Bespoke Shipping

User Guide

QUICK START

This app allows you to build your own shipping rules without the inherit limitations of other apps.

If you have programming experience, examples can be found here to get you started:

http://parcelintelligence.com.au/cs/documentation/examples

Alternatively,

We offer an end to end setup service, if you are interested in this; please email hello@parcelintelligence.com.au with a detailed description of your requirements.

1. CONTENTS

2. What is Bespoke Shipping?

Bespoke Shipping is a Shopify App designed to let you implement your own shipping rates, it is essentially a macro editor allowing you to implement whatever you want.

The App is focused on generating rates for your customers; it works with your existing themes and shipping estimator widgets seamlessly.

This App does not handle fulfilment after the order has been placed.

3. Is this App for me?

This App is intended for stores with bespoke shipping rate requirements which Shopify's default configuration options cannot provide. The App is a blank slate and while in the form of a macro editor, gives you full control in what business logic can be implemented. There is no GUI which restricts what features are available or not.

Since you are required to create your own macro, some programming experience in PHP (http://php.net) is required. Depending on how complex your rate calculations are, basic to intermediate knowledge of PHP is recommended.

See FAQs section for examples

Furthermore, this app requires the Real-time shipping plugin (Carrier Calculated Shipping) on your Shopify store (please note that this is an at-cost feature and is not included in the monthly fee of this App). This plugin comes FREE with the Advanced plan or higher, however can be added to any other plan, you can get this for free if you subscribe to Shopify annually (and get 10% off your plan!) or added to any plan for $20/month extra. Please contact your Shopify for either of these options.

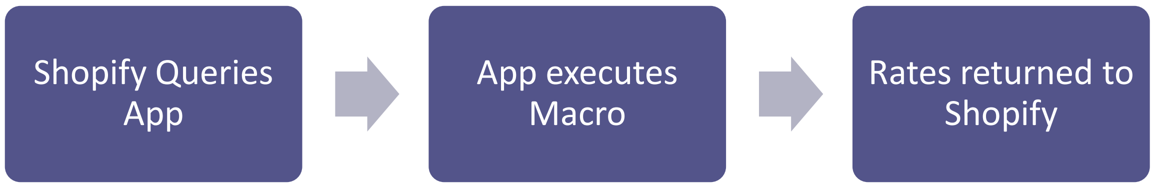
4. How does this App work?

Every time a customer places an order, or uses a shipping estimator, Shopify will query:

1. In-built shipping configuration
2. This App
3. Any other real-time shipping calculator Apps

Shopify will then aggregate all rates returned by all Apps, order them from cheapest to most expensive, then display them to your customer.

Every time Shopify queries this App, it sends the App your customers' cart details, such as destination details and quantity, price and weight of all items in the cart. You can use this information in your macro to generate a list of rates for your customer; this list is then sent back to Shopify.



4.1 The Macro Editor

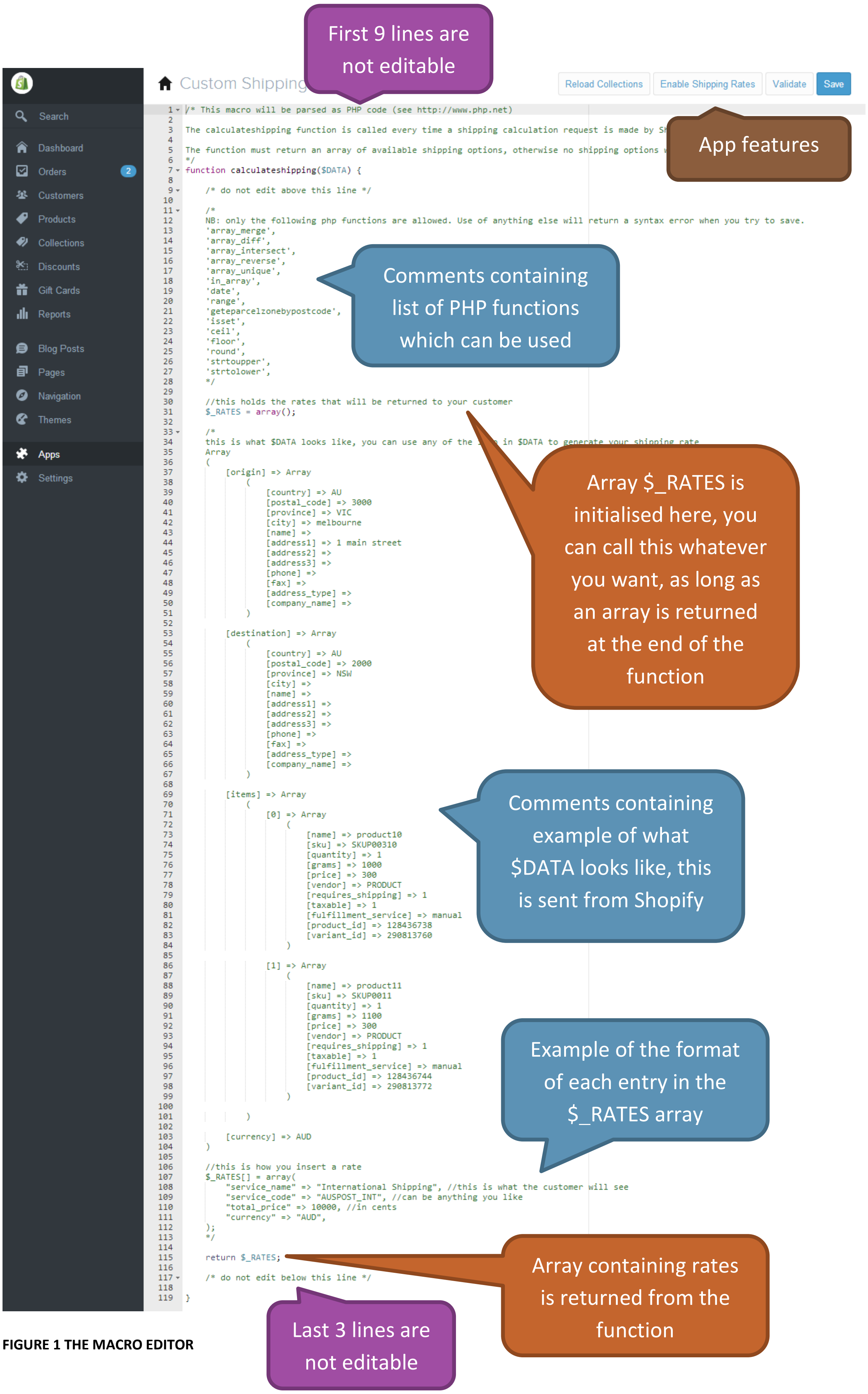
When you open up the App you are presented with a macro editor as depicted in Figure 1. The Macro is a PHP function called **calculateshipping** which is called by the App. On the function call, the data sent from Shopify is passed to the function in the variable **$DATA**. **$DATA** and Shopify Collections (more on this later) are the only dynamic data available to you for use in your macro. The function must return an array which contains your rates.

