SF-LWR Extension Framework : Tax Integration

Integration/Customization Name	Tax Integration
Description / Requirements Abstract	This integration can be used wherever there is a need to calculate tax using services like Avalara. It takes care of backend/integration only & any UI change would be specific to the project as per the requirement.
Status	COMPLETE
External References	**Salesforce Commerce Extensions** https://developer.salesforce.com/docs/commerce/salesforce-commerce/guide/extensions.html **Dashboard** Sandbox dashboard **-** https://sandbox.admin.avalara.com/ **Transaction Type** Create Transaction - https://developer.avalara.com/api-reference/avatax/rest/v2/methods/Transactions/CreateTransaction/ **Request/Response** https://developer.avalara.com/api-reference/avatax/rest/v2/models/CreateTransactionModel/ https://developer.avalara.com/api-reference/avatax/rest/v2/models/LineItemModel/ **Avalara tax codes** https://taxcode.avatax.avalara.com/ https://knowledge.avalara.com/bundle/dqa1657870670369_dqa1657870670369/page/Avalara_tax_codes.html **Tax code-mapping template** https://knowledge.avalara.com/bundle/bla1700809896571_bla1700809896571/page/Guidelines_for_the_item_and_tax_code_import_template.html **Understand harmonized system tariff codes** https://knowledge.avalara.com/bundle/dqa1657870670369_dqa1657870670369/page/Understand_harmonized_system_tariff_codes.html **Understanding freight taxability** https://knowledge.avalara.com/bundle/dqa1657870670369_dqa1657870670369/page/Understanding_freight_taxability.html
Notes	Before using this integration pls make sure client confirms with Avalara about how rate limiting works in case if an application connects with Avalara using this integration rather than the ones which are already available like Avalara connector (available on AppExchange) as these connector may have rate limiting controlled by their licenses.

Overview

Overview

This document provides details about how to integrate B2B/D2C commerce with a tax provider (like Avalara etc.) to get taxes for products & shipping.

This integration is based on **Salesforce Commerce Extension** framework which was introduced in Winter' 24 release and going forward it is ***recommended***

to use extensions over integrations because they offer more targeted customizations for B2B/D2C store.

Capabilities

- Using this integration you can get tax calculated for carts
  - Having single/multiple delivery group(s)
  - Having Shipping methods
  - Using product tax codes
  - Supports customer level tax exemption
  - Supports product level tax exemption
- In case a store supports multiple locales/countries , can configure locale specific shipping providers so a store can have multiple shipping providers configured

Technical Overview

This integration implementation can be divided in to two parts :

- **Configuration**
  - Custom Meta Data Types
    - Tax Provider
    - HTTP Service
  - Named Credentials
- **Source Code**
  - CartExtension.TaxCartCalculatorExtension

    - TaxConfiguration
    - TaxMetaData
    - TaxProviderFactory
    - TaxProvider
    - TaxProviderRequest
    - TaxProviderResponse
    - HTTPService

Extensibility

There are various points where this implementation can be extended to achieve project specific requirement:

- Meta Data
    - Can add more fields to both, the meta data types Tax Provider & HTTP Service
- Code
    - Following classes can be extended , details are available under respective item number
        - TaxCartCalculatorExtension
        - TaxProvider
        - HTTPService

Limitations

- Currently HTTS Service supports only JSON (as data format)

