Avalara AvaTax

AvaTax Mapper

Guide for Extending Avalara AvaTax For Salesforce

Version 1.0
Release Date: 8/31/18
Product Release: AvaTax Mapper 1.0

Avalara
Making sales tax less taxing

Table of Contents

AvaTax Mapper

AvaTax Mapper is a framework that helps you to create **Address Validation** and **Tax Calculation** request based upon the XML configuration supplied. It includes a large library for **Analyzing** the **XML** and **Formulating** the **service request** models. It supports a variety of Salesforce-based products like CPQ and Billing.

Architecture

- XML Analyzer
- Tax Calculator
- Address Validator
- Hook Manager

Architecture

XML Analyzer

XML analyzer provides a platform for reading XML configuration file, which extracts a meaningful data and converts it into a map/dictionary of an understandable data type. This map/dictionary plays an important role while creating service request objects required by AvaTax services.

Tax Calculator

This class handles all the complexities of creating tax calculation request from an XML file. It also provides out of the box Hooks, which can be extended by the customers to override the default flow of the AvaTax Mapper. So that, it can fit into their business process if required.

The following extension methods are provided for tax calculation:

1. Before Tax Calculation
2. Tax Validation
3. Override Request Line
4. Before Update
5. After Update

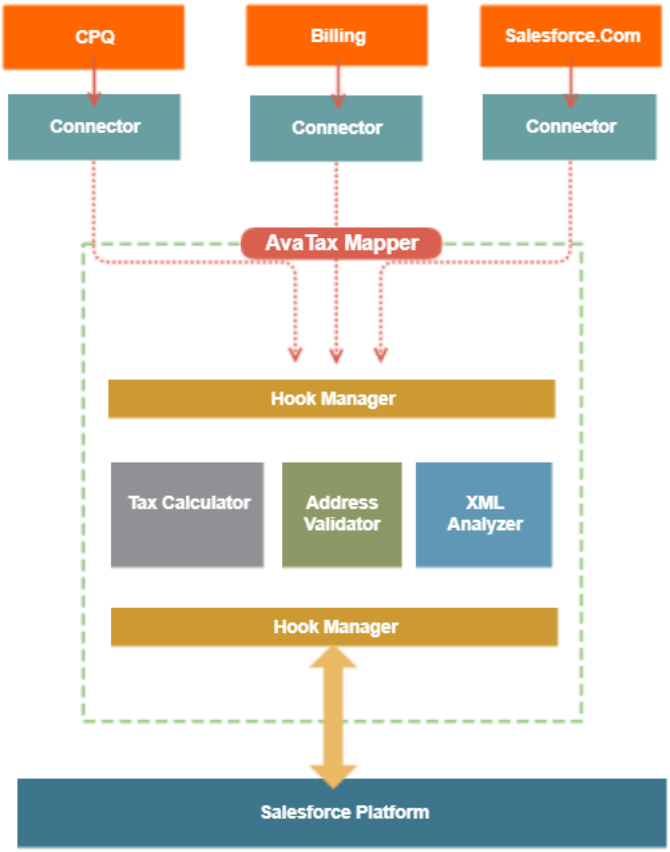
Address Validator

This class handles all the complexities of creating address validation request and displaying the validated address on the separate page. It also provides an option to update the original address if the user wants to update the address.

