# Avalara AvaTax for Salesforce B2C Commerce

## SiteGenesis JavaScript Controllers (SGJC)

Version: v20.1.0


# Table of Contents

# 1.0. Summary

The AvaTax cartridge provides rapid integration for Salesforce B2C Commerce implementations. The AvaTax cartridge is a self-contained cartridge that can easily integrate into any project. This cartridge can be configured in the Business Manager and contains all elements necessary to perform a successful best practices implementation of AvaTax.

# 2.0. Component Overview

## 2.1. Functional Overview

1. AvaTax Update is a powerful, online sales tax compliance solution that provides all businesses with a level of sales tax automation that until now has been available only to Fortune 500 companies. AvaTax Update's transaction-based service model and seamless integration into business applications gives you rapid access to all jurisdiction assignments and real-time sales tax calculations. Cutting-edge technologies and superior processing logic can manage even the most complicated tax issues, such as situs, nexus, tax tiers, tax holidays, exemptions, certificate management and product taxability rules.
2. During check out shopping cart and customer information is transmitted via the AvaTax API web service. Tax information is sent back and applied to products in the cart.
3. Address Validation logic has been added to assist with filtering certain countries (configured in the custom site preferences) and to enable/disable the Address Validation calls.
4. Detailed transaction logging is available on the AvaTax Update dashboard.

## 2.2. Use Cases

This cartridge can be used to calculate taxes on the storefront through the AvaTax REST v2 API web service. It can also be used for address validation for all the US and Canada addresses.

Following features are provided as part of the cartridge:

1. Tax calculation
2. Address validation

## 2.3. Limitations, Constraints

As of now, only limitation is that AvaTax address validation is only available for US and Canada.

AvaTax currently only supports service messages in English.

## 2.4. Compatibility

This release is tested with SGJC 104.1.3 and works with compatibility mode 19.10.

## 2.5. Privacy, Payment

Customer addresses are sent to and from AvaTax for tax calculation and address verification. No customer payment information is accessed or stored.

# 3.0. Implementation Guide

## 3.1. Setup

### 3.1.1. Import Cartridges

Import `bm_avatax`, `int_avatax` and `int_avatax_svcclient` cartridges into the Eclipse UX Studio Workspace.

1. Open UX Studio in Eclipse IDE.
2. Go to **File > Import > General > Existing Projects into Workspace**
3. Browse to the directory where the cartridges have been downloaded.
4. Click **Finish**.
5. Click **OK,** when prompted to link the cartridges to the sandbox.
6. Modify the Site Path in **Business Manager > Administration > Manage Sites.** Make sure the cartridge names `'int_avatax'` and `'int_avatax_svcclient'` appear before any other cartridges.

If using Visual Studio code, use the below dw.json format to upload the cartridges to the sandbox. Place it at the root of the working directory.

```
{
  "hostname": "your-sandbox-hostname.demandware.net",
  "username": "yourlogin",
  "password": "yourpwd",
  "code-version": "version_to_upload_to"
}
```