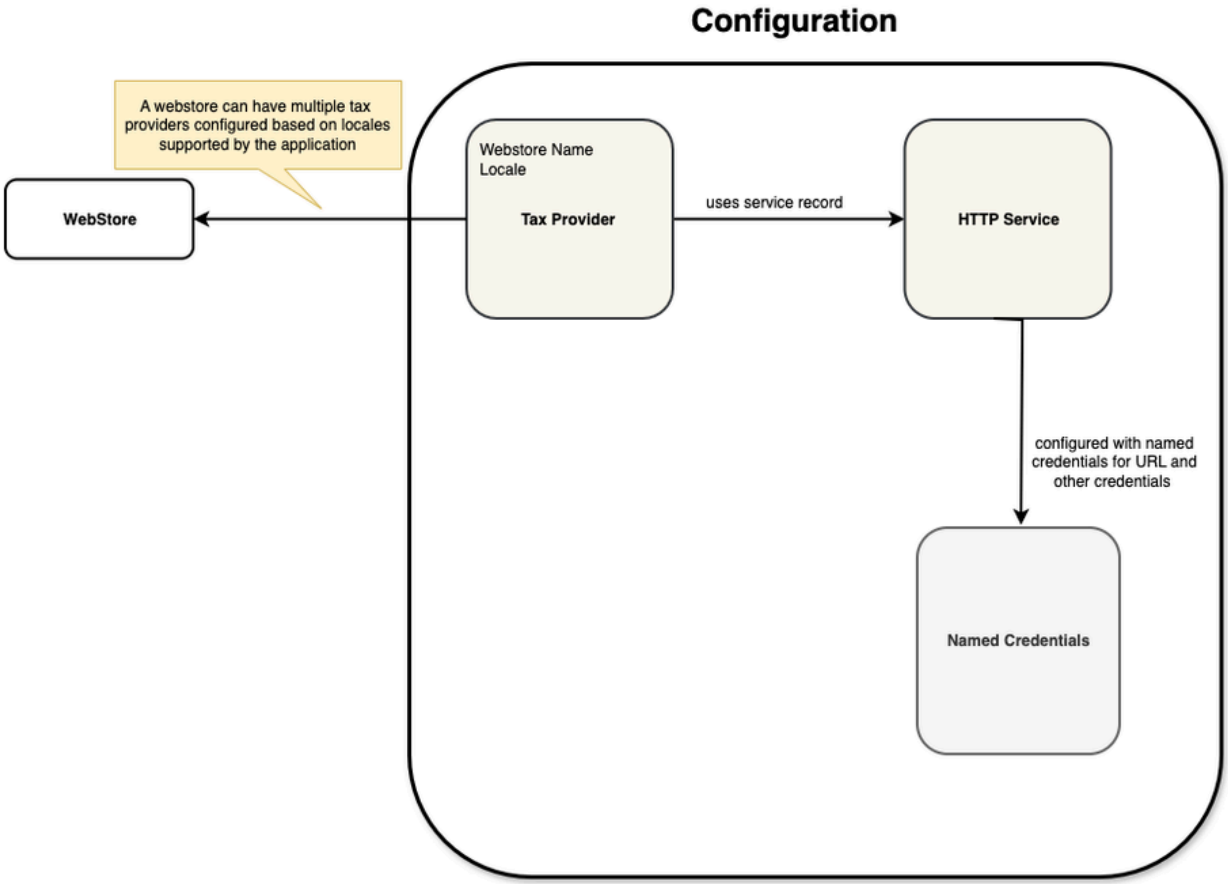
Prerequisites

- Standard object **Product2** should have following attributes to support product based exemption & tax codes.
    - Exemption Code
    - Entity Use Code
    - Taxable
    - Tax Class Id
- Standard object **Account** should have following attributes to support account based exemption
    - Exemption Code
    - Entity Use Code

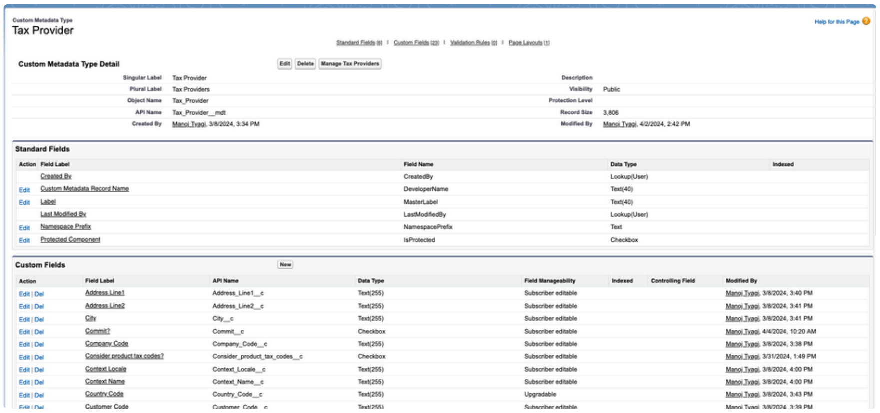
Upcoming Features

- Logging enhancements
- HTTP Service - Support for XML data format
- HTTP Service - Implementing Retry logic (in case callout fails)
- *Anything else as per feedback*

Configuration


Configuration
A webstore can have multiple tax providers configured based on locales supported by the application
WebStore
Webstore Name
Locale
Tax Provider
uses service record
HTTP Service
configured with named credentials for URL and other credentials
Named Credentials

Configuration Details

	Section	Task	Details
1	Salesforce Org	Create a Custom Metadata Type **Tax Provider**	This is how it looks like & further details can be found under Meta Data section : Setup > Custom Code > Custom Metadata Types > **Tax Provider** 

Custom Metadata Type
Tax Provider

Help for this Page

Standard Fields [6] | Custom Fields [23] | Validation Rules [0] | Page Layouts [1]