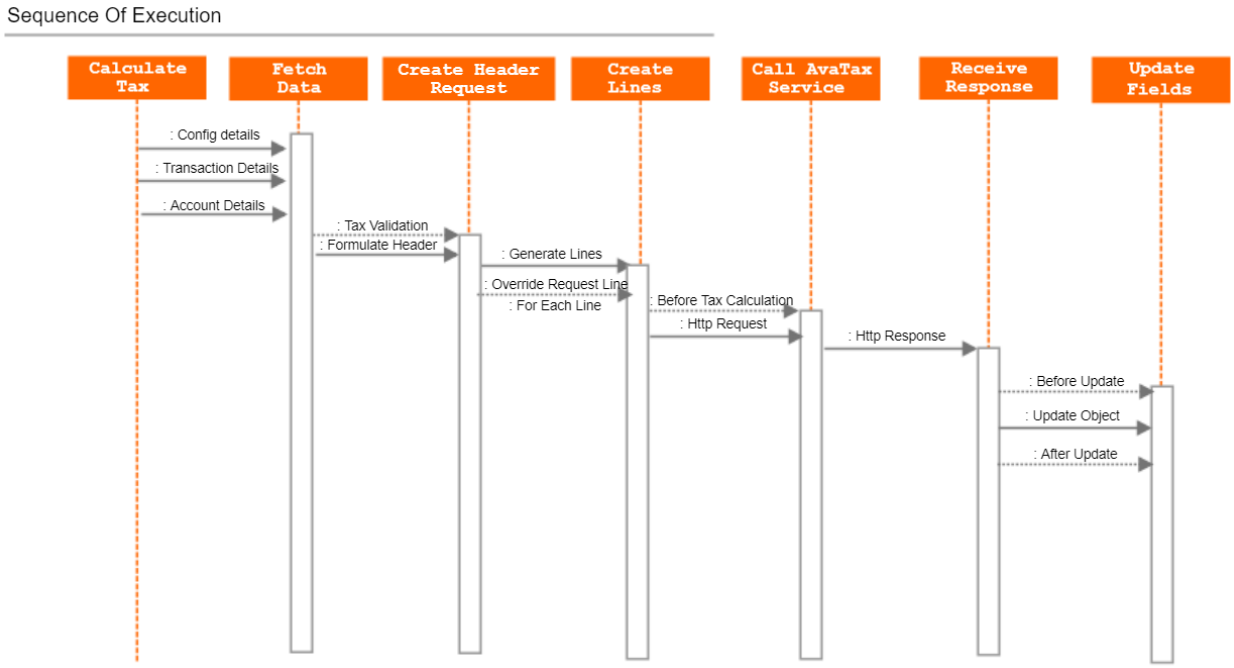
Hook Manager

This manager is the core of providing extension, so that the customer/SI partners can modify the web request that is going to the AvaTax engine. The administrator or developer needs to create a class, which implements a particular interface for accomplishing certain modification in the system to handle custom business requirement. Then the user can provide the class name in XML config file. The hook manager takes into consideration about the extensions that have been provided in XML config.

The following sequence diagram gives you more information on the hook manager and the extensions:



To elaborate more about the hook manager and the extensions which we provide let's consider following sequence diagram.



AVATAX Mapper XML Configuration Guideline

AVATAX Mapper XML configuration file help to specify "Request | Response" parameter essential for consuming AvaTax service. The Mapper XML is part of the Base AvaTax package. The XML formulate the request and upgrade the response for Tax Calculation, Address Validation, and Document Commits.

This topic describes the main configuration settings that you can make in the AVATAX XML.config file.

Configuration File Section

The XML configuration file is divided in various node that are specified as:

XML Scaffold

```
<avatax>
<FeatureNode1>
 <ObjectNode1>
    <ObjectComponent1>
        <Hook></Hook>
        <QueryNode></QueryNode>
        <ParameterNode> </ParameterNode>
        <MappingNode></MappingNode>
        <UpdateMappingNode> </UpdateMappingNode>
    </ObjectComponent1>
 </ObjectNode1>
 . . .
  <ObjectNode n> </ObjectNode n>
 </FeatureNode1>
 . . .
 <FeatureNode n></FeatureNode n>
</avatax>
```

Beginning Node

This node specifies the beginning of the AVATAX XML Mapper file. For example, **<avatax> </avatax>**.

Feature Node

This node specifies the beginning of the function name that needs to be covered under the XML. Based on the value and purpose, the AVATAX Salesforce Core package decides, which AVATAX API need to call. For example, **<addressvalidation> </addressvalidation>**.

Object Node

This node specifies Salesforce Object for Feature Node. For example, **<Account></Account>**.

Object Component

- This node specifies the Object Component that needs to be considered for the Feature node. For example, Billing Address From Account to be considered for Address Validation.
- The account contains two Addresses "Billing and Shipping." In this example, you are considering Billing Address.

