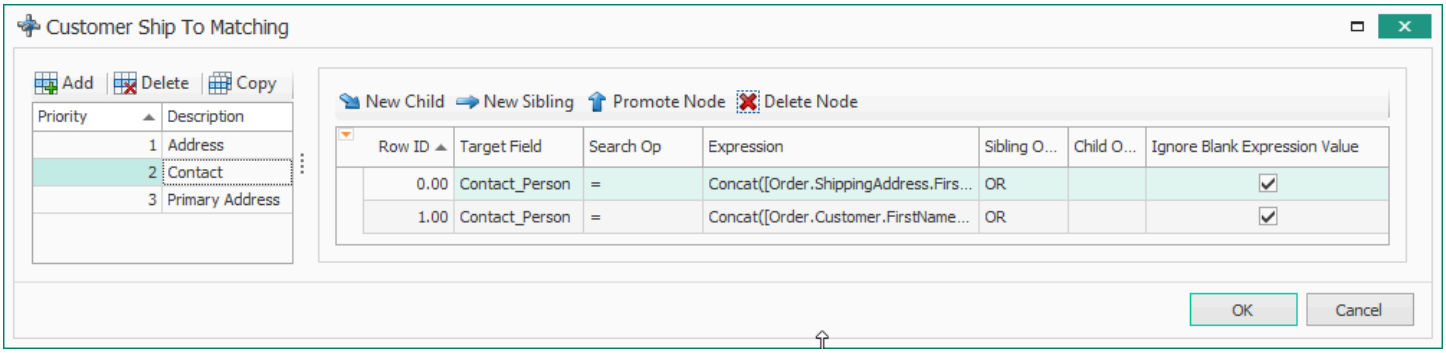
**Priority 1**

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



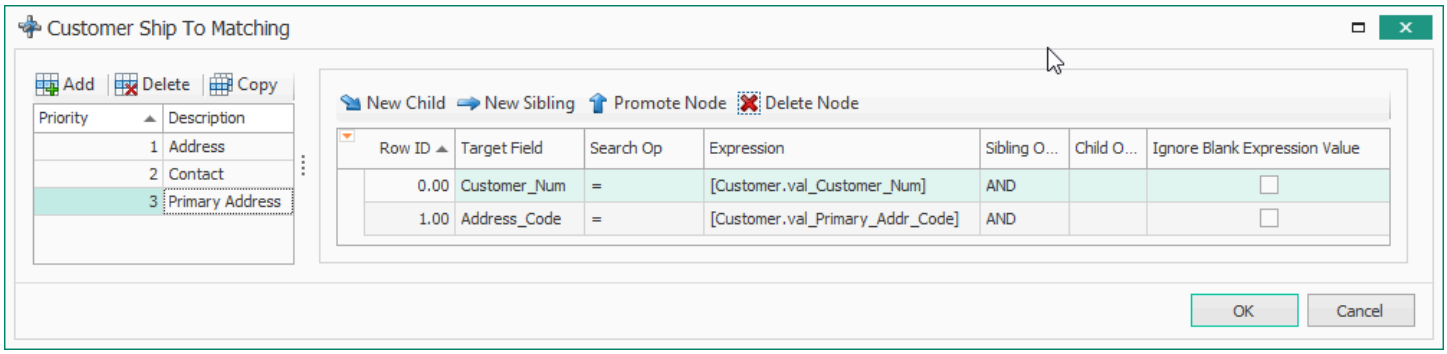

**Priority 2**

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




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

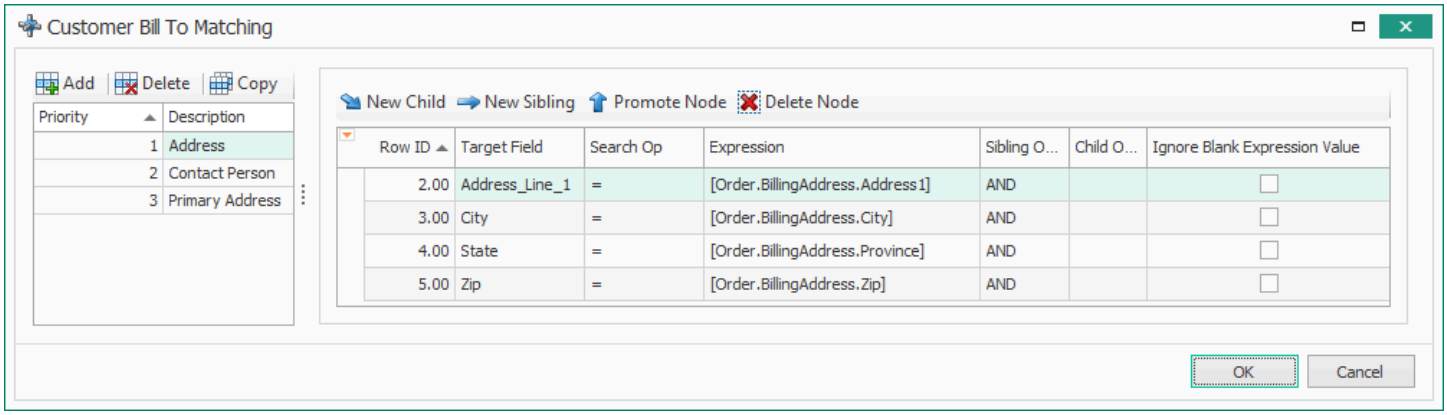
This setting determines how SalesPad will attempt to look up the Bill To Address for an incoming Shopify order. If a SalesPad Customer was matched via the **Customer Matching** setting, the Bill To Address must also belong to that customer. Otherwise, the search will consider address codes across all 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 Billing Address in three different steps.