Custom Metadata Type Detail

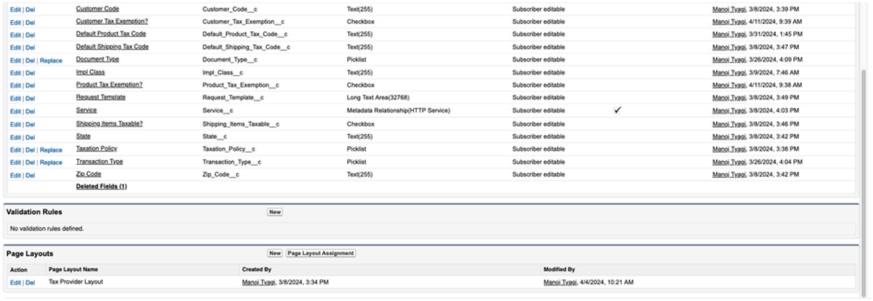
Singular Label	Tax Provider	Description	
Plural Label	Tax Providers	Visibility	Public
Object Name	Tax_Provider	Protection Level	
API Name	Tax_Provider__mdt	Record Size	3,806
Created By	Manoj Tyagi, 3/8/2024, 3:34 PM	Modified By	Manoj Tyagi, 4/2/2024, 2:42 PM

Standard Fields

Action	Field Label	Field Name	Data Type	Indexed
	Created By	CreatedBy	Lookup(User)	
Edit	Custom Metadata Record Name	DeveloperName	Text(40)	
Edit	Label	MasterLabel	Text(40)	
	Last Modified By	LastModifiedBy	Lookup(User)	
Edit	Namespace Prefix	NamespacePrefix	Text	
Edit	Protected Component	IsProtected	Checkbox	

Custom Fields

Action	Field Label	API Name	Data Type	Field Manageability	Indexed	Controlling Field	Modified By
Edit | Del	Address Line1	Address_Line1__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:40 PM
Edit | Del	Address Line2	Address_Line2__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:41 PM
Edit | Del	City	City__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:41 PM
Edit | Del	Commit?	Commit__c	Checkbox	Subscriber editable			Manoj Tyagi, 4/4/2024, 10:20 AM
Edit | Del	Company Code	Company_Code__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:38 PM
Edit | Del	Consider product tax codes?	Consider_product_tax_codes__c	Checkbox	Subscriber editable			Manoj Tyagi, 3/31/2024, 1:49 PM
Edit | Del	Context Locale	Context_Locale__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 4:00 PM
Edit | Del	Context Name	Context_Name__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 4:00 PM
Edit | Del	Country Code	Country_Code__c	Text(255)	Upgradable			Manoj Tyagi, 3/8/2024, 3:43 PM
Edit | Del	Customer Code	Customer_Code__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:39 PM

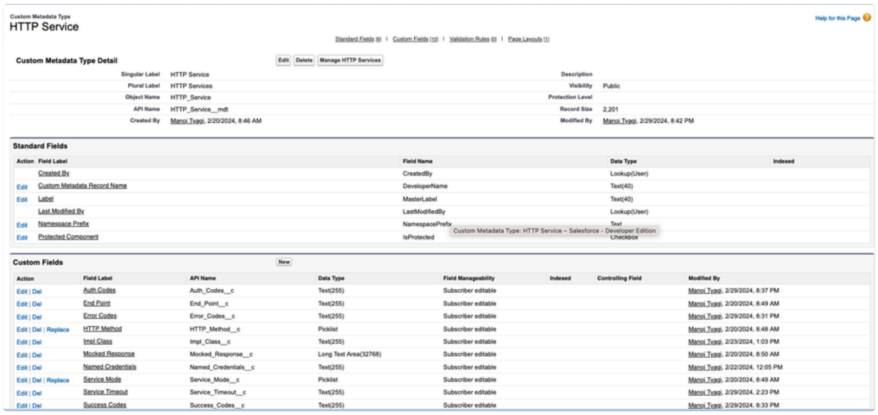
Edit | Del	Customer Code	Customer_Code__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:39 PM
Edit | Del	Customer Tax Exemption?	Customer_Tax_Exemption__c	Checkbox	Subscriber editable			Manoj Tyagi, 4/11/2024, 9:39 AM
Edit | Del	Default Product Tax Code	Default_Product_Tax_Code__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/31/2024, 1:45 PM
Edit | Del	Default Shipping Tax Code	Default_Shipping_Tax_Code__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:47 PM
Edit | Del | Replace	Document Type	Document_Type__c	Picklist	Subscriber editable			Manoj Tyagi, 3/26/2024, 4:09 PM
Edit | Del	Impl Class	Impl_Class__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/9/2024, 7:46 AM
Edit | Del	Product Tax Exemption?	Product_Tax_Exemption__c	Checkbox	Subscriber editable			Manoj Tyagi, 4/11/2024, 9:38 AM
Edit | Del	Request Template	Request_Template__c	Long Text Area(32768)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:49 PM
Edit | Del	Service	Service__c	Metadata Relationship(HTTP Service)	Subscriber editable	✓		Manoj Tyagi, 3/8/2024, 4:03 PM
Edit | Del	Shipping Items Taxable?	Shipping_Items_Taxable__c	Checkbox	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:46 PM
Edit | Del	State	State__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:42 PM
Edit | Del | Replace	Taxation Policy	Taxation_Policy__c	Picklist	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:38 PM
Edit | Del | Replace	Transaction Type	Transaction_Type__c	Picklist	Subscriber editable			Manoj Tyagi, 3/26/2024, 4:04 PM
Edit | Del	Zip Code	Zip_Code__c	Text(255)	Subscriber editable			Manoj Tyagi, 3/8/2024, 3:42 PM
	Deleted Fields (1)						

Validation Rules New

No validation rules defined.