Feature Node	Object Node	Object Component
Address Validation	Account	Billing

The object component contains below nodes:

1. Query Node
2. Parameter Node
3. Mapping Node
4. Update Mapping Node

Query Node

a. This node specifies the SOQL that need to query for fetching the attribute of the Object Component specified in the Object Component Node.

b. This node specifies the attribute that needs to be queried and can be further specified for the Mapping Node.

```
<query>
    select BillingStreet, BillingCity,BillingState,BillingCountry,Billingpostalcode
    from    Account     where id = '{AccountId}'
</query>
```

Parameter Node

This node specifies the attribute that needs to be used for the WHERE clause in SOQL specified in Query Node.

```
<query>
   select  BillingStreet,  BillingCity,BillingState,BillingCountry,Billingpostalcode
   from Account where id = '{AccountId}'
</query>
<parameters>
   <AccountId>Id</AccountId>
</parameters>
```

Note: The id - <AccountId>Id</AccountId> here is the record id of the object.

Mapping Node

a. This node specifies the Request parameters required for Request creation for Feature Node.

b. Each parameter in the node is a Feature Request Parameter.

```
<Mapping>
         <line1>BillingStreet</line1>
         <city>BillingCity</city>
         <region> BillingState </region>
         <country>BillingCountry</country>
         <postalcode> Billingpostalcode </postalcode>
</Mapping>
```

Note: The field mark in **Red** are **AvaTax** Request Model field, and field mark **Teal** are **Salesforce** standard field.

Update Mapping Node

This node specifies the Response field, where Service Response needs to be shown. For example, Address Validation Feature.

```
<UpdateMapping>
                <line1>BillingStreet</line1>
                 <city>BillingCity</city>
                 <region> BillingState </region>
                 <country>BillingCountry</country>
                 <postalcode> Billingpostalcode </postalcode>
 </UpdateMapping>
```

Note: The field mark in **Red** are **AvaTax** Request Model field, and field mark **Teal** are **Salesforce** standard field.

Sample Tax Calculation XML

The XML hierarchy remains the same till the Object Node, and the change from Object Component Node are:

Let's now consider Tax Calculation XML

Header Node

This node is used for specifying the Header of the Tax Calculation request.

The header node contains below node:

Query Node

This node specifies the SOQL that need to query for fetching the attribute of the Object Component specified in the Object Component Node. Also, specifies the attribute to be queried and can be further specified for the Mapping Node.

```
<query>
    select Id, AccountId,SalesTax__c,CloseDate,LeadSource from Opportunity where id
    = '{OpportunityId}'
</query>
```

Parameter Node

This node specifies the attribute to be used for the WHERE clause in SOQL specified in Query Node.

```
<query>
       select Id, AccountId,SalesTax__c,CloseDate,LeadSource from Opportunity where
       id = '{OpportunityId}'
</query>
<parameters>
        <OpportunityId>Id</OpportunityId>
</parameters>
```

Note: The id - < OpportunityId >Id</ OpportunityId > here is the record id of the object.

Mapping Node

This node specifies the Request parameters essential for Request creation for Feature Node. Also, each parameter in the node is a Feature Request Parameter.

```
<Mapping>
            <code>Id</code>
            <customerCode>AccountId</customerCode>
            <description>LeadSource</description>
</Mapping>
```

Note: The field mark in **Red** are **AvaTax** Request Model field, and field mark **Teal** are **Salesforce** standard field.

Update Mapping Node

This node specifies the Response field, and where Service Response needs to be shown. For example, Address Validation Feature.

```
<UpdateMapping>
       < SalesTax__c >totalTax</line1>
</UpdateMapping>
```

Note: The field mark in **Red** are **AvaTax** Request Model field, and field mark **Teal** are **Salesforce** standard field.

Line Node

This node is used for specifying the Line of the Tax Calculation request.

The line node contains the below nodes:

Query Node

a. This node specifies the SOQL that need to be queried for fetching the attribute of the Object Component specified in the Object Component Node.

b. This node specifies attribute to be queried and can be further specified for the Mapping Node.

```
<query>
select Id, Pricebookentry.product2.ProductCode, Description, SalesTax__c,
Quantity, TotalPrice, UnitPrice from OpportunityLineItem where OpportunityId =
'{OpportunityId}'
</query>
```

Parameter Node

This node specifies the attribute to be use for the WHERE clause in SOQL specified in Query Node.

```
<query>
      select Id, Pricebookentry.product2.ProductCode, Description, SalesTax__c,
      Quantity, TotalPrice, UnitPrice from OpportunityLineItem where OpportunityId
      = '{OpportunityId}'
</query>
<parameters>
       <OpportunityLineId>Id</ OpportunityLineId >
</parameters>
```

Note: The id - < OpportunityLineId >Id</ OpportunityLineId > here is the record id of the object.