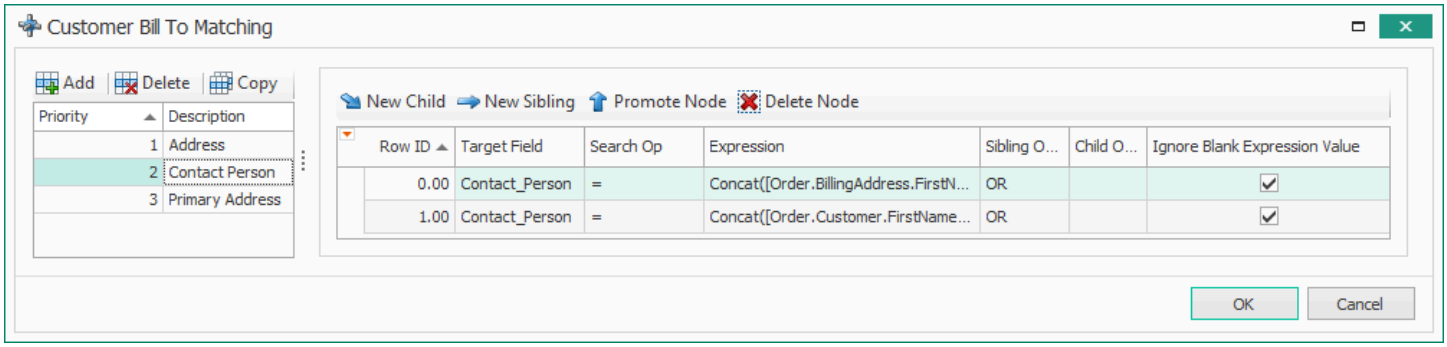
**Priority 1**

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



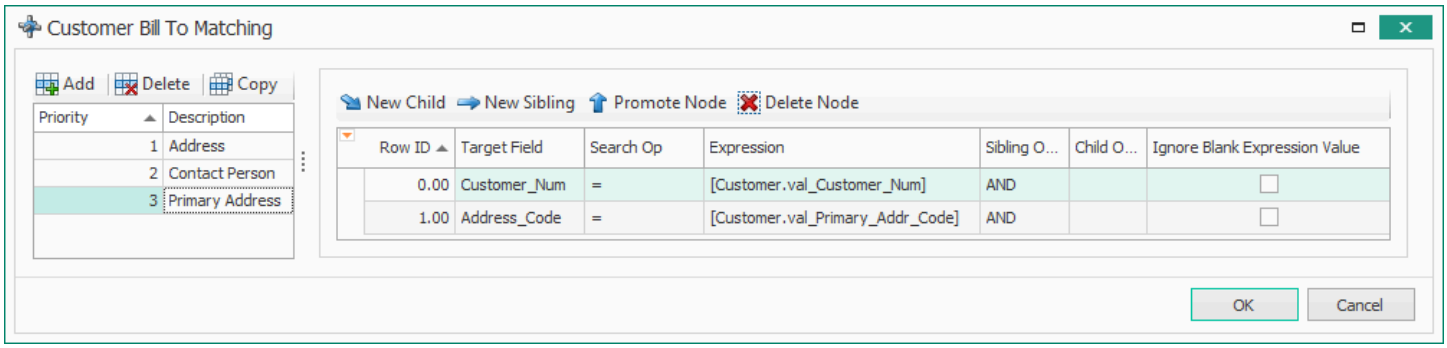

**Priority 2**

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




**Priority 3**

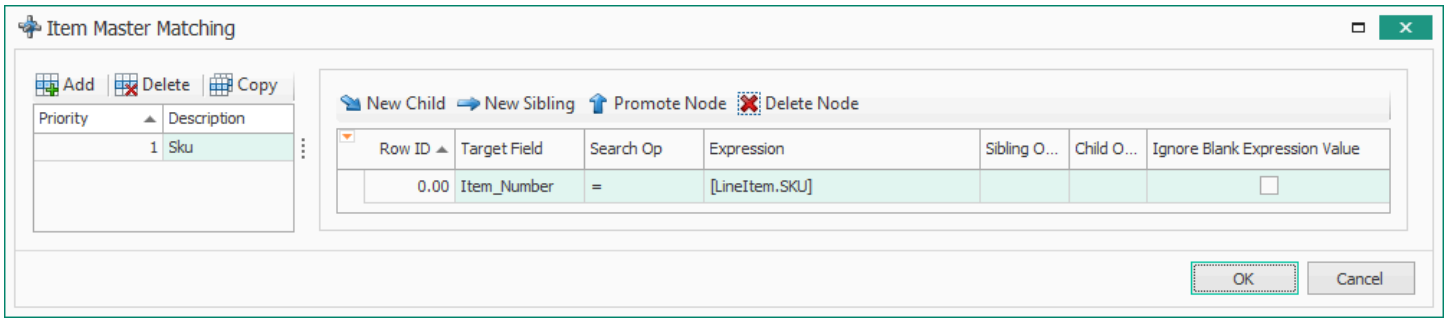
Primary Address: Attempt to find a match based on the Primary Address for the Customer. This assumes that the Customer was matched via the *Customer Matching* setting.




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

Each Shopify Line Item will use this setting to find the corresponding SalesPad Item.