Page Layouts New Page Layout Assignment

Action	Page Layout Name	Created By	Modified By
Edit | Del	Tax Provider Layout	Manoj Tyagi, 3/8/2024, 3:34 PM	Manoj Tyagi, 4/4/2024, 10:21 AM

2	Salesforce Org	Create a Custom Metadata Type **HTTP Service**	Here are Metadata type details - Setup > Custom Code > Custom Metadata Types > **HTTP Service** 

Custom Metadata Type

HTTP Service

Help for this Page
Standard Fields [6] | Custom Fields [10] | Validation Rules [0] | Page Layouts [1]

Custom Metadata Type Detail Edit Delete Manage HTTP Services

Singular Label	HTTP Service	Description	
Plural Label	HTTP Services	Visibility	Public
Object Name	HTTP_Service	Protection Level	
API Name	HTTP_Service__mdt	Record Size	2,201
Created By	Manoj Tyagi, 2/20/2024, 8:46 AM	Modified By	Manoj Tyagi, 2/29/2024, 8:42 PM

Standard Fields

Action	Field Label	Field Name	Data Type	Indexed
	Created By	CreatedBy	Lookup(User)	
Edit	Custom Metadata Record Name	DeveloperName	Text(40)	
Edit	Label	MasterLabel	Text(40)	
	Last Modified By	LastModifiedBy	Lookup(User)	
Edit	Namespace Prefix	NamespacePrefix	Text	
Edit	Protected Component	IsProtected	Checkbox	

Custom Metadata Type: HTTP Service ~ Salesforce - Developer Edition

Custom Fields New

Action	Field Label	API Name	Data Type	Field Manageability	Indexed	Controlling Field	Modified By
Edit | Del	Auth Codes	Auth_Codes__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/29/2024, 8:37 PM
Edit | Del	End Point	End_Point__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/20/2024, 8:49 AM
Edit | Del	Error Codes	Error_Codes__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/29/2024, 8:31 PM
Edit | Del | Replace	HTTP Method	HTTP_Method__c	Picklist	Subscriber editable			Manoj Tyagi, 2/20/2024, 8:48 AM
Edit | Del	Impl Class	Impl_Class__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/23/2024, 1:03 PM
Edit | Del	Mocked Response	Mocked_Response__c	Long Text Area(32768)	Subscriber editable			Manoj Tyagi, 2/20/2024, 8:50 AM
Edit | Del	Named Credentials	Named_Credentials__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/22/2024, 12:05 PM
Edit | Del | Replace	Service Mode	Service_Mode__c	Picklist	Subscriber editable			Manoj Tyagi, 2/20/2024, 8:49 AM
Edit | Del	Service Timeout	Service_Timeout__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/29/2024, 2:23 PM
Edit | Del	Success Codes	Success_Codes__c	Text(255)	Subscriber editable			Manoj Tyagi, 2/29/2024, 8:33 PM

Custom Metadata Type - Tax Provider

Field Label	API Name	Data Type	Notes
Address Line1	Address_Line1__c	Text(255)	Ship from street address
Address Line2	Address_Line2__c	Text(255)	Ship from line 2
City	City__c	Text(255)	Ship from city
Commit?	Commit__c	Checkbox	Is this transaction should be committed?
Company Code	Company_Code__c	Text(255)	Company Code (provided by tax provider)
Consider product tax codes?	Consider_product_tax_codes__c	Checkbox	Are tax codes configured at product level and set in tax request for each taxable product ?
Context Locale	Context_Locale__c	Text(255)	
Context Name	Context_Name__c	Text(255)	
Country Code	Country_Code__c	Text(255)	
Customer Code	Customer_Code__c	Text(255)	
Customer Tax	Customer_Tax_Exemption	Checkbox	

Exemption?	__c		
Default Product Tax Code	Default_Product_Tax_Code__c	Text(255)	
Default Shipping Tax Code	Default_Shipping_Tax_Code__c	Text(255)	
Document Type	Document_Type__c	Picklist	
Impl Class	Impl_Class__c	Text(255)	
Product Tax Exemption?	Product_Tax_Exemption__c	Checkbox	
Request Template	Request_Template__c	Long Text Area(32768)	
Service	Service__c	Metadata Relationship(HTTP Service)	
Shipping Items Taxable?	Shipping_Items_Taxable__c	Checkbox	
State	State__c	Text(255)	
Taxation Policy	Taxation_Policy__c	Picklist	
Transaction Type	Transaction_Type__c	Picklist	
Zip Code	Zip_Code__c	Text(255)	