When you first install the App you are presented with a default template, the first 9 lines and last 3 lines are not editable, these correspond to the function wrapper syntax, if you manage to delete them, the entire configuration will reset back to the default template. The default configuration will output no rates to your customer and only includes inline comments.

To use the macro editor, implement your macro, then click the **Validate** button, this will check your macro for syntax errors only, it won't check for logic errors.

After clicking validate, you must click **Save** to save the macro. The App is also able to pull an array of productid’s from a collection (more on this later). If you use this feature, you must goto **Settings** and click **Reload Collections** to make the App cache your collection data. You must click this button every time your collection is updated.

You can also disable/enable this App anytime by going to **Settings** and clicking the **Enable Shipping Rates** button or **Disable Shipping Rates** button.


First 9 lines are not editable
Custom Shipping
Reload Collections
Enable Shipping Rates
Validate
Save
App features
Search
Dashboard
Orders 2
Customers
Products
Collections
Discounts
Gift Cards
Reports
Blog Posts
Pages
Navigation
Themes
Apps
Settings
Comments containing list of PHP functions which can be used
Array $_RATES is initialised here, you can call this whatever you want, as long as an array is returned at the end of the function
Comments containing example of what $DATA looks like, this is sent from Shopify
Example of the format of each entry in the $_RATES array
Array containing rates is returned from the function
Last 3 lines are not editable

```
/* This macro will be parsed as PHP code (see http://www.php.net)

The calculateshipping function is called every time a shipping calculation request is made by Sh

The function must return an array of available shipping options, otherwise no shipping options w
*/
function calculateshipping($DATA) {

    /* do not edit above this line */

    /*
    NB: only the following php functions are allowed. Use of anything else will return a syntax error when you try to save.
    'array_merge',
    'array_diff',
    'array_intersect',
    'array_reverse',
    'array_unique',
    'in_array',
    'date',
    'range',
    'geteparcelzonebypostcode',
    'isset',
    'ceil',
    'floor',
    'round',
    'strtoupper',
    'strtolower',
    */

    //this holds the rates that will be returned to your customer
    $_RATES = array();

    /*
    this is what $DATA looks like, you can use any of the [illegible] in $DATA to generate your shipping rate
    Array
    (
        [origin] => Array
            (
                [country] => AU
                [postal_code] => 3000
                [province] => VIC
                [city] => melbourne
                [name] =>
                [address1] => 1 main street
                [address2] =>
                [address3] =>
                [phone] =>
                [fax] =>
                [address_type] =>
                [company_name] =>
            )

        [destination] => Array
            (
                [country] => AU
                [postal_code] => 2000
                [province] => NSW
                [city] =>
                [name] =>
                [address1] =>
                [address2] =>
                [address3] =>
                [phone] =>
                [fax] =>
                [address_type] =>
                [company_name] =>
            )

        [items] => Array
            (
                [0] => Array
                    (
                        [name] => product10
                        [sku] => SKUP00310
                        [quantity] => 1
                        [grams] => 1000
                        [price] => 300
                        [vendor] => PRODUCT
                        [requires_shipping] => 1
                        [taxable] => 1
                        [fulfillment_service] => manual
                        [product_id] => 128436738
                        [variant_id] => 290813760
                    )

                [1] => Array
                    (
                        [name] => product11
                        [sku] => SKUP0011
                        [quantity] => 1
                        [grams] => 1100
                        [price] => 300
                        [vendor] => PRODUCT
                        [requires_shipping] => 1
                        [taxable] => 1
                        [fulfillment_service] => manual
                        [product_id] => 128436744
                        [variant_id] => 290813772
                    )

            )

        [currency] => AUD
    )

    //this is how you insert a rate
    $_RATES[] = array(
        "service_name" => "International Shipping", //this is what the customer will see
        "service_code" => "AUSPOST_INT", //can be anything you like
        "total_price" => 10000, //in cents
        "currency" => "AUD",
    );
    */

    return $_RATES;

    /* do not edit below this line */

}
```

FIGURE 1 THE MACRO EDITOR

4.2 Valid PHP functions

The App only allows certain PHP functions which have been whitelisted for use to protect you and other users. If you would like to use a function not in the following list, please email us at hello@parcelintelligence.com.au and discuss your requirements.