Mapping Node

a. This node specifies the Request parameters essential for Request creation for Feature Node.

b. Each parameter in the node is a Feature Request Parameter.

```
<Mapping>
        <quantity> Quantity </code>
        <itemCode> Pricebookentry.product2.ProductCode </customerCode>
        <amount> TotalPrice </description>
</Mapping>
```

Note: The field mark in **Red** are **AvaTax** Request Model field, and field mark **Teal** are **Salesforce** standard field

Update Mapping Node

a. This node specifies the Response field, and where Service Response needs to be shown. For example, Address Validation Feature.

```
<UpdateMapping>
        <SalesTax__c >totalTax</line1>
</UpdateMapping>
```

Note: The field mark in **Red** are **AvaTax** Request Model field, and field mark **Teal** are **Salesforce** standard field.

How to Use AvaTax Mapper?

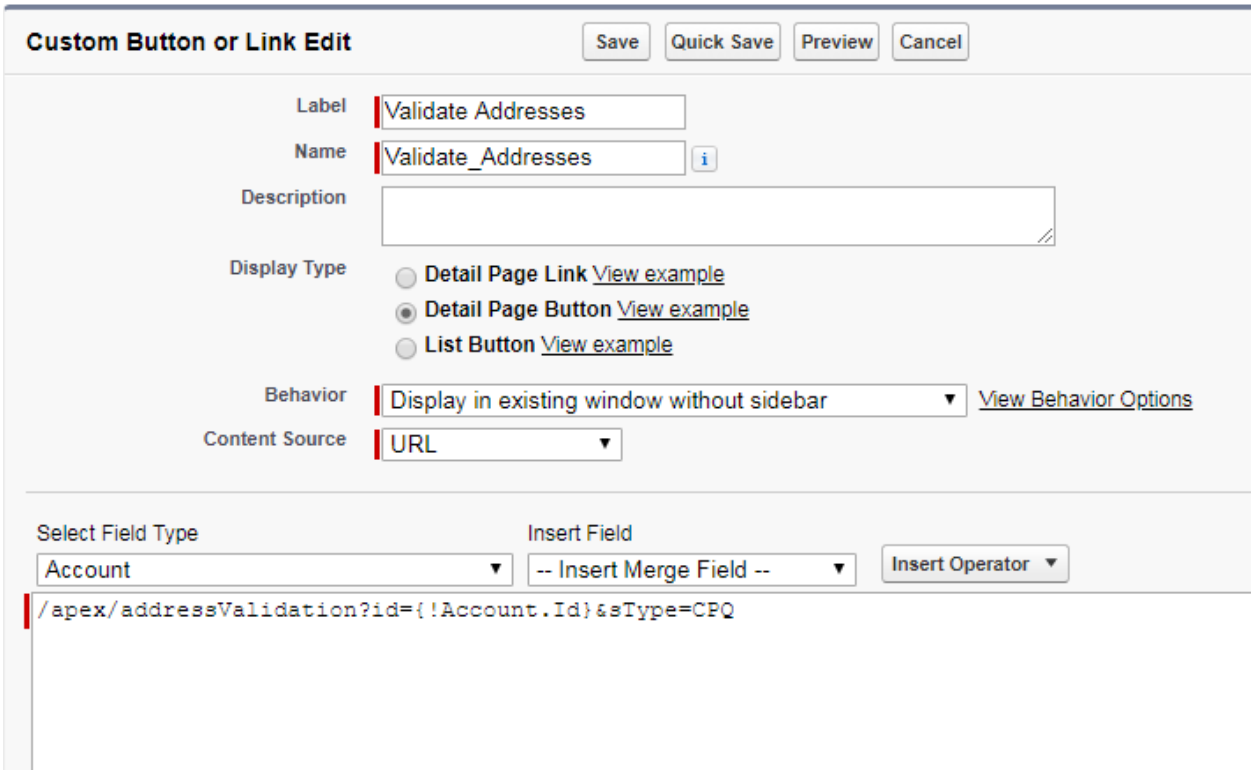
Address Validation

Address Validation follows easy and convenient approach using AvaTax Mapper.