Custom Metadata Type - HTTP Service

Field Label	API Name	Data Type
Auth Codes	Auth_Codes__c	Text(255)
End Point	End_Point__c	Text(255)
Error Codes	Error_Codes__c	Text(255)
HTTP Method	HTTP_Method__c	Picklist
Impl Class	Impl_Class__c	Text(255)
Mocked Response	Mocked_Response__c	Long Text Area(32768)
Named Credentials	Named_Credentials__c	Text(255)
Service Mode	Service_Mode__c	Picklist
Service Timeout	Service_Timeout__c	Text(255)
Success Codes	Success_Codes__c	Text(255)

Product2

Field Label	API Name	Data Type

Tax Class Id	Tax_Class_Id__c	Text(255)
Taxable?	Taxable__c	Checkbox

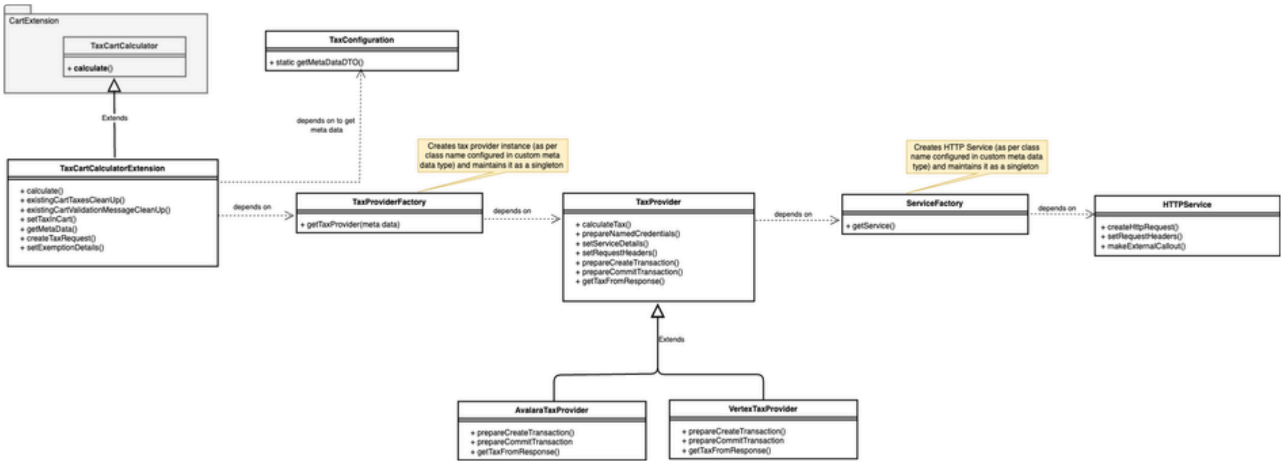
Account

Field Label	API Name	Data Type
Entity Use Code	Entity_Use_Code__c	Picklist
Exemption Code	Exemption_Code__c	Text(255)

Webstore

Field Label	API Name	Data Type
Tax Transaction Type	Tax_Transaction_Type__c	Text(255)
Tax Document Type	Tax_Document_Type__c	Text(255)

Class Diagram


CartExtension
TaxCartCalculator
+ calculate()
Extends
TaxConfiguration
+ static getMetaDataDTO()
depends on to get meta data
TaxCartCalculatorExtension
+ calculate()
+ existingCartTaxesCleanUp()
+ existingCartValidationMessageCleanUp()
+ setTaxInCart()
+ getMetaData()
+ createTaxRequest()
+ setExemptionDetails()
depends on
Creates tax provider instance (as per class name configured in custom meta data type) and maintains it as a singleton
TaxProviderFactory
+ getTaxProvider(meta data)
depends on
TaxProvider
+ calculateTax()
+ prepareNamedCredentials()
+ setServiceDetails()
+ setRequestHeaders()
+ prepareCreateTransaction()
+ prepareCommitTransaction()
+ getTaxFromResponse()
depends on
Creates HTTP Service (as per class name configured in custom meta data type) and maintains it as a singleton
ServiceFactory
+ getService()
depends on
HTTPService
+ createHttpRequest()
+ setRequestHeaders()
+ makeExternalCallout()
Extends
AvalaraTaxProvider
+ prepareCreateTransaction()
+ prepareCommitTransaction
+ getTaxFromResponse()
VertexTaxProvider
+ prepareCreateTransaction()
+ prepareCommitTransaction
+ getTaxFromResponse()