Valid PHP functions Please consult php.net for function definitions		
add	explode	round
array_chunk	floatval	rsort
array_column	floor	rtrim
array_combine	gmdate	sort
array_diff	hex2bin	str_pad
array_filter	implode	str_replace
array_flip	in_array	stripos
array_intersect	intval	strlen
array_key_exists	is_array	strpos
array_keys	isset	strtolower
array_map	json_decode	strtotime
array_merge	json_encode	strtoupper
array_rand	krsort	substr
array_reduce	ksort	time
array_reverse	ltrim	trim
array_search	max	uasort
array_shift	min	ucwords
array_unique	number_format	usort
arsort	pow	var_dump
asort	preg_grep	
ceil	preg_match	
count	preg_match_all	
date	preg_replace	
date_default_timezone_set	print_r	
date_diff	range	

Valid custom functions available to your macro
These functions are specific to Bespoke Shipping and are here to help you simplify your code

removeaccents($str)

Function removes accents on certain letters, i.e. à becomes a
This is useful when you want to match certain international province names where customers may enter accented characters, use this function to conform all input to the same format.

input: string
output: string
example:

```
echo removeaccents('àpple');
```

```
outputs
apple
```

getcollection($collectionid,$donotcache=false)

Function retrieves a list of productids from a given collection. When $donotcache = true, Bespoke will fetch the items from Shopify on the fly. If you use lots of collections this is not recommended as this will take a long time to fetch all the products for every collection. It is recommended you use tags of product types instead.

WARNING: If you set $donotcache to true, be aware that this will be subject to Shopify API call limits and response times. If the Shopify API takes a long time to respond then you will see no rates, it's advised to not use this feature unless absolutely necessary.

input: integer
output: array of integer
example:

```
print_r(getcollection(133264457));
```

```
outputs
Array
(
     [0] => 296582573
     [1] => 296582599
     [2] => 296582635
)
```

getCurrencyExchange()

This function fetches the currency exchange list from Shopify (https://cdn.shopify.com/s/javascripts/currencies.js). If you want to simply convert currency, use the convertCurrency() function

input: none
output: array of exchange rates compared to USD keyed by currency code
example:

```
print_r(getCurrencyExchange());
```

```
outputs
```

```
Array
(
    [USD] => 1
    [EUR] => 1.06732
    [GBP] => 1.21624
    [CAD] => 0.742528
    [ARS] => 0.064683
    ...
    [XBT] => 1155.5
    [NTD] => 0.0337206
    [BYN] => 0.52648
)
```

convertCurrency ($value,$fromCurrency,$toCurrency,$rates=null)

This function uses getCurrencyExchange() to fetch the exchange rates automatically then converts the value of currency. If you are going to call this function multiple times, it is recommended you call getCurrencyExchange() once and pass the results in as $rates to prevent multiple calls out to Shopify which will slow down your rate generation

input: float, string, string, array

output: the converted rate

example:

```
var_dump(convertCurrency(1.0, 'USD', 'CAD'));
```

outputs

```
float(1.3467505602482)
```

example for multiple calls to this function:

```
$xe = getCurrencyExchange();
var_dump(convertCurrency(1.0, 'USD', 'CAD',$xe));
var_dump(convertCurrency(1.0, 'USD', 'GBP',$xe));
var_dump(convertCurrency(1.0, 'USD', 'AUD',$xe));
```

outputs

```
float(1.3467505602482)
float(0.8222061435243)
float(1.3252932211252)
```

enrichProductDetails($DATA)

This function retrieves additional details regarding each line item such as inventory levels, product type, product tags, product handles and variant options. Call this function at the start of the config to ensure $DATA has been populated with the enriched fields before using it.

WARNING: This function will automatically cache product details for up to an hour. Therefore when you are using variables such as inventory quantity, these values may be up to an hour behind.

input: $DATA

output: the $DATA array with enriched fields

example:

```
$DATA = enrichProductDetails($DATA);
```

After calling enrichProductDetails(), each line item will have the additional properties available

- handle (string)
- tags (array of string)
- product_type (string)
- position (integer)
- option1 (string)
- option2 (string)
- option3 (string)
- barcode (string)
- inventory_policy (string)
- inventory_management (string)
- inventory_quantity (integer)
- weight (float)
- weight_unit (string)