Consider you need to have Address Validation on Account Object, and then you need to follow these steps:

Steps

1. Create a Button on the Object where Tax Calculation needs to be done.
2. Select the Content Source as URL for the button.



3. In URL box enter the data `/apex/addressValidation?id={!Account.Id}&sType=CPQ`
4. `{!Account.id}` specifies the Object id for Object where Address Validation need to be done.

AvaTax XML

```
<addressvalidation>
      <Account>
         <Billing>
             <query>select BillingStreet,
       BillingCity,BillingState,BillingCountry,Billingpostalcode from Account where
       id = '{AccountId}'</query>
                  <parameters>
                      <AccountId>Id</AccountId>
                  </parameters>
              <Mapping>
                  <line1>BillingStreet</line1>
                  <city>BillingCity</city>
                  <region>BillingState</region>
                  <country>BillingCountry</country>
                  <postalcode>Billingpostalcode</postalcode>
              </Mapping>
              <UpdateMapping>
                  <line1>BillingStreet</line1>
                  <city>BillingCity</city>
                  <region>BillingState</region>
                  <country>BillingCountry</country>
                  <postalcode>Billingpostalcode</postalcode>
              </UpdateMapping>
          </Billing>
</Account>
</addressvalidation>
```

Common Configuration

```
       global class Configuration extends ConfigurationBase{

         public Configuration()
         {
             environment = AvaTaxEnvironment.Sandbox;
             companyCode = 'AvaTax Company Code';
             username = 'AvaTax Username';
             password = 'AvaTax Password';
             isTaxCalculationEnabled = true;
         }
       }
```

Tax Calculation

Tax Calculation is easy and convenient using AvaTax Mapper.

Steps

1. Create a Visualforce page and add the Apex class as Controller Extension as shown
2. One must create a Configuration file for Config fetching, and the usage is depicted in Class TaxCalculator.

Visualforce Page

```
<apex:page standardController="Account" extensions=" TaxCalculator" action=”
taxCalculator”>
</apex:page>
```

Apex Class

```
      Public class TaxCalculator
      {
      Public void taxCalculator() {
              Configuration cm = new Configuration();
              ConfigurationBase config = cm.getConfig();
                   Map<String,String> oppParam = new Map<String,String>();



           TaxCalculationInput taxCalcInput = new TaxCalculationInput();
              taxCalcInput.recordId = opp.id;
              taxCalcInput.controller =
      opp.id.getSObjectType().getDescribe().getName().toLowerCase();
              taxCalcInput.optionalParams = oppParam;
              taxCalc.calculateTax(taxCalcInput);
          }
      }
```

3. Create a button on an Object and the Content Source for the button > Visualforce Page
4. Add the Visualforce Page create in step 2 and create the button.
5. For the scope of this example, we have created a field “SalesTax__c” on Opportunity and Opportunity product to show the response from the tax service.

Sample XML

```
                <Opportunity>
                            <query>select Id,
      AccountId,SalesTax__c,CloseDate,LeadSource from Opportunity where id =
      '{OpportunityId}'</query>
                            <parameters>
                                <OpportunityId>Id</OpportunityId>
                            </parameters>
                            <Mapping>
                                <code>Id</code>
                                <customerCode>AccountId</customerCode>
                                <description>LeadSource</description>
                            </Mapping>
                            <UpdateMapping>
                                <SalesTax__c>totalTax</SalesTax__c>
                            </UpdateMapping>
                        </Opportunity>
```