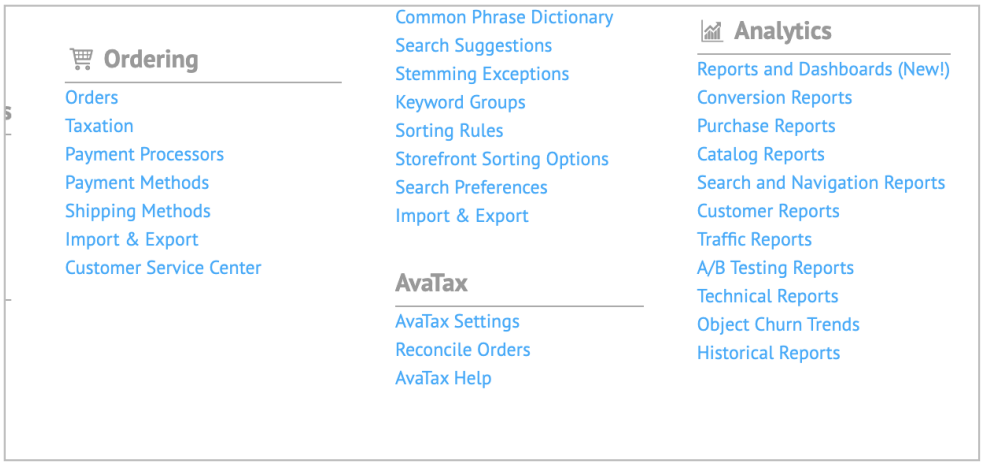
### 3.1.2. Import Metadata

1. Locate `site-import.zip` file inside **metadata** folder. Extract its content. Change site name if needed inside **sites** folder and compress the site-import folder to site-import.zip again.
2. Log in to the **Business Manager**.
3. Click **Administration > Site Development > Site Import & Export**.
4. Use the upload control to browse for the site-import.zip file located in the **metadata** folder.
5. Click **Upload**.
6. Select the site-import.zip and click **Import**. Import should complete successfully.

### 3.1.3. AvaTax Business Manager Extension Cartridge Setup

1. Make sure that the `'bm_avatax'` cartridge is imported in UX Studio workspace.
2. Add `'bm_avatax'` and `'int_avatax_svcclient'` cartridges in BM cartridge path by going to – **Administration > Sites > Manage Sites > Business Manager – Settings**
3. Make sure that the AvaTax BM module has been added to Business Manager modules and has been assigned necessary permissions as follows –

- Go to **Administration > Organization > Roles & Permissions**
- Select the Role to give access to. e.g. Administrator
- Go to **Business Manager Modules** section. Select the context for any or all of the sites.
- In the list of BM modules that are displayed, enable **AvaTax**. And, click **Update**.
- A new module **AvaTax** and its menu items should appear under **Merchant Tools**. Refer the below screenshot.




## 3.2. Configuration

### 3.2.1. Configure AvaTax Services

1. Make sure that the AvaTax services have been imported as part of site import step (3.1.2) performed above. Below service configurations should be present in the **Services** section.
2. Log in to the **Business Manager**.
3. Navigate to - **Administration > Operations > Services**
4. Configure your AvaTax credentials under **Credentials** tab, in **credentials.avatax.rest**.
5. Make sure that the service URL used is one of the following, depending on whether the sandbox is a production or development:
    - **Development**: `https://sandbox-rest.avatax.com`
    - **Production**: `https://rest.avatax.com`

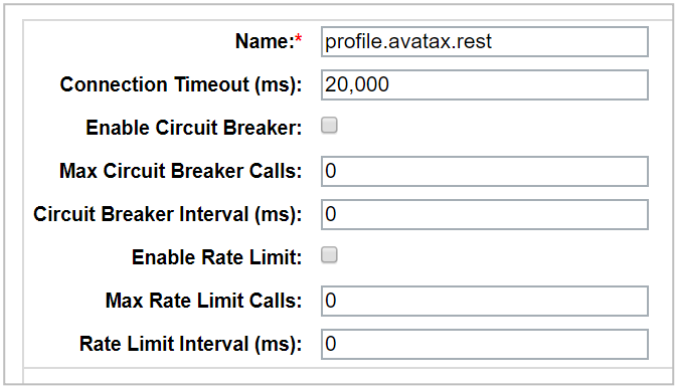
**Notes:**

- A service named `logentries.avatax.svc` should also have been created with corresponding Profile and Credential items. DO NOT modify or remove these.
- DO NOT modify service name(s) of any of the services.