New fields highlighted

example:

```
print_r($DATA);
```

outputs

```
Array
(
    [origin] => Array
        (
            [country] => US
            [postal_code] => 91210
            [province] => CA
            [city] => Glendale
            [name] =>
            [address1] => 1 Main St
            [address2] =>
            [address3] =>
            [phone] => 0000000
            [fax] =>
            [address_type] =>
            [company_name] => ACME Co
        )

    [destination] => Array
        (
            [country] => US
            [postal_code] => 91210
            [province] => CA
            [city] =>
            [name] =>
            [address1] =>
            [address2] =>
            [address3] =>
            [phone] =>
            [fax] =>
            [address_type] =>
            [company_name] =>
        )
    [items] => Array
        (
            [0] => Array
                (
                    [name] => ACME Product A
                    [sku] => SKU00A
                    [quantity] => 1
                    [grams] => 227
                    [price] => 40000
                    [vendor] => ACME
                    [requires_shipping] => 1
                    [taxable] => 1
                    [fulfillment_service] => manual
                    [properties] => Array
                        (
                        )

                    [product_id] => 91994962205
                    [variant_id] => 32978413005
                    [handle] => acme-product-a
                    [tags] => Array
                        (
                            [0] => acme-essentials
                            [1] => sale-items
                        )

                    [product_type] => Essentials
                    [position] => 1
                    [option1] => Black
                    [option2] => Small
                    [option3] =>
                    [barcode] => BARCODEA
                    [inventory_policy] => continue
                    [inventory_management] => shopify
                    [inventory_quantity] => 2
                    [weight] => 0.5
                    [weight_unit] => lb
                )
        )
    [currency] => USD
)
```

enrichCartDetails($DATA)

This function retrieves additional details regarding cart details. The app performs a lookup of abandoned checkouts and matches based on address and items in the cart.
All attributes in Shopify's abandoned checkout API are returned with this call, which will provide you with customer tags, attributes and additional information about the cart such as discount coupons.
Call this function at the start of the config to ensure $DATA has been populated with the enriched fields before using it.

WARNING: As this function looks up abandoned checkouts, this means customers must have gone through the Shopify checkout and entered their contact details before an abandoned checkout is created, therefore if customers have just added an item to the cart and use the shipping estimator widget on the cart page (which comes with certain themes) then no cart information will be available at this point in time. You will need to cater for the scenario that there is no cart data.

This feature ONLY works with Shopify's checkout system. If you use a 3rd party app to process

your checkouts then Bespoke won't be able to retrieve your cart details.

input: $DATA
output: the $DATA array with enriched fields
example:

```
$DATA = enrichCartDetails($DATA);
```

After calling enrichCarttDetails(), $DATA['cart '] will be populated with an array if successful, otherwise null

New fields highlighted

example:

```
print_r($DATA);
```

outputs

```
Array
(
    [origin] => Array
        (
            [country] => US
            [postal_code] => 10001
            [province] => NY
            [city] => new york
            [name] =>
            [address1] => 1 Main St
            [address2] =>
            [address3] =>
            [phone] => 000000000
            [fax] =>
            [email] =>
            [address_type] =>
            [company_name] => ACME Co
        )

    [destination] => Array
        (
            [country] => US
            [postal_code] => 91210
            [province] => CA
            [city] =>
            [name] =>
            [address1] => 1 main st
            [address2] =>
            [address3] =>
            [phone] =>
            [fax] =>
            [email] =>
            [address_type] =>
            [company_name] =>
        )

    [items] => Array
        (
            [0] => Array
                (
                    [name] => ACME Product A
                    [sku] => SKU00A
                    [quantity] => 1
                    [grams] => 500
                    [price] => 100
                    [vendor] => ACME
                    [requires_shipping] => 1
                    [taxable] =>
                    [fulfillment_service] => manual
                    [properties] => Array
                        (
                        )

                    [product_id] => 126870280
                    [variant_id] => 287840882
                )

        )
    [currency] => USD
    [locale] => en
    [cart] => Array
        (
            [id] => 1488406675509
            [token] => b3c28d0eaed2aed87548b780a7286692
            [cart_token] => 9a28f4db6a1252a4e44e7f7d50d7495b
            [email] => test@test.com
            [gateway] =>
            [buyer_accepts_marketing] => 1
            [created_at] => 2018-07-01T23:19:22-04:00
            [updated_at] => 2018-07-02T07:35:58-04:00
            [landing_site] => /admin
            [note] =>
            [note_attributes] => Array
                (
                )

            [referring_site] =>
            [shipping_lines] => Array
                (
                    [0] => Array
                        (
                            [code] => Local Pickup
                            [price] => 0.00
                            [source] => shopify
                            [title] => Local Pickup
                            [phone] =>
                            [tax_lines] => Array
                                (
                                )

                            [custom_tax_lines] =>
                            [validation_context] =>
                            [markup] => 0.00
                            [delivery_category] =>
                            [carrier_identifier] =>
                            [carrier_service_id] =>
                            [api_client_id] =>
                            [requested_fulfillment_service_id] =>
                            [applied_discounts] => Array
                                (
                                )

                            [id] => 60ddc5044e72b2c34031716993c0078e
                        )

                )

            [subtotal_price] => 0.90
            [taxes_included] => 1
            [total_discounts] => 0.10
            [total_line_items_price] => 1.00
            [total_price] => 0.90
            [total_tax] => 0.00
            [total_weight] => 500
            [currency] => USD
            [completed_at] =>
            [closed_at] =>
            [user_id] =>
            [location_id] =>
            [source_identifier] =>
            [source_url] =>
            [device_id] =>
            [phone] =>
            [customer_locale] => en
            [line_items] => Array
                (
                    [0] => Array
                        (
                            [applied_discounts] => Array
                                (
                                )

                            [key] => 3da1f472b18cb275c554db0a45d88151
                            [compare_at_price] => 2.00
                            [destination_location_id] => 484795777077
                            [fulfillment_service] => manual
                            [gift_card] =>
                            [grams] => 500
                            [line_price] => 1.00
                            [origin_location_id] => 335260909621
                            [price] => 1.00
                            [product_id] => 126870280
                            [properties] => Array
                                (
                                )

                            [quantity] => 1
                            [requires_shipping] => 1
                            [sku] => SKU00A
                            [tax_lines] => Array
                                (
                                )

                            [taxable] =>
                            [title] => ACME Product A
                            [variant_id] => 287840882
                            [variant_title] => DEFAULT
                            [vendor] => ACME
                        )

                )

            [name] => #1488406675509
            [source] =>
            [discount_codes] => Array
                (
                    [0] => Array
                        (
                            [code] => 10OFF
                            [amount] => 0.10
                            [type] => percentage
                        )

                )

            [abandoned_checkout_url]                                               =>
https://yourstore.myshopify.com/999999/checkouts/b3c28d0eaed2aed87548b780a7286692/rec
over?key=e62345dcfdea1c5d80d773771fde555d
            [tax_lines] => Array
                (
                )

            [source_name] => web
            [shipping_address] => Array
```

```
                (
                    [first_name] => test
                    [address1] => 1 main st
                    [phone] =>
                    [city] => Glendale
                    [zip] => 91210
                    [province] => California
                    [country] => Australia
                    [last_name] => test
                    [address2] =>
                    [company] =>
                    [latitude] => 34.052235
                    [longitude] => -118.243683
                    [name] => test test
                    [country_code] => US
                    [province_code] => CA
                )

            [customer] => Array
                (
                    [id] => 117364530
                    [email] => test@test.com
                    [accepts_marketing] => 1
                    [created_at] => 2013-03-20T09:17:54-04:00
                    [updated_at] => 2018-07-02T23:19:41-04:00
                    [first_name] => test
                    [last_name] => test
                    [orders_count] => 4
                    [state] => disabled
                    [total_spent] => 0.00
                    [last_order_id] => 111166521370
                    [note] =>
                    [verified_email] => 1
                    [multipass_identifier] =>
                    [tax_exempt] =>
                    [phone] =>
                    [tags] => Array
                        (
                        )

                    [last_order_name] => XXXAU-1062
                    [admin_graphql_api_id] => gid://shopify/Customer/117364530
                    [default_address] => Array
                        (
                            [id] => 118441967642
                            [customer_id] => 117364530
                            [first_name] => test
                            [last_name] => test
                            [company] =>
                            [address1] => test
                            [address2] =>
                            [city] => test
                            [province] => California
                            [country] => United States
                            [zip] => 91210
                            [phone] =>
                            [name] => test test
                            [province_code] => CA
                            [country_code] => US
                            [country_name] => United States
                            [default] => 1
                        )

                )

        )

)
```