AvaTax XML

```
<taxcalculation>
    <Opportunity>
        <hooks>
            <overriderequestline></overriderequestline>
            <beforetaxcalculation></beforetaxcalculation>
            <beforeupdate></beforeupdate>
            <afterupdate></afterupdate>
        </hooks>
        <Header>
            <Queries>
                <Opportunity>
                    <query>select Id,
AccountId,SalesTax__c,CloseDate,LeadSource from Opportunity where id =
'{OpportunityId}'</query>
                    <parameters>
                        <OpportunityId>Id</OpportunityId>
                    </parameters>
                    <Mapping>
                        <code>Id</code>
                        <customerCode>AccountId</customerCode>
                        <description>LeadSource</description>
                    </Mapping>
                    <UpdateMapping>
                        <SalesTax__c>totalTax</SalesTax__c>
                    </UpdateMapping>
                </Opportunity>
                <Account>
                    <query>select Id, AccountNumber, Name, ShippingStreet,
ShippingCity, ShippingState, Shippingpostalcode, ShippingCountry,
                        BillingStreet, BillingCity, BillingState,
Billingpostalcode, BillingCountry from Account where Id =
'{AccountId}'</query>
                    <parameters>
                        <AccountId>Opportunity.AccountId</AccountId>
                    </parameters>
                    <Mapping>
                        <exemptNo>AccountNumber</exemptNo>

<addresses.shipTo.line1>ShippingStreet</addresses.shipTo.line1>
                        <addresses.shipTo.line2 />
                        <addresses.shipTo.line3 />

<addresses.shipTo.city>ShippingCity</addresses.shipTo.city>

<addresses.shipTo.region>ShippingState</addresses.shipTo.region>

<addresses.shipTo.country>ShippingCountry</addresses.shipTo.country>

<addresses.shipTo.postalcode>Shippingpostalcode</addresses.shipTo.postalcode
>

<addresses.shipTo.latitude></addresses.shipTo.latitude>
```

```
        <addresses.shipTo.longitude></addresses.shipTo.longitude>

        <addresses.shipFrom.line1>BillingStreet</addresses.shipFrom.line1>
                                <addresses.shipFrom.line2 />
                                <addresses.shipFrom.line3 />

        <addresses.shipFrom.city>BillingCity</addresses.shipFrom.city>

        <addresses.shipFrom.region>BillingState</addresses.shipFrom.region>

        <addresses.shipFrom.country>BillingCountry</addresses.shipFrom.country>

        <addresses.shipFrom.postalcode>Billingpostalcode</addresses.shipFrom.postalc
        ode>

        <addresses.shipFrom.latitude></addresses.shipFrom.latitude>

        <addresses.shipFrom.longitude></addresses.shipFrom.longitude>
                            </Mapping>
                            <UpdateMapping>
                                <SalesTax__c>totalTax</SalesTax__c>
                            </UpdateMapping>
                        </Account>
                    </Queries>
                </Header>
                <Line>
                    <Queries>
                        <OpportunityLineItem>
                            <query>select Id, Pricebookentry.product2.ProductCode,
        Description, SalesTax__c,
                                Quantity, TotalPrice, UnitPrice from
        OpportunityLineItem where OpportunityId = '{OpportunityId}'</query>
                            <parameters>
                                <OpportunityId>Opportunity.Id</OpportunityId>
                            </parameters>
                            <Mapping>
                                <description>Description</description>
                                <quantity>Quantity</quantity>

        <itemCode>Pricebookentry.product2.ProductCode</itemCode>
                                <amount>TotalPrice</amount>
                            </Mapping>
                            <UpdateMapping>
                                <SalesTax__c>tax</SalesTax__c>
                            </UpdateMapping>
                        </OpportunityLineItem>
                    </Queries>
                </Line>
            </Opportunity>
         </taxcalculation>
```

Post-Tax Calculation

Post-Tax Calculation is easy and convenient using AvaTax Mapper.

Steps

1. Create a Visualforce page and add the Apex class as Controller Extension as shown
2. One must create a Configuration file for Config fetching, and the usage is depicted in Class TaxCalculator.

Visualforce Page

```
<apex:page standardController="Account" extensions=" PostTaxCalculator" action=”
taxCalculator”>
</apex:page>
```

Apex Class

```
Public class PostTaxCalculator
{
Public void taxCalculator() {
        Configuration cm = new Configuration();
        ConfigurationBase config = cm.getConfig();
             Map<String,String> oppParam = new Map<String,String>();


   PostTaxCalculator ptc = new PostTaxCalculator('CPQ',config);
        PostTaxCalculationInput postTaxCalcInput = new PostTaxCalculationInput();
        postTaxCalcInput.companyCode = 'DEFAULT';
        postTaxCalcInput.transactionCode = String.valueof(opp.id);
        postTaxCalcInput.commitFlag = true;
        postTaxCalcInput.docType = DocumentType.SalesInvoice;
             ptc.postTax(postTaxCalcInput);
    }
}
```

3. Create a button on an Object and the Content Source for the button > Visualforce Page
4. Add the Visualforce Page create in step 2 and create the button.