The default settings attempt to match a Shopify SKU directly to a SalesPad Item Number.



## Assignment Settings

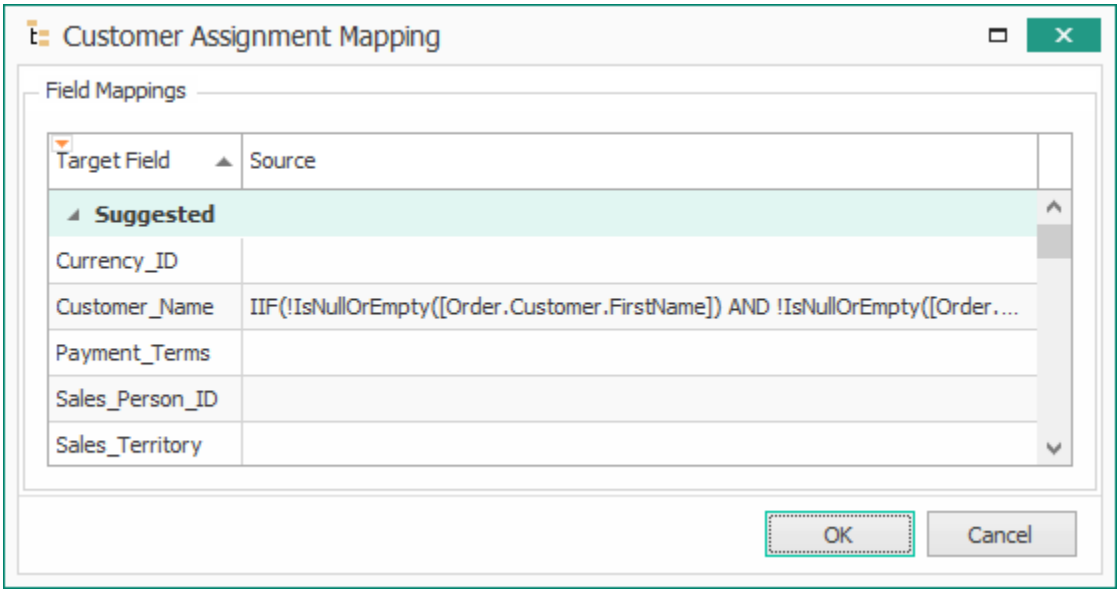
Assignment settings use Expression Editors to define how newly created Customers, Orders and Sales Lines will be populated when created during the order import process.

The target objects will be SalesPad Customers, Contacts, Sales Documents, and Sales Line Items. The source objects will be Shopify Customers, Contacts, Orders, and Line Items.

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.