4.3 FUNCTION INPUT

The **calculateshipping** function has one input, an array called **$DATA**. **$DATA** contains information sent by Shopify.

$DATA has the following structure

```
Array
       (
              [origin] => Array
                     (
                            [country] => AU
                            [postal_code] => 3000
                            [province] => VIC
                            [city] => melbourne
                            [name] =>
                            [address1] => 1 main street
                            [address2] =>
                            [address3] =>
                            [phone] =>
                            [fax] =>
                            [address_type] =>
                            [company_name] =>
                     )

              [destination] => Array
                     (
                            [country] => AU
                            [postal_code] => 2000
                            [province] => NSW
                            [city] =>
                            [name] =>
                            [address1] =>
                            [address2] =>
                            [address3] =>
                            [phone] =>
                            [fax] =>
                            [address_type] =>
                            [company_name] =>
                     )

              [items] => Array
                     (
                            [0] => Array
                                   (
                                          [name] => product10
                                          [sku] => SKUP00310
                                          [quantity] => 1
                                          [grams] => 1000
                                          [price] => 300
                                          [vendor] => PRODUCT
                                          [requires_shipping] => 1
                                          [taxable] => 1
                                          [fulfillment_service] => manual
                                          [product_id] => 128436738
                                          [variant_id] => 290813760
                                   )
                            [1] => Array
                                   (
                                          [name] => product11
                                          [sku] => SKUP0011
                                          [quantity] => 1
                                          [grams] => 1100
                                          [price] => 300
                                          [vendor] => PRODUCT
                                          [requires_shipping] => 1
                                          [taxable] => 1
                                          [fulfillment_service] => manual
                                          [product_id] => 128436744
                                          [variant_id] => 290813772
                                   )

                      )

               [currency] => AUD
        )
```

Example:

To reference the destination postcode, use **$DATA['destination']['postal_code']**

To referencing the currency, use **$DATA['currency']**

$DATA can be enriched with additional fields, see the custom function enrichProductDetails().

4.4 Function output

The **calculateshipping** function must return an array, if the array is empty, no shipping rates from this App will be show to the customer, if the function does not return an error, or contains syntax errors, Shopify will ignore output from this App.

The output array must contain a list of rates, where each rate is also represented in the form of an array.