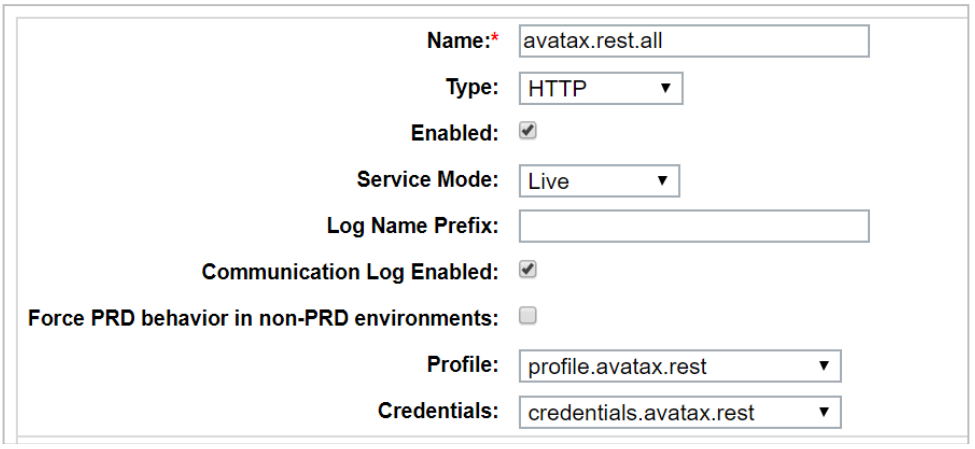
Below are the screenshots for reference.

**Credentials:**


**Profile:**




**Service:**




### 3.2.2. AvaTax Settings Configuration

Configure Avalara AvaTax related settings using the AvaTax module in the Business Manager.

1. Log into the **Business Manager**.
2. Select the desired site from the dropdown.
3. Navigate to **Merchant Tools > AvaTax > AvaTax Settings**
4. Select/fill all applicable settings for the storefront. Below is the description for these settings in brief.

**AvaTax Options:**

- **Tax Calculation** – When enabled, this will enable AvaTax tax calculation on all transactions for the current site.
- **Address Validation** – Not applicable for SGJC.
- **Save Transactions to AvaTax** – When enabled, this will save all orders to AvaTax. When disabled, orders will not be saved to AvaTax. However, the tax estimates appear on the storefront.
- **Commit Transactions to AvaTax** - If enabled, successful orders on the storefront will be marked 'Committed' when posting to Avalara AvaTax.
- **Company Code** – This is the AvaTax company code configured on the admin console of the AvaTax account. All successful orders placed on the current storefront will be posted under this company. Configurations such as Importer of Record, Sales Tax nexus of this company will be applicable for all transactions for the current storefront.
- **Customer code** - Select what value is sent as the customer code for authenticated customers. For guest customers, Email ID entered by them during checkout is used.
- **Shipping Tax Code** – If the shipping methods of the storefront do not have tax class assigned, this tax code will be used to calculate taxes for shipping methods. If not specified, AvaTax freight code 'FR' will be used.

**Cross Border Calculation**

- **Taxation Policy:** *(Read-only)* If cross border tax/customs duty calculation applies to the storefront, the taxation policy should be Net.

**Address Details**

- **Location Code** – If the Location Code value is provided, AvaTax prefers it over other address fields for tax calculation. Location code is configured under Avalara account as company location. If this value is present, the origin address fields can be omitted.
- **Origin (shipFrom) Address fields** – Details of the Company location from where the products will be shipped. These will be used for calculating taxes for all transactions on the current storefront.

  Address line 1, line 2, line 3, City, State, Zip/Postal Code, and Country Code should be filled out as per exact Company address to help calculate taxes accurately.

  If shipments are being shipped from US or Canada, fill in the complete and valid address details to avoid taxation errors.

**Note:**

- Navigate to **Merchant Tools > Site Preferences > Order > Order Access Settings**, and set **Limit Storefront Order Access** to **No.**
- Make sure to select the required Promotion Preference by Navigating to **Merchant Tools > Site Preferences > Promotion > Discount Taxation**.

## 3.3. Custom Code

The following files need code changes to be able to work seamlessly with AvaTax. Patch files have been generated with the necessary changes to avoid manual code changes.