AvaTax XML

```
<posttaxcalculation>
        <Opportunity>
            <hooks>
            </hooks>
            <Queries>
       <Opportunity>
                    <query>select Id, AccountId,SalesTax__c,CloseDate,LeadSource
from Opportunity where id = '{OpportunityId}'</query>
                    <parameters>
                        <OpportunityId>Id</OpportunityId>
                    </parameters>
                    <Mapping>

              <transactionCode>Id </transactionCode>

  </Mapping>
  </Queries>
        </Opportunity>
    </posttaxcalculation>
```

Cancel Tax Calculation

Cancel Tax Calculation is easy and convenient using AvaTax Mapper.

Steps

1. Create a Visualforce page and add the Apex class as Controller Extension as shown
2. One must create a Configuration file for Config fetching, and the usage is depicted in Class TaxCalculator.

Visualforce Page

```
<apex:page standardController="Account" extensions=" CancelTaxCalculator" action=”
taxCalculator”>
</apex:page>
```

Apex Class

```
Public class CancelTaxCalculator
{
Public void taxCalculator() {
        Configuration cm = new Configuration();
        ConfigurationBase config = cm.getConfig();
             Map<String,String> oppParam = new Map<String,String>();


   CancelTaxCalculationInput cancelTaxCalcInput = new CancelTaxCalculationInput();
        cancelTaxCalcInput.recordId = opp.id;
        cancelTaxCalcInput.controller =
opp.id.getSObjectType().getDescribe().getName().toLowerCase();
        cancelTaxCalcInput.optionalParams = oppParam;
        cancelTaxCalcInput.companyCode = 'DEFAULT';
        cancelTaxCalcInput.transactionCode = String.valueof(opp.id);
        cancelTaxCalcInput.code = VoidReasonCode.DocVoided;
        cancelTaxCalcInput.docType = DocumentType.SalesInvoice;
             ctc.cancelTax(cancelTaxCalcInput);
   }
}
```

3. Create a button on an Object and the Content Source for the button > Visualforce Page
4. Add the Visualforce Page create in step 2 and create the button.

Apex Class

```
    <canceltaxcalculation>
                <Opportunity>
            <hooks>
            </hooks>
            <Queries>
       <Opportunity>
                    <query>select Id, AccountId,SalesTax__c,CloseDate,LeadSource
from Opportunity where id = '{OpportunityId}'</query>
                    <parameters>
                        <OpportunityId>Id</OpportunityId>
                    </parameters>
                    <Mapping>

              <transactionCode>Id </transactionCode>

  </Mapping>
  </Queries>
        </Opportunity>


</canceltaxcalculation>
```

Hooks Manager

This manager is the core of providing extension so that customer/SI partners can modify the web request that is going to AvaTax engine.

Hook manager takes into consideration about the extensions that have been provided in XML config.

Hooks available for tax calculation are below:

AvaTax XML

```
<taxcalculation>
        <Order>
            <hooks>
                <overriderequestline></overriderequestline>
                <beforetaxcalculation></beforetaxcalculation>
                <beforeupdate></beforeupdate>
                <afterupdate></afterupdate>
            </hooks>
</Order>
</taxCalculation>
```

Example for Hooks Definition

BeforeTaxCalculationExtender:

```
global class BeforeTaxCalculationExtender implements IBeforeTaxCalculation {
   global void hookExtention(CreateTransactionModel model,
Map<String,List<SObject>> headerResults) {
       model.code = 'overrideDocCode';
   }
}
```

Now to provide the extension in XML file you need to update XML as follows:

```
<taxcalculation>
        <Order>
            <hooks>
                <overriderequestline></overriderequestline>
                <beforetaxcalculation>BeforeTaxCalculationExtender
</beforetaxcalculation>
                <beforeupdate></beforeupdate>
                <afterupdate></afterupdate>
            </hooks>
</Order>
</taxCalculation>
```

Extending with the AvaTax REST API

With the AvaTax REST API you can extend what the integration is capable of. To stay up to date with all of the features of the API, you will want to use our developer.avalara.com website.

Here are a few links to help you get started:

- REST V2 API Reference:
  - https://developer.avalara.com/api-reference/avatax/rest/v2/
- Developer Guide:
  - https://developer.avalara.com/avatax/dev-guide/
- Developer Forums:
  - https://community.avalara.com/avalara/category_sets/developers