A **single** rate has the following structure

```
Array
(
    [service_name] => International Shipping
    [service_code] => AUSPOST_INT
    [description] => Description
    [total_price] => 10000
    [currency] => AUD
)
```

The parameters have the following meanings

Parameter	Definition
service_name	The text the customer sees
service_code	Internal code for this rate, the customer won't see this
total_price	The price in cents
currency	The 3 letter currency code, i.e. USD, CAD, AUD
description	A description which shows on the checkout page only

Example:

Return a Pickup option for two locations to all customers

```
//initialise the return array
$_RATES = array();

//insert pickup location A
$_RATES[] = array(
             "service_name" => "Pickup location A",
             "service_code" => "PICKUPA",
             "description" => "1 Main st",
             "total_price" => 0,
             "currency" => "AUD",
);

//insert pickup location B
$_RATES[] = array(
             "service_name" => "Pickup location B",
             "service_code" => "PICKUPB",
             "description" => "2 Main st",
              "total_price" => 0,
              "currency" => "AUD",
);

return $_RATES;
```

4.5 Collections

Collections can be used to organise products into groups. For example if you have items which attract special shipping rates, they can be added into a hidden collection in order for the App to differentiate them.

NB: collections can be hidden

A custom PHP function has been created to allow the app to read collections:

```
getcollection($collectionid,$donotcache=false)
```

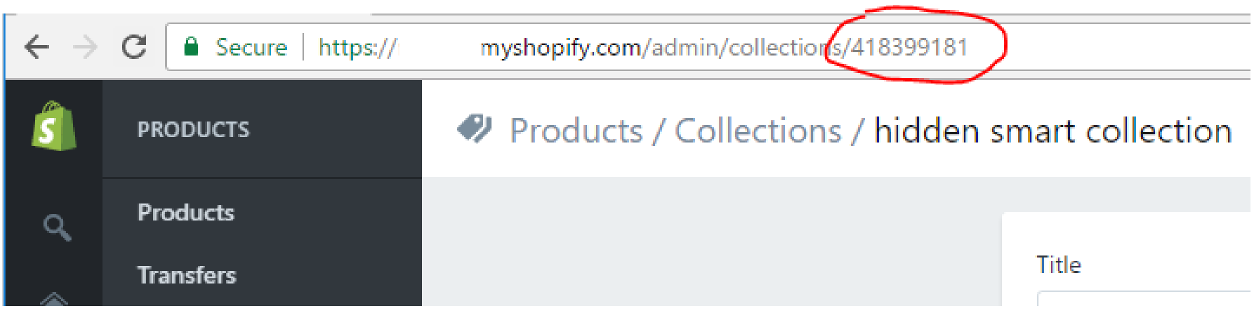
This function accepts one mandatory parameter, which is the collection id, and returns a list of productids. There is also an optional parameter to override the cache.

If you use the getcollection() function in your macro, you choose to have bespoke cache the products in the collection or fetch it on the fly from Shopify, caching products works well when your products do not change often and you want your rates to be returned quickly, if you have lots of collections it is recommended you **cache** your collections.

To cache a collection, goto **Settings** and click on the **Reload Collections** button before or after you save your config. This will instruct the App to cache a copy of products in each of the collections used. If you add or remove products to your collection, you must click **Reload Collections** again for changes to be reflects.

To obtain a collectionid:

1. log into your Shopify admin page and click on **collections**
2. click on the collection
3. read the numbers off the end of the URL


Secure
https:// myshopify.com/admin/collections/418399181
PRODUCTS
Products
Transfers
Products / Collections / hidden smart collection
Title

4. In this example, the collection id is **418399181**

Example:

A store had a group of special items, the following example shows how to count the number of special items in the order and the number of non-special items. In this example, the special items have been added into a collection with a collection id of 26115365.

```
$specialItems = getcollection(26115365);

$countSpecialItems = 0;
$countNormalItmes = 0;
foreach ($DATA['items'] as $item) {
      if (in_array($item['product_id'],$specialItems)) {
            $countSpecialItems += $item['quantity'];
      } else {
            $countNormalItmes += $item['quantity'];
      }
}
```

5. API FOR EXTERNAL APPLICATIONS

Bespoke offers an API for 3rd party apps to integrate. You will need your API key which can be found by clicking **Settings**.

End-point: http://parcel-intelligence.com/cs/api_ex.php

POST to this end point with the json payload as per below (this is the same as Shopify's carrier service API definition)

You also need to set the following headers (headers are case sensitive)

HTTP_X_STORE_NAME = xxxx.myshopify.com

HTTP_X_STORE_TOKEN = Your API Key

Example request:

```
{
   "origin":{
      "country":"CA",
      "postal_code":"L5M 0N1",
      "province":"ON",
      "city":"Mississauga",
      "name":null,
      "address1":"Main Avenue",
      "address2":"",
      "address3":null,
      "phone":"",
      "fax":null,
      "email":null,
      "address_type":null,
      "company_name":"ACME CO"
   },
   "destination":{
      "country":"CA",
      "postal_code":"L5M5C7",
      "province":"ON",
      "city":"Mississauga",
      "name":"John Doe",
      "address1":"test",
      "address2":"",
      "address3":null,
      "phone":null,
      "fax":null,
      "email":null,
      "address_type":null,
      "company_name":null
   },
   "items":[
      {
         "name":"Astounding Product",
         "sku":"SKU-87067-3",
         "quantity":1,
         "grams":15467,
         "price":24362,
         "vendor":"Vendor A ",
         "requires_shipping":true,
         "taxable":true,
         "fulfillment_service":"manual",
         "properties":null,
         "product_id":11608837318,
         "variant_id":49281384134
      },
      {
         "name":"Long Lasting And Durable Product",
         "sku":"27804",
         "quantity":2,
         "grams":56245,
         "price":55261,
         "vendor":"Acme co",
         "requires_shipping":true,
         "taxable":true,
         "fulfillment_service":"manual",
         "properties":null,
         "product_id":11619411334,
         "variant_id":49383258438
      }
   ],
   "currency":"CAD",
   "locale":"en"
}
```