1) app_storefront_controllers/cartridge/controllers/COPlaceOrder.js

2) app_storefront_controllers/cartridge/controllers/COShipping.js

3) app_storefront_controllers/cartridge/scripts/models/CartModel.js

4) app_storefront_core/cartridge/scripts/cart/calculate.js

### 3.3.1. Automatic Changes using diff/patch files

If you prefer manual code changes skip to section 3.3.2 after this.

Files which need code changes:
1) app_storefront_controllers/cartridge/controllers/COPlaceOrder.js (Patch file: COPlaceOrder.js.patch)

2) app_storefront_controllers/cartridge/controllers/COShipping.js (Patch file: COShipping.js.patch)

3) app_storefront_controllers/cartridge/scripts/models/CartModel.js (Patch file: CartModel.js.patch)

4) app_storefront_core/cartridge/scripts/cart/calculate.js (Patch file: calculate.js.patch)

Patch files are present in `/documentation/Sitegenesis Patches` folder.

On Unix/Mac, use below commands –

- $ patch calculate.js < calculate.js.patch
- $ patch CartModel.js < CartModel.js.patch
- $ patch COPlaceOrder.js < COPlaceOrder.js.patch
- $ patch COShipping.js < COShipping.js.patch.

How to consume patch files:

- https://www.shellhacks.com/create-patch-diff-command-linux/
- https://stackoverflow.com/questions/517257/how-do-i-apply-a-diff-patch-on-windows

### 3.3.2. Manual changes using IDE

1. In **CartModel.js** –

Add the following Golden coloured code in **calculate()** function just before the default tax hook.

```
calculate: function () {
  var avaconfig =
JSON.parse(require('dw/system/Site').getCurrent().getCustomPreferenceValue('ATSett
ings'));
  if (avaconfig.taxCalculation) {

require('int_avatax/cartridge/scripts/app').getController('Avatax').CalculateTaxes
(this.object);
  }
  dw.system.HookMgr.callHook('dw.ocapi.shop.basket.calculate', 'calculate',
this.object);
}
```

2. In the core cartridge, inside **calculate.js** –

Add the following line of code at the beginning of the file.

```
var avaconfig =
JSON.parse(require('dw/system/Site').getCurrent().getCustomPreferenceValue('ATSett
ings'));
```

Add the following if statement around **calculateTax()** to avoid calling it if Avatax is enabled:

```
// ================================================
    // =====          CALCULATE TAX                  =====
    // ================================================
if (!avaconfig.taxCalculation) {
        calculateTax(basket);
}
```

3. In **COShipping.js** controller, inside function **updateShippingMethodList**, add following lines of code –

```
// Transaction controls are for fine tuning the performance of the
database interactions when calculating shipping methods
    Transaction.begin();

    session.privacy.NoCall = true; // add this session variable

    for (i = 0; i < applicableShippingMethods.length; i++) {
        method = applicableShippingMethods[i];

        cart.updateShipmentShippingMethod(cart.getDefaultShipment().getID(),
method.getID(), method, applicableShippingMethods);
        cart.calculate();
        shippingCosts.put(method.getID(), cart.preCalculateShipping(method));
    }

    session.privacy.NoCall = false; // add this session variable

    Transaction.rollback();
```

4. In **COPlaceOrder.js** controller, inside **start()** function, add following code
   a. Add the following lines just before cart.calculate() transaction and after.

```
var OrderNo = OrderMgr.createOrderNo();

Transaction.wrap(function () {
       cart.calculate();
});

var basket = dw.order.BasketMgr.currentBasket;
```

   b. Find the following line of code inside start function.

```
var order = cart.createOrder();
```

Pass OrderNo parameter to createOrder() function, as below.

```
var order = cart.createOrder(OrderNo);
```

c. Add following lines of code inside `!orderPlacementStatus.error` if block.

```
if (!orderPlacementStatus.error) {
        // avalara code starts
        var avaconfig =
JSON.parse(require('dw/system/Site').getCurrent().getCustomPreferenceValue('ATSett
ings'));
        if (avaconfig.taxCalculation) {
            session.privacy.NoCall = false;
            session.privacy.finalCall = true;
            session.privacy.OrderNo = OrderNo;

require('int_avatax/cartridge/scripts/app').getController('Avatax').CalculateTaxes
(basket);
            session.privacy.OrderNo = null;
        }
        // avalara code end
        clearForms();
    }
```

## 3.4. Cross Border, VAT and Invoice messages

**NOTE**: This section can be skipped if the storefront doesn't ship products from one country to another or does not need to store customs duty or cross border details.