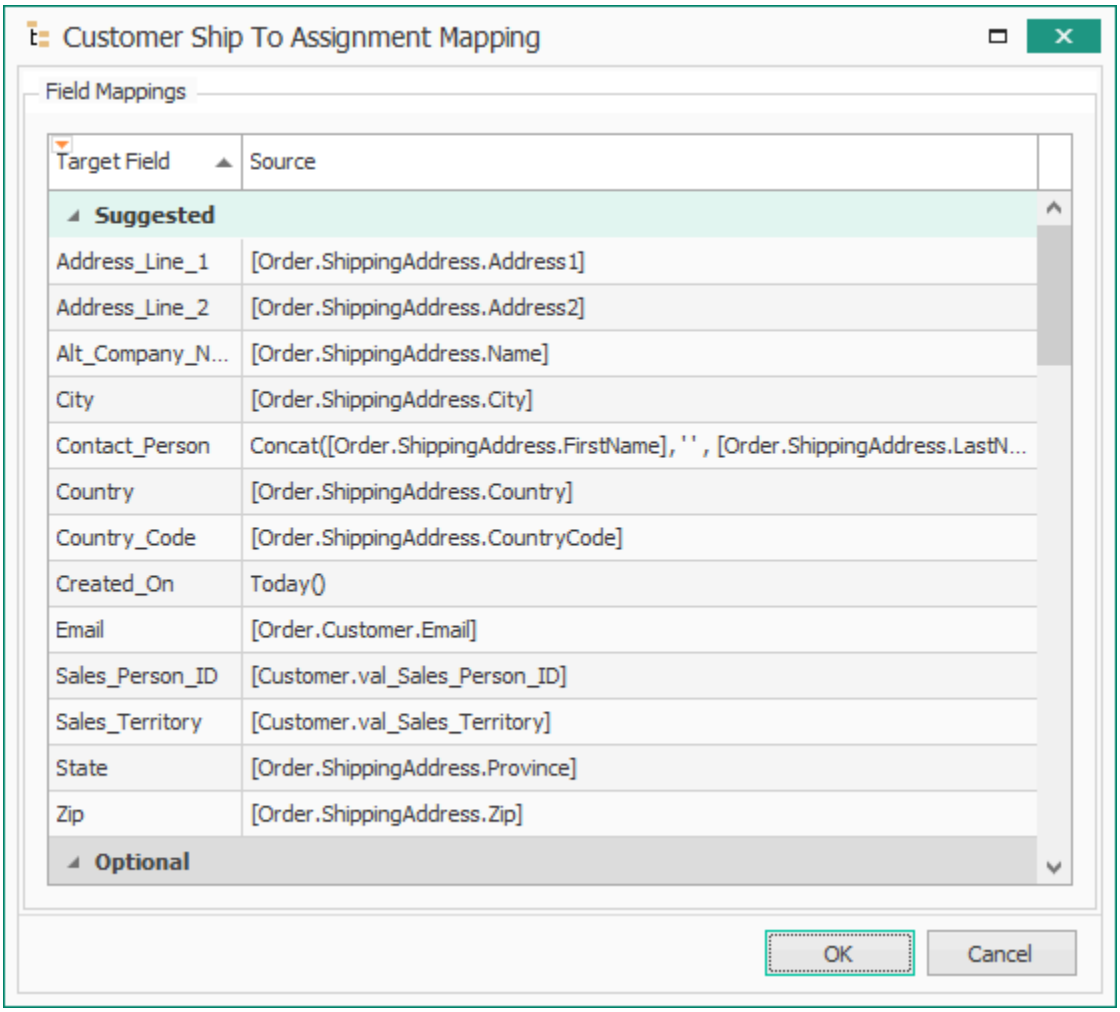
**Customer Assignment Mapping** - *Define the mappings to be used when creating a new Customer.*