Example response:

```
{
   "rates":[
      {
         "service_name":"Bespoke Test Rate",
         "service_code":"STANDARD_SHIPPING",
         "total_price":10000,
         "currency":"CAD"
      }
   ]
}
```

6. FAQs

1. **ARE THERE ANY CODE SAMPLES?**

   Examples can be found in the link below. They are provided as is, you may need to adapt it to suite your own needs. If you need assistance writing the config, please contact us at hello@parcelintelligence.com.au and we can provide a quote for end to end setup.

   http://parcelintelligence.com.au/cs/documentation/examples/

2. **HOW DO I GET THE CART PRICE TOTAL**

   The following snippet of code will calculate the cart price total.

```
$totalPrice = 0; //in dollars
foreach ($DATA['items'] as $item) {
  $totalPrice += $item['quantity']*$item['price']/100;
}
```

3. **HOW CAN I GET THE CART WEIGHT TOTAL**

   The following snippet of code will calculate the cart weight total; it will also skip over items which do not require shipping.

```
$totalWeightInKG = 0; //in KG
$totalWeightInLB = 0; //in LB
foreach ($DATA['items'] as $item) {
  $totalWeightInKG += $item['quantity']*$item['grams']/1000;
  $totalWeightInLB += $item['quantity']*round($item['grams']/1000*2.20462,2);
}
```

4. **I'VE SETUP WEIGHT RANGES, BUT ALL WHEN TESTING, THE LARGEST RANGE IS ALWAYS USED, WHY DOES IT NOT WORK?**

   Shopify sends all product weight in grams, make sure your code uses grams as well, or the weight total you have calculated is converted into kg.

5. **MY RATES SHOW 100 TIMES LESS THAN WHAT IT IS SUPPOSED TO BE, WHY IS THAT?**

   Shopify requires all rates to be sent as cents. Make sure your units are correct.

6. **CAN I USE COLLECTIONS?**

   Yes you can. Use the function getcollection() to grab a list of productids for a given collection. Refer to section 4.5 for more details.

7. How do I find my collection id?

Your Shopify collection id can be found by visiting your Shopify Admin page, then going to the collections page, then clicking on the collection. The collection id is found at the end of the URL. Refer to section 4.5 for more details.

8. Can I use product tags?

Yes, call enrichProductDetails() custom function at the start of the config.

9. Can I use product types?

Yes, call enrichProductDetails() custom function at the start of the config.

10. Can I use customer tags?

Yes, call enrichCartDetails(), the following snippet will return the customer tags to the $customerTags variable (in the form of an array)

```
$DATA = enrichCartDetails($DATA);

$customerTags = array();
if (isset($DATA['cart']['customer']['tags'])) {
  $customerTags = $DATA['cart']['customer']['tags'];
}
```

11. How can I order the list of rates my customer sees?

Bespoke offers a solution to re-order your rates, Contact us for more details!

12. I don't know PHP, Can someone write/maintain my configuration for me?

Of course! We're here to help! We offer competitive hourly rates for development work, we're also happy to give you some free pointers to get you started. Email us at hello@parcelintelligence.com.au to discuss!

You are also free to find your own developer with PHP knowhow to implement for you.

13. Do you have any redundancy on your servers?

We understand that your business is important to you and that uptime is important, the Bespoke app is delivered within an enterprise grade environment, we have multiple servers handling your requests as well as redundancy and backups.

14. What version of PHP do you run?

We run PHP 5.6 on our servers.

15. **How do I turn off logging?**
Click on **Settings** then disable logging. If logs are disabled, we will not be able to assist with any debugging enquiries.

7. App Integrations

Bespoke has direct integrations with the following Shopify Apps, this allows you to use Bespoke seamlessly with these app with peace of mind knowing there won't be any conflicts.

7.1 Zapiet Store Pickup + Delivery

Fully featured, customizable and ideal whether you have one location or thousands; Zapiet is Shopify's most popular pickup and local delivery app, powering stores all over the world.

Integration with Bespoke means your shipping rates only show when customers select the relevant delivery option i.e. ***shipping*** or ***local delivery***. Additionally, you can use Bespoke to customise different rates for both ***shipping*** and ***local delivery*** options.

Link to app: https://apps.shopify.com/click-and-collect

7.2 Zapiet Delivery Rates by Distance

Bespoke's integration with the Zapiet Delivery Rates by Distance app enables distance calculations within Bespoke, driven by Zapiet's distance calculation engine, calculate exact distances between two addresses either by line of sight or driving distance.

Example usage:
https://parcelintelligence.com.au/cs/documentation/examples/delivery%20by%20distance.txt

Link to app: https://apps.shopify.com/delivery-rate-by-distance

7.3 Bold Cashier

Cashier is a feature-rich global checkout solution designed to help your business scale. Create a flawless shopping experience for your customers with advanced features such as Upsell after checkout, stored credit card accounts, the ability to sell in 150+ currencies, and much more. Best of all, the high converting one-page checkout can be fully customized to match your branding, complete with custom URL and design.