AvaTax service returns taxes and invoice messages based on origin and destination locations in different countries. If the customs duties are applicable, these values are also stored in custom attributes.

### Cross Border Tax Calculation:

The custom duties that are calculated based on origin and destination addresses are stored in custom attributes as follows. These can be utilized in a model to display on the storefront.

| Sr. No. | Value | Object |
|---|---|---|
| 1. | **Customs duty** | `basket.custom. ATCustomsDuty` |
| | | `order.custom.ATCustomsDuty` |
| 2. | **Taxes** | `basket.custom.ATTax` |
| | | `order.custom.ATTax` |
| 3. | **Cross border messages** | `basket.custom.ATLandedCost` |
| | | `order.custom.ATLandedCost` |
| 4. | **Transaction Summary** | `basket.custom.ATGenericMessage` |
| | | `order.custom. ATGenericMessage` |
| 5. | **Invoice Messages** | `basket.custom.ATInvoiceMessage` |
| | | `order.custom. ATInvoiceMessage` |
| 6. | **Tax Break-up (JSON)** | `basket.custom.ATTaxDetail` |
| | | `order.custom. ATTaxDetail` |

These values are stored as custom attributes in –

1. `Basket` object, if the order is still not placed.
2. `Order` object, after the current Basket is converted into an Order or the order has been placed.

### Invoice Messages

The invoice messages object is stored in below JSON format.

```
{"InvoiceMessageMasterList":[{"MessageCode":0,"Message":"No applicable messaging for this line."},{"MessageCode":1,"Message":"Intra-EU Supply of Goods as per Art. 138 EU VAT Directive 2006/112"}],"InvoiceMessageList":[{"TaxLineNo":"98ed5002d09d3b33abba8b10ed","MessageCode":1},{"TaxLineNo":"f9a79cafa05441b14e72c0e2a6","MessageCode":0}]}
```

The messages may be utilized to display to the site customer during checkout as required.

'InvoiceMessageMasterList' is an array of all applicable messages.

'MessageCode' is the index used to refer to the corresponding 'Message'.

'InvoiceMessageList' is an array of all line items in the Basket and their corresponding MessageCodes.

'TaxLineNo' indicates UUID of the line items in the current Basket object.

**For example,**

`{"TaxLineNo":"98ed5002d09d3b33abba8b10ed","MessageCode":1}` indicates that a line item in the current basket with UUID 98ed5002d09d3b33abba8b10ed has a VAT invoice message 'Intra-EU Supply of Goods as per Art. 138 EU VAT Directive 2006/112' which corresponds to `MessageCode` 1.

**Tax Breakdown details**

When taxes are calculated and returned by Avalara services, it may be displayed to the end user on the storefront.

To accommodate this, AvaTax integration stores the tax details for every line item (Product, Shipping, Gift certificate etc.) in the Basket or Order object in the form of a custom attribute based on whether the order is in placed state or not.

1. If the Basket object is still in checkout process, i.e. If the order is not yet placed, then the tax details are stored in a custom attribute of `basket` system object.