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



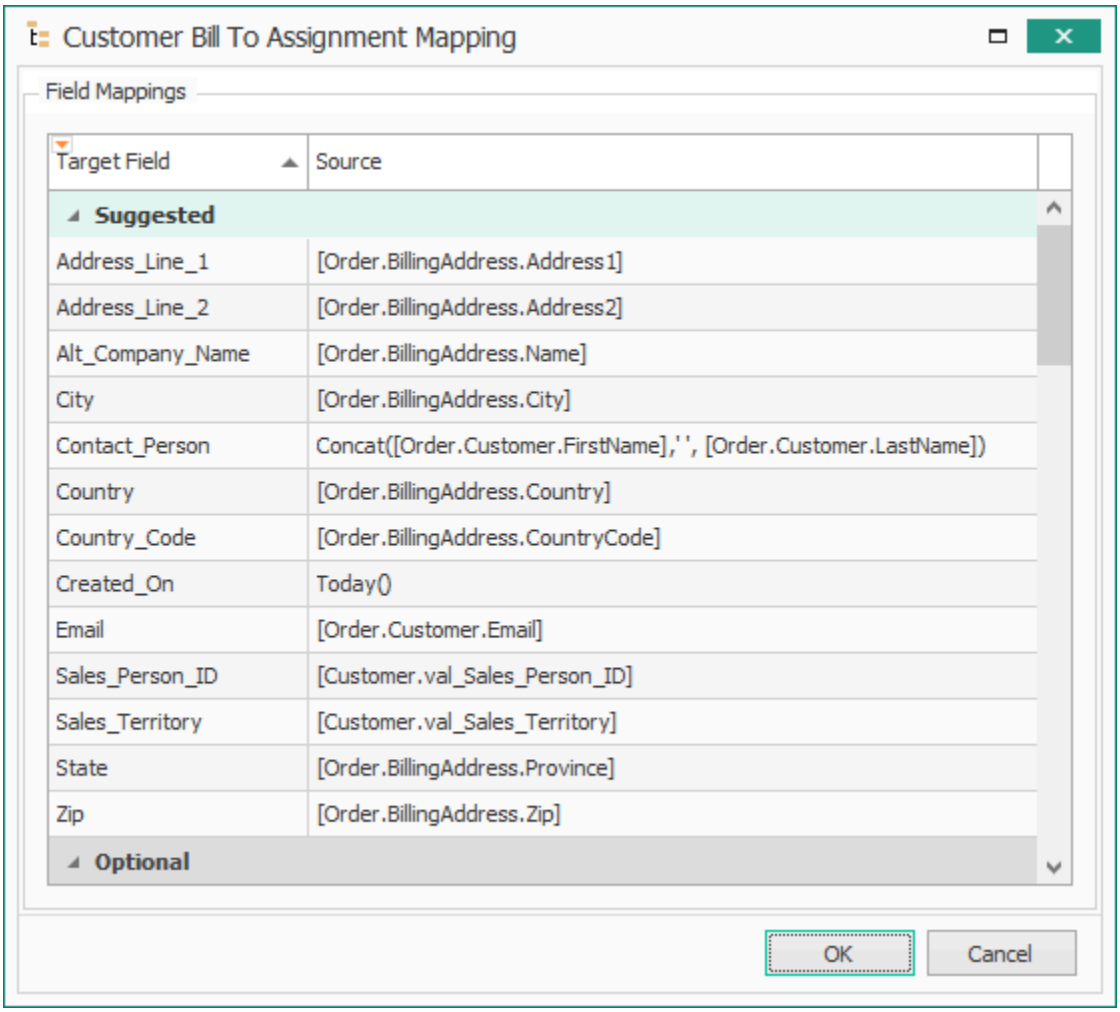

**Customer Ship To Assignment Mapping** - *Define the mappings to be used when creating a new Customer Ship To Address.*

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



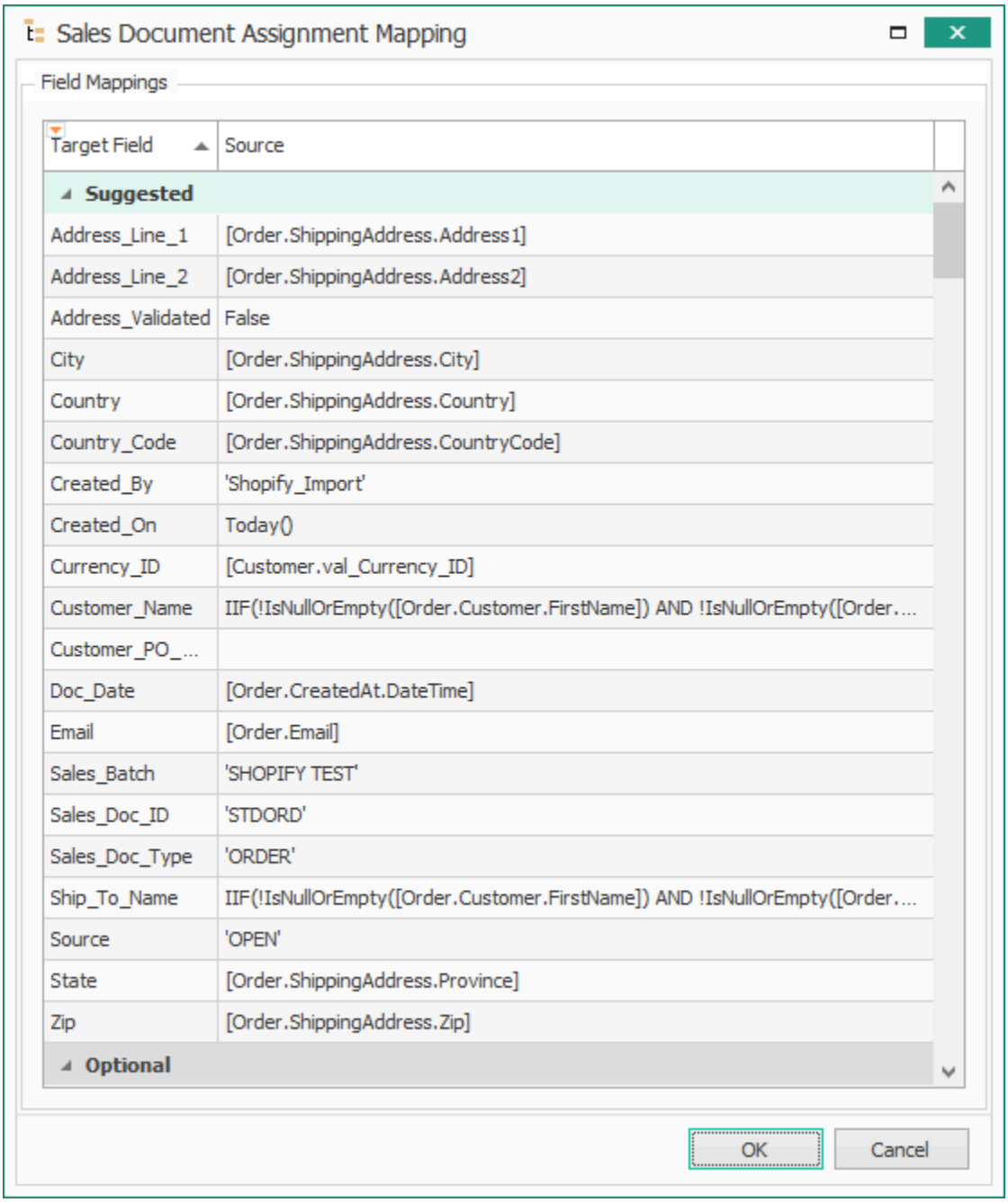

**Customer Bill To Assignment Mapping** - *Define the mappings to be used when creating a new Customer Bill To Address.*

If an existing SalesPad 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.




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

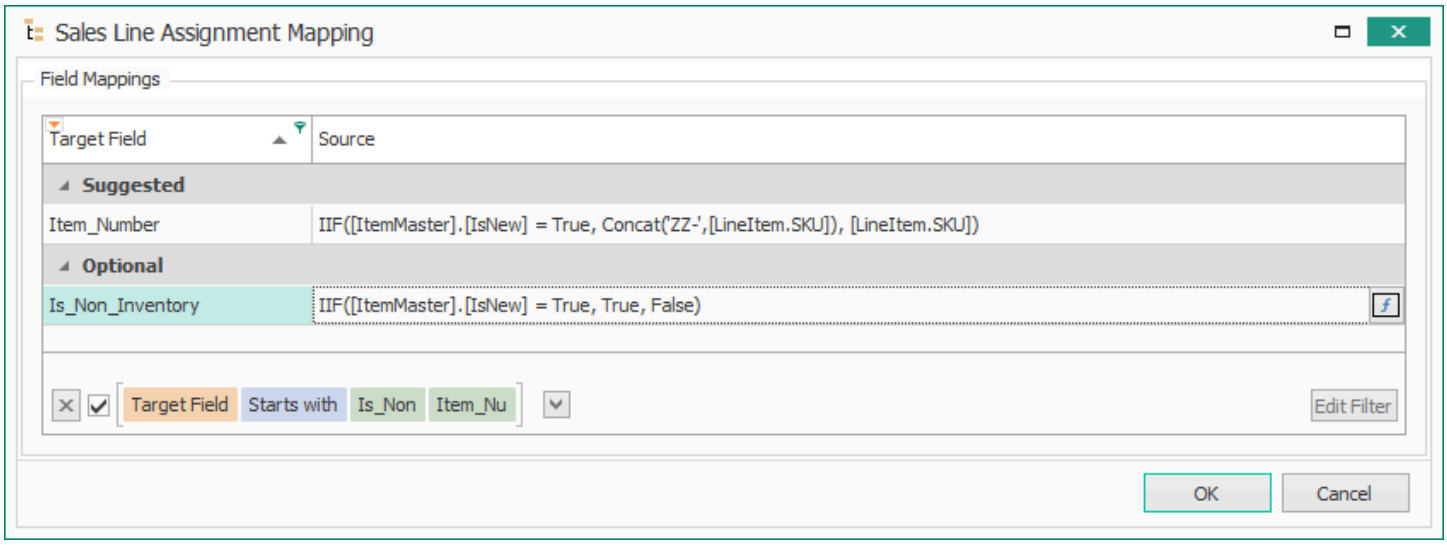
When the Shopify Order is imported, this setting will be used to populate the header level fields on the SalesPad Sales Document.




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

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

The Item Number is determined by the Item Master that is matched by the *Item Master Matching* setting. If a valid item was not found, then an expression like the one below can be used to import the Shopify Line Item as a Sales Line Item that is a non-inventory item. (Note that *ZZ-* is the default value in the Non-Inventory Prefix setting and should be updated to the local environment's value)




A few other line item fields are set by the integration component by default, but they can be overridden by this setting: *Unit_Price*, *Markdown_Amount*, *Tax_Amount*, *Funct_Tax_Amount*.

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

## Processing

For each sales order in the website, the Order Import component attempts to match a SalesPad Customer to the Shopify Customer based on the *Customer Ship To Matching*, *Customer Bill To Matching*, and *Customer Matching settings*. If no match is found, a new customer is created in SalesPad based on the *Customer Assignment Mapping setting*, and address codes for the new customer are 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 is 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 via the *Sales Document Payment Mapping* setting. A tag is added to each order in Shopify to indicate that importing was completed.

## Endpoints

### Retrieve Orders Keys To Import

The Customer and Order import begins by retrieving the keys of sales orders in Shopify which are ready to be imported into SalesPad.

This GraphQL query can change based on some of the General Settings mentioned above. The below query looks for documents that have an open status, a fulfillment status of unshipped, a financial status of paid or unpaid, and that the order does not have a tag called "EXPORTED_TO_SALESPAD" which indicates that the order has already been imported into SalesPad.