Link to app: http://affiliate.boldcommerce.com/apps/cashier?ref=pm7

7.4 Custom API integration

Bespoke also features its own restful API you can use to obtain shipping rates via our rating engine, if you have customised tools, you can use the Bespoke API in order to obtain shipping rates.

The payload is similar to Shopify's carrier service API (see: https://help.shopify.com/api/reference/shipping_and_fulfillment/carrierservice)

API endpoint: https://parcel-intelligence.com/cs/api_ex.php

Headers (headers are case sensitive):

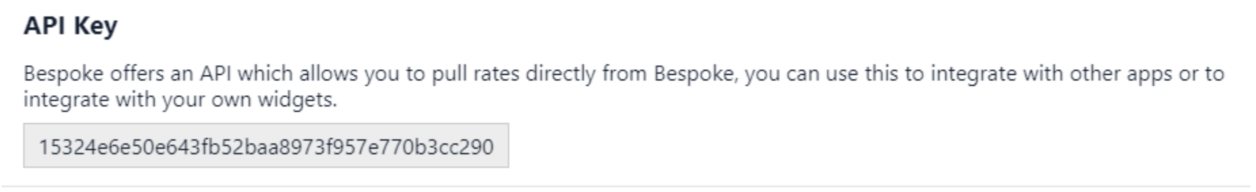
Header	**Value**	**Comments**
Content-type	application/json	Mandatory
HTTP_X_STORE_NAME	xxxxx.myshopify.com	Mandatory, this is your store name in the format xxx.myshopify.com
HTTP_X_STORE_TOKEN	xxxxxxxxxxx	Mandatory, this is your API key
HTTP_X_SHOW_DEBUG	True	Optional, when set, this will return the output of echo, var_dump and print_r functions within your config. Omit if you do not want to see these, also Omit in production, otherwise the json output will not parse correctly.

The response and payload is in JSON

Example request

```
{"origin":{"country":"US","postal_code":"91210","province":"CA","city":"Glendale"
,"name":null,"address1":"1 main
st","address2":"","address3":null,"phone":"","fax":null,"address_type":null,"comp
any_name":"Bespoke
Shipping"},"destination":{"country":"US","postal_code":"10001","province":"NY","c
ity":"New York","name":"John Doe","address1":"123 Main
St","address2":"","address3":null,"phone":"9876543210","fax":null,"address_type":
null,"company_name":""},"items":[{"name":"Product A -
250gr","sku":"","quantity":20,"grams":300,"price":1400,"vendor":"ACME","requires_
shipping":true,"taxable":true,"fulfillment_service":"manual","properties":null,"p
roduct_id":9062306182,"variant_id":33082410182}],"currency":"USD"}
```

You can get your API key by going to admin > apps > bespoke shipping > settings (scroll to bottom)


API Key

Bespoke offers an API which allows you to pull rates directly from Bespoke, you can use this to integrate with other apps or to integrate with your own widgets.

15324e6e50e643fb52baa8973f957e770b3cc290

8. LIMITATIONS

There are certain things this App cannot do, your macro must use the information made available to you in the **$DATA** array passed into the macro function, and Shopify collections. If you require any other piece of information, it must be hard coded into the actual macro or it cannot be done.

Some examples of what cannot be achieved.

Shopify Cache

Shopify caches all rates returned by all shipping apps. In some cases where a customer has used the back button and updated the items in their cart or made other changes, Shopify may not retrieve fresh rates from Bespoke and opt to return a set of old rates. This means your customers may not see the most up-to-date rates. Unfortunately this is a quirk of Shopify’s checkout system and there is no workaround for this issue.

Execution limit

Shopify imposes a 15second timeout on all Shipping apps, this means your code within Bespoke must finish executing and return the output to Shopify within this timeframe. In most cases this is not an issue, however when making multiple calls to 3rd party APIs this may cause delays. Unfortunately the performance of 3rd party APIs cannot be controlled by Bespoke, it is advisable for you to test your rates thoroughly and if you find your carrier's API takes a long time to respond, you should raise the issue with your carrier's technical support team directly.

Discount coupons

Discount coupons can be obtained using the enrichCartDetails() function, however as Shopify caches rates returned by Bespoke, when customers add/remove/change coupons on the cart, this may not trigger Shopify to reload their rates, therefore you should not rely on discount coupon data being reliable.

3rd Party checkout systems

Bespoke is designed to operate with the built in Shopify checkout, it may also work with 3rd party checkout systems, however certain functionality may be lost when used with 3rd party checkouts. Specifically, the following will not work unless the built in Shopify checkout is used:

- Discount coupon and customer tag detection (also does not work with Shopify draft orders)
- Currency selection on shipping rates does not work with Bold Cashier specifically, all rates will be returned as per the store currency rate regardless what the currency is set to within Bespoke. Please contact support at hello@parcelintelligence.com.au for alternative options.

Calls to 3rd party APIs

Bespoke has built in connectors to major shipping carriers, see the App store listing for the most up-to-date list.

https://apps.shopify.com/custom-shipping-rates

If your carrier is not listed here, contact us to discuss options for integrating it!

9. Uninstalling the App

For any reason, should you need to uninstall the App, login to your Shopify admin control panel, visit the Apps section to uninstall the App. Your subscription will be automatically cancelled, however no pro-rata refunds will be provided. Uninstalling the App will also mean your configuration is lost forever, all configuration data related to your store is deleted immediately on uninstall.