   The attribute name is - `ATTaxDetail`. It can be accessed as - `basket.custom.ATTaxDetail`

2. If the order is in placed state, the tax details are stored on `order` system object.

   The attribute name is - `ATTaxDetail`. It can be accessed as - `order.custom.ATTaxDetail`

The tax details are stored in the form of JSON string. It can be utilized as needed. Below is an example of tax details JSON for a Basket.


```
[
   {
      "lineitemid":"750518703299",
      "shipmentid":"me",
      "taxes":[
         {
            "jurisdictiontype":"State",
            "jurisdiction":"WASHINGTON",
            "exempt":0,
            "nontaxable":0,
            "taxable":299.99,
            "rate":0.065,
            "tax":19.5
         },
         {
            "jurisdictiontype":"County",
            "jurisdiction":"KITSAP",
            "exempt":0,
            "nontaxable":0,
            "taxable":299.99,
            "rate":0,
            "tax":0
         },
         {
            "jurisdictiontype":"City",
            "jurisdiction":"BAINBRIDGE ISLAND",
            "exempt":0,
```

```
                "nontaxable":0,
                "taxable":299.99,
                "rate":0.025,
                "tax":7.5
             }
          ]
       },
       {
          "lineitemid":"me",
          "shipmentid":"me",
          "taxes":[
             {
                "jurisdictiontype":"State",
                "jurisdiction":"WASHINGTON",
                "exempt":0,
                "nontaxable":0,
                "taxable":9.99,
                "rate":0.065,
                "tax":0.65
             },
             {
                "jurisdictiontype":"County",
                "jurisdiction":"KITSAP",
                "exempt":0,
                "nontaxable":0,
                "taxable":9.99,
                "rate":0,
                "tax":0
             },
             {
                "jurisdictiontype":"City",
                "jurisdiction":"BAINBRIDGE ISLAND",
                "exempt":0,
                "nontaxable":0,
                "taxable":9.99,
                "rate":0.025,
                "tax":0.25
             }
          ]
       }
    ]
```

**JSON attributes:**

lineitemid represents the object ID of the line item from Business Manager. For example, for the product line item, this value would be its Product ID.

shipmentid attribute in each object represents the Shipment to which current line item belongs.

jurisdictiontype represents the tax jurisdiction of in which the current tax amount is calculated. e.g. Country, State, City, County, Special etc.

jurisdiction represents the jurisdiction name. e.g. Canada, Ontario, California etc.

taxname represents a short tax description for the tax applied. e.g. CANADA GST/TPS, TN STATE TAX etc.

nontaxable has a value which is not taxable taxable attribute indicates the amount that is taxed.

exempt attribute indicates the amount which is tax-exempt.

rate contains the tax rate used for calculating the taxes tax has a value which is the tax that is calculated.

## 3.5. Services

Some features are also included in this integration as experimental ones. These can be accessed in the AvaTax Settings menu under AvaTax.

Navigate to – **Merchant Tools > AvaTax > AvaTax settings > Services**

- **Test Connection** – This can be used to verify whether connection to AvaTax is authenticated/successful. If it indicates that the connection is not authenticated, you may want to have a look at the Service credentials configured in step 3.2.1
- **Void Transaction** – This feature can be used void/cancel an order posted earlier to AvaTax. It cancels the order, or it does not, depending on if the order can be cancelled or not. In either case the service response is returned and displayed in the text area below this section.
- **Commit Transaction** – This feature can be used commit an order posted earlier to AvaTax. It marks the order committed or it does not, depending on if the order can be committed or not. In either case the service response is returned and displayed in the text area below this section.
- **Validate Address** – This feature can be used to validate shipping address of an order and receive a validated address as response.

All the above features can be scaled to process many orders on the storefront.

### 3.6. External Interfaces

All requests are done through AvaTax REST v2 API. The full reference guide, along with the resource structure for requests, can be found on their developer site - https://developer.avalara.com/api-reference/AvaTax/rest/v2/

### 3.7. Firewall Requirements

No firewall changes are required.

### 3.8. Testing

The cartridge also contains unit tests.

You can run `npm test` in the terminal or windows command prompt to execute all unit tests in the project. Unit tests are present under **/test/unit** directory.

# 4.0. Operations, Maintenance

## 4.1. Data Storage

The only Custom Objects stored in Salesforce B2C Commerce are the ones created during the initial import. These are described in the Configuration section.