Source Code

Class	Details	Code
TaxCartCalculatorExtension	This class extends **CartExtension.TaxCartCalculator** provided by extension framework	As per need, you can extend this class & override following methods : public virtual void calculate public virtual void existingCartTaxesCleanUp public virtual void existingCartValidationMessageCleanUp public virtual void setTaxInCart public virtual TaxMetaData getMetaData

public virtual TaxProviderRequest createTaxRequest

public virtual void setExemptionDetails

```
public without sharing class TaxCartCalculatorExtension extend
  // You MUST change this to be your service or you must launc
  // and add the host in Setup | Security | Remote site settin
  private static String externalTaxHost = 'https://example.com

  // You MUST change the useExternalService to True if you wan
  private static Boolean useExternalService = false;
  public virtual override void calculate(
    CartExtension.CartCalculateCalculatorRequest request
  ) {
    try {
      CartExtension.Cart cart = request.getCart();
      existingCartValidationMessageCleanUp(cart);
      existingCartTaxesCleanUp(cart);
      CartExtension.CartItemList cartItems = cart.getCartItems
      TaxMetaData taxMetaData = getMetaData(cart.getWebstoreId
      TaxProviderRequest taxProviderRequest = createTaxRequest
        cart,
        taxMetaData
      );
      if (taxMetaData.customerTaxExmeption) {
        setExemptionDetails(taxProviderRequest, taxMetaData);
      }
      TaxProvider tProvider = TaxProviderFactory.getTaxProvide
      if (tProvider != null) {
        TaxProviderResponse tpResponse = tProvider.calculateTa
          taxProviderRequest
        );
        setTaxInCart(cart, taxMetaData, tpResponse);
      }
    } catch (Exception e) {
      // For testing purposes, this example treats exceptions
      // displayed to the buyer user. In production, you proba
      // errors. In that case, throw the exception here and ma
      // in place to let the admin know that the error occurre
      // handling for details about how to create that notific
      throw new CalloutException('There was a problem with the
    }
    return;
  }

  public virtual void existingCartTaxesCleanUp(CartExtension.C
    CartExtension.CartItemList cartItemCollection = cart.getCa
    for (Integer i = (cartItemCollection.size() - 1); i >= 0;
      CartExtension.CartItem cartItem = cartItemCollection.get
      if (cartItem.getCartTaxes().size() > 0) {
        for (Integer j = (cartItem.getCartTaxes().size() - 1);
          cartItem.getCartTaxes().remove(cartItem.getCartTaxes
        }
      }
      if (cartItem.getCartItemPriceAdjustments().size() > 0) {
        for (
          Integer k = (cartItem.getCartItemPriceAdjustments().
          k >= 0;
          k--
        ) {
          CartExtension.CartTaxList cipaTaxes = cartItem.getCa
          .get(i)
          .getCartTaxes();
        if (cipaTaxes.size() > 0) {
          for (
            Integer l = (cartItem.getCartItemPriceAdjustment
            l >= 0;
            l--
          ) {
            cipaTaxes.remove(cipaTaxes.get(l));
          }
        }
      }
    }
  }
}

public virtual void existingCartValidationMessageCleanUp(Car
  CartExtension.CartValidationOutputList cartValidationOutpu
  for (Integer i = (cartValidationOutputCollection.size() - 
    CartExtension.CartValidationOutput cvo = cartValidationO
      i
    );
    if (cvo.getType() == CartExtension.CartValidationOutputT
      cartValidationOutputCollection.remove(cvo);
    }
  }
}

public virtual void setTaxInCart(
  CartExtension.Cart cart,
  TaxMetaData taxMetaData,
  TaxProviderResponse tpResponse
) {
  Map<String, TaxableCartItem> tLineItems = tpResponse.taxab
  CartExtension.CartItemList cartItemCollection = cart.getCa
  for (Integer i = (cartItemCollection.size() - 1); i >= 0; 
    CartExtension.CartItem cartItem = cartItemCollection.get
    if (
      !taxMetaData.shippingItemsTaxable &&
      cartItem.getType() == CartExtension.SalesItemTypeEnum.
    ) {
      continue;
    }
    TaxableCartItem tLineItem;
    if (cartItem.getType() == CartExtension.SalesItemTypeEnu
      tLineItem = tLineItems.get(cartItem.getId());
    } else {
      tLineItem = tLineItems.get(cartItem.getCartDeliveryGro
    }

    if (tLineItem != null) {
      for (LineItemTaxDetails litDetails : tLineItem.lineIte
        CartExtension.CartTaxList cartTaxCollection = cartIt
        CartExtension.CartTax cartTax = new CartExtension.Ca
          CartExtension.TaxTypeEnum.ESTIMATED,