GraphQL Query:

```
Unset
{
    orders (query: "status:open AND fulfillment_status:unshipped AND (financial_status:paid
OR financial_status:unpaid) AND NOT tag:EXPORTED_TO_SALESPAD"){
        edges {
            node {
                legacyResourceId
            }
        }
    }
}
```

This query is wrapped in a Bulk Operation Run Query which returns each order.

Sample Response:

```
Unset
{"legacyResourceId":"6055540457731"}
```

### Retrieve Orders To Import

Retrieving the full order information uses Shopify's ShopifySharp library's OrderService. The keys from the first step are passed into this service as a list, and the service returns all orders that correspond to

those keys. This ShopifySharp library uses Shopify's REST library and therefore likely uses these endpoints.

### Update Processed Order Tag

After an order has been successfully imported into SalesPad, if the Processed Order Tag setting has a value, then a tag is created on the order via the ShopifySharp library's OrderService.

## 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 Shopify 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.

**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 Script** - *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.

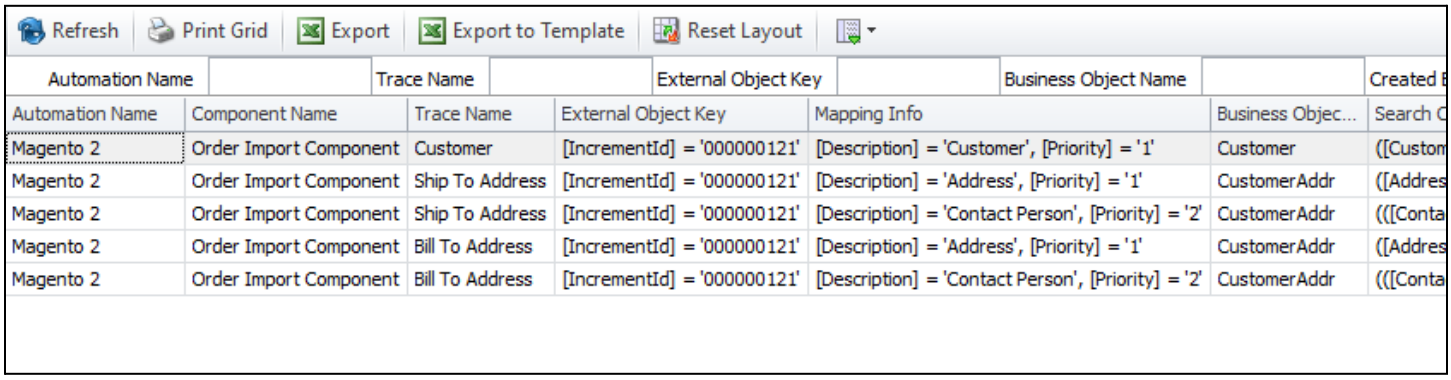
**Item Master Matching Script** - *A C# Script that can be used to load SalesPad.Bus.ItemMaster item.*

*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 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.




| Automation Name | Component Name | Trace Name | External Object Key | Mapping Info | Business Objec... | Search C |
|---|---|---|---|---|---|---|
| Magento 2 | Order Import Component | Customer | [IncrementId] = '000000121' | [Description] = 'Customer', [Priority] = '1' | Customer | ([Custom |
| Magento 2 | Order Import Component | Ship To Address | [IncrementId] = '000000121' | [Description] = 'Address', [Priority] = '1' | CustomerAddr | ([Addres |
| Magento 2 | Order Import Component | Ship To Address | [IncrementId] = '000000121' | [Description] = 'Contact Person', [Priority] = '2' | CustomerAddr | (([Conta |
| Magento 2 | Order Import Component | Bill To Address | [IncrementId] = '000000121' | [Description] = 'Address', [Priority] = '1' | CustomerAddr | ([Addres |
| Magento 2 | Order Import Component | Bill To Address | [IncrementId] = '000000121' | [Description] = 'Contact Person', [Priority] = '2' | CustomerAddr | (([Conta |

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 sends tracking numbers and fulfillments back to Shopify for orders that were originally created by the Order Import component. It targets a specific workflow queue, and it forwards orders out of that queue once processing is complete. This component can also trigger Shopify to capture the order's payment and/or to notify the customer that order updates have occurred.