For data stored by Avalara, please refer to website - http://www.avalara.com/

## 4.2. Availability

Cartridge functionality is dependent on the availability of the AvaTax API service.

Current AvaTax operational status can be viewed here - https://status.avalara.com/

## 4.3. Support

If you encounter any issues related to configuring or testing this integration, you may contact Avalara customer support for assistance. You can submit a case by email, telephone or chat. Please go to http://avalara.com/Technical-Support for more details as availability will vary depending on your support package.

## 5.0. FAQ

- **How to ensure accurate tax calculation?**
  - Make a note to configure tax codes, UPC codes, and any discounts for products in Business Manager. This information is used by AvaTax to accurately calculate taxes for a product.
  - This can either be manually configured in Business Manager > Merchant Tools > Products module or using a Product feed containing these attribute values.
- **When exactly is tax calculated in the e-commerce journey?**
  - On Shipping page
    a. When the end-user fills the shipping address details and continues to billing page
    b. When the end-user changes shipping method (provided that the Basket has shipping address accessible)
- **How many calls are made to AvaTax?**

  While the number of calls made to AvaTax are subject to Basket modifications, address modifications, discounts applications etc., the below list indicates a minimum number of calls made to AvaTax during the checkout journey.
  - On shipping page, after filling shipping address and continuing to billing page (as SalesOrder document)
  - On Billing page, on filling the billing address (as SalesOrder document)
  - On Placing the order (as SalesInvoice document in uncommitted state)
  - Calls to AvaTax may also be made if the content of the Basket changes in any sense during the checkout process. i.e. If the user adds/removes a product in/from the Basket, or applies a discount, changes an address, or changes the shipping method etc.
- **How to disable AvaTax on a site?**

  To disable AvaTax tax calculation and/or address validation for a site –
  - Set the values of the Custom Preferences *'Enable AvaTax'* and *'Enable AvaTax Address Validation'* to No. (Business Manager > Merchant Tools > Site Preferences > Custom Preferences > Avatax)
  - Remove any code changes made as a part of AvaTax integration.
- **What happens when the taxes cannot be calculated because of service unavailability, for example?**
  - The tax values are set to Zero instead of fallback to system defined tax rates. This is to ensure that the tax calculation is entirely handled by AvaTax for compliance.

## 6.0. Known Issues

No known existing issues. Please report any issues to support@avalara.com.

## 7.0. Release History

| Version | Date | Changes |
|---|---|---|
| 0.1 | 8/8/2010 | Initial release |
| 1.1 | 4/6/2011 | Shipping Address Validation call added |
| 1.2 | 5/4/2011 | Shipping Address Validation standalone call added |
| 2.0 | 7/29/2013 | Avatax Testing Center added for clients and developers<br>Test Connection, Commit and Cancel Tax features added<br>AVS enhancements added<br>Transaction logging added<br>Customer and Entity/Use code added<br>Fixed order level promotion bugs<br>Fixed documentation and added enhancement steps<br>Added more custom preferences and configurations for the enhancements displayed above<br>Cleaned up code and added detailed comments |
| 2.1 | 7/16/2015 | Fixed Multi-Shipping<br>Implemented use of Web Service Framework Configuration on SFCC Platform<br>Added metadata files and instructions on importing |
| 3.0 | 8/12/2015 | Refactored cartridge. |
| 3.1 | 6/8/2016 | Changed Detail Level<br>Updated to use total tax per line instead of recalculating based on tax rate |
| 3.2 | 6/30/2016 | Added Country Code, Currency Code, and VAT ID (if applicable) to GetTax request.<br>Reduced number of calls made to the service. |
| 3.3 | 12/10/2017 | Added support for controllers. |
| 19.1 | 07/12/2018 | AvaTax REST API support<br>Added support for VAT and Invoice Messages for EU countries.<br>Reconciliation Utility BM Extension<br>Commit Transaction on successfully placing order<br>Custom Customer Code support |
| 19.2 | 06/12/2019 | Added cross border tax calculation support<br>New BM tools for AvaTax settings |