          litDetails.tax,
          'Sales'
        );
        cartTax.setTaxRate(String.valueOf(litDetails.rate));
        cartTaxCollection.add(cartTax);
        }
      }
    }
  }
  public virtual TaxMetaData getMetaData(String webStoreId) {
    return TaxConfiguration.getMetaDataDTO(webStoreId);
  }

  public virtual TaxProviderRequest createTaxRequest(
    CartExtension.Cart cart,
    TaxMetaData taxMetaData
  ) {
    Map<String, TaxableCartItem> taxableCartItems = new Map<St
    TaxProviderRequest request = new TaxProviderRequest();
    String cartId = cart.getId();
    request.cartId = cartId;
    request.taxMetaData = taxMetaData;
    request.currencyCode = cart.getCurrencyIsoCode();
    CartExtension.CartDeliveryGroupList cartDeliveryGroups = c
    if (cartDeliveryGroups.size() == 1) {
      CartExtension.CartDeliveryGroup cartDeliveryGroup = cart
        0
      );
      request.hasMultipleShipments = false;
      request.street = cartDeliveryGroup.getDeliverToAddress()
      request.city = cartDeliveryGroup.getDeliverToAddress().C
      request.state = cartDeliveryGroup.getDeliverToAddress().
      request.postalCode = cartDeliveryGroup.getDeliverToAddre
      request.country = cartDeliveryGroup.getDeliverToAddress(
    } else {
      request.hasMultipleShipments = true;
    }
    CartExtension.CartItemList cartItemCollection = cart.getCa
    for (Integer i = (cartItemCollection.size() - 1); i >= 0;
      CartExtension.CartItem cartItem = cartItemCollection.get
      if (
        !taxMetaData.shippingItemsTaxable &&
        cartItem.getType() == CartExtension.SalesItemTypeEnum.
      ) {
        continue;
      }

      TaxableCartItem tcItem = new TaxableCartItem();

      tcItem.amount = cartItem.getTotalAmount();
      tcItem.sku = cartItem.getSku();
      if (request.hasMultipleShipments) {
        tcItem.street = cartItem.getCartDeliveryGroup()
          .getDeliverToAddress()
          .Street;
        tcItem.city = cartItem.getCartDeliveryGroup()
          .getDeliverToAddress()
          .City;
        tcItem.state = cartItem.getCartDeliveryGroup()
          .getDeliverToAddress()
          .State;
        tcItem.postalCode = cartItem.getCartDeliveryGroup()
          .getDeliverToAddress()
          .PostalCode;
        tcItem.country = cartItem.getCartDeliveryGroup()
```

```
          .getDeliverToAddress()
          .Country;
      }
      if (cartItem.getType() == CartExtension.SalesItemTypeEnu
        tcItem.id = cartItem.getId();
        tcItem.lineItemType = 'Product';
        taxableCartItems.put(cartItem.getId(), tcItem);
      } else {
        tcItem.id = cartItem.getCartDeliveryGroup().getId();
        tcItem.lineItemType = 'Charge';
        taxableCartItems.put(cartItem.getCartDeliveryGroup().g
      }
    }
    if (taxMetaData.useProductTaxCodes || taxMetaData.productT
      String query = 'SELECT Id, Sku, TotalLineAmount, Quantit
      if (taxMetaData.useProductTaxCodes) {
        query += ',Product2.Tax_Class_Id__c';
      }
      if (taxMetaData.productTaxExmeption) {
        query += ',Product2.Entity_Use_Code__c,Product2.Exempt
      }
      query += ' FROM CartItem WHERE  cartId=:cartId';
      if (!taxMetaData.shippingItemsTaxable) {
        query += ' AND Type = \'Product\'';
      }
      TaxableCartItem tci;
      for (CartItem cartItem : Database.query(query)) {
        if (cartItem.Type.equals('Product')) {
          tci = taxableCartItems.get(cartItem.Id);
        } else {
          tci = taxableCartItems.get(cartItem.CartDeliveryGrou
        }
        if (tci != null) {
          if (
            taxMetaData.productTaxExmeption && !cartItem.Produ
          ) {
            tci.productExemptionCode = cartItem.Product2.Exemp
            tci.entityUseCode = cartItem.Product2.Entity_Use_C
          }
          if (taxMetaData.useProductTaxCodes && cartItem.Produ
            if (cartItem.Product2.Tax_Class_Id__c != null) {
              tci.taxClassId = cartItem.Product2.Tax_Class_Id_
            } else if (cartItem.Type.equals('Product')) {
              tci.taxClassId = taxMetaData.defaultProductTaxCo
            } else {
              tci.taxClassId = taxMetaData.defaultShippingTaxC
            }
          }
        }
      }
    }
    request.taxTransacionType = taxMetaData.documentType != nu
      ? taxMetaData.documentType
      : 'SalesOrder';
    request.taxableCartItems = taxableCartItems;
    return request;
  }

  public virtual void setExemptionDetails(
    TaxProviderRequest taxProviderRequest,
```