## General Settings

**Capture Payment Upon Completing Fulfillment** - *If set to True, pending authorizations on Shopify orders will be captured by Shopify once the order is completely fulfilled.*

**Export Queue** - *Queue that contains orders ready to be exported to Shopify. 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 Shopify store even if the order originated from a different automation instance.*

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

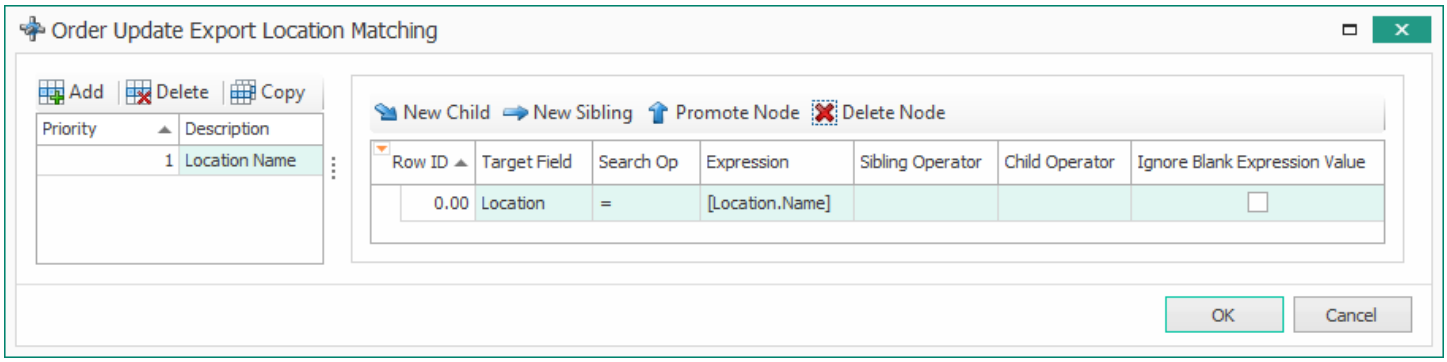
**Notify Customers** - *If set to True, Shopify will email the order's customer when the Order Update Export creates or updates fulfillments. Defaults to True.*

**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.

## Matching Settings

**Order Update Export Location Matching** - *Define the criteria for matching a Shopify location to a GP location in the Order Update Export.*




**Sales Line Matching** - *Define the optional mappings for matching a Sales Line to a Shopify line. Can be used to override the default line matching.*

When an order is imported from Shopify, a link record is created per sales line within the spAASalesLineImportLink database table. This record contains the sales line's External ID. If this setting is not defined, or it does not match to an existing SalesPad Sales Line, then the External ID for the sales line will be used to find the corresponding Shopify order line.

## Assignment Settings

**Captured Payment Mapping** - *Define the mappings to be used when creating a Sales Document Payment after capturing an authorization.*

## Processing

The Order Update Export processes all sales orders in the *Export Queue* setting. It uses the link that was created when each order was imported to call the Shopify API and update the corresponding Shopify order. For sales documents, the External ID designates the linked Shopify Order ID. For sales lines, the *Sales Line Matching* setting is used to find the corresponding Shopify line. The sales line also has an External ID that stores the ID of the corresponding Shopify line item; if the *Sales Line Matching* setting has been left blank, or did not find a match for a given line, then the External ID is used to find the Shopify line.

Fulfillments and tracking information are added for each Shopify sales line. If *Notify Customers* is enabled, then a flag is sent to trigger Shopify to notify the customer that this information has been updated. If *Capture Payment Upon Completing Fulfillment* is enabled, then a flag is sent to trigger Shopify to capture the pending payment.

The SalesPad order is forwarded in workflow if all processing was successful, and otherwise it is moved to the workflow queue in the *Export Failure Queue* setting.

## Endpoints

The Order Update Export retrieves each SalesPad order's corresponding Shopify order via Shopify's ShopifySharp library's OrderService, determines if fulfillment updates are needed, and then creates fulfillment header and detail information as needed. Each header is sent via GraphQL with the operation name as fulfillmentCreateV2 and the variable as the fulfillment header information.

GraphQL Query

```
Unset
mutation fulfillmentCreateV2($fulfillment: FulfillmentV2Input!) {
  fulfillmentCreateV2(fulfillment: $fulfillment) {
    fulfillment {
      status
    }
    userErrors {
      field
      message
    }
  }
}
```

# Order Voided Export

## Overview

The Order Voided Export component is used to automatically void orders in Shopify after they have been voided in SalesPad.

## General Settings

**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 Voided Order Export. Defaults to 50.*

## Matching Settings

**Order Cancel Options Assignment Mapping** - *Define the mappings to be used when canceling an order.*

## Processing

The Order Voided Export loads SalesPad orders that have been voided in the timeframe defined in the *Number Of Days To Look Back* setting and are linked to a Shopify order. For each order, if it has not already been voided in Shopify, then the *Order Cancel Options Assignment Mapping* setting is used to cancel it in Shopify.

## Endpoints

The Order Voided Export retrieves each voided SalesPad order's corresponding Shopify order via Shopify's ShopifySharp library's OrderService. Then the *Order Cancel Options Assignment Mapping* setting is used to map values from SalesPad to Shopify, and ShopifySharp's OrderService is used to cancel the order. The endpoint used to cancel the order is:

```
Unset
POST /admin/api/2023-07/orders/orderid/cancel.json
```