<table>
<tr>
<td></td>
<td></td>
<td>

```
    TaxMetaData taxMetaData
  ) {
    if (taxMetaData.customerTaxExmeption) {
      User user = [
        SELECT
          Id,
          AccountId,
          Account.Exemption_Code__c,
          Account.Entity_Use_Code__c
        FROM User
        WHERE id = :UserInfo.getUserId()
        LIMIT 1
      ];
      taxProviderRequest.customerExemptionCode = user.Account.
      taxProviderRequest.entityUseCode = user.Account.Entity_U
    }
  }
}


```

</td>
</tr>
<tr>
<td>TaxConfiguration</td>
<td>This class fetches custom meta data and stores in a map so that can be used further down in the execution flow</td>
<td>

```
public with sharing class TaxConfiguration {
  private static Map<String, TaxMetaData> taxProviders = new Ma
  private static Tax_Provider__mdt taxProviderMDT;
  private TaxConfiguration() {
  }

  public static TaxMetaData getMetaDataDTO(String contextId) {
    TaxMetaData taxMetaDataDTO;
    String currentLocale = UserInfo.getLocale();
    String dataKey = contextId + '-' + currentLocale;
    if (taxProviders.containsKey(dataKey)) {
      taxMetaDataDTO = taxProviders.get(dataKey);
    } else {
      String webStoreName = [
        SELECT Name
        FROM WebStore
        WHERE Id = :contextId
        LIMIT 1
      ]
      .Name;
      taxProviderMDT = [
        SELECT
          Id,
          QualifiedApiName,
          Taxation_Policy__c,
          Customer_Code__c,
          Company_Code__c,
          Shipping_Items_Taxable__c,
          Default_Shipping_Tax_Code__c,
          Default_Product_Tax_Code__c,
          Consider_product_tax_codes__c,
          Product_Tax_Exmeption__c,
          Customer_Tax_Exmeption__c,
          Document_Type__c,
          Transaction_Type__c,
          Commit__c,
          Address_Line1__c,
```

</td>
</tr>
</table>

```
          Address_Line2__c,
          City__c,
          State__c,
          Zip_Code__c,
          Country_Code__c,
          Impl_Class__c,
          Request_Template__c,
          Service__r.End_Point__c,
          Service__r.HTTP_Method__c,
          Service__r.Mocked_Response__c,
          Service__r.Service_Mode__c,
          Service__r.Service_Timeout__c,
          Service__r.Impl_Class__c,
          Service__r.Named_Credentials__c
        FROM Tax_Provider__mdt
        WHERE
          Context_Name__c = :webStoreName
          AND Context_Locale__c = :currentLocale
        LIMIT 1
      ];
      if (taxProviderMDT != null) {
        taxMetaDataDTO = new TaxMetaData();
        taxMetaDataDTO.requestJSON = taxProviderMDT.Request_Tem
        taxMetaDataDTO.providerImplClass = taxProviderMDT.Impl_
        taxMetaDataDTO.companyCode = taxProviderMDT.Company_Cod
        taxMetaDataDTO.customerCode = taxProviderMDT.Customer_C
        taxMetaDataDTO.shipFromLine1 = taxProviderMDT.Address_L
        taxMetaDataDTO.shipFromLine2 = taxProviderMDT.Address_L
        taxMetaDataDTO.shipFromCity = taxProviderMDT.City__c;
        taxMetaDataDTO.shipFromState = taxProviderMDT.State__c;
        taxMetaDataDTO.shipFromCountry = taxProviderMDT.Country
        taxMetaDataDTO.shipFromZipCode = taxProviderMDT.Zip_Cod
        taxMetaDataDTO.defaultShippingTaxCode = taxProviderMDT.
        taxMetaDataDTO.taxationPolicy = taxProviderMDT.Taxation
        taxMetaDataDTO.shippingItemsTaxable = taxProviderMDT.Sh
        taxMetaDataDTO.defaultProductTaxCode = taxProviderMDT.D
        taxMetaDataDTO.useProductTaxCodes = taxProviderMDT.Cons
        taxMetaDataDTO.productTaxExmeption = taxProviderMDT.Pro
        taxMetaDataDTO.customerTaxExmeption = taxProviderMDT.Cu
        taxMetaDataDTO.transactionType = taxProviderMDT.Transac
        taxMetaDataDTO.documentType = taxProviderMDT.Document_T
        taxMetaDataDTO.isCommit = taxProviderMDT.Commit__c;
        taxMetaDataDTO.endPoint = taxProviderMDT.Service__r.End
        taxMetaDataDTO.httpMethod = taxProviderMDT.Service__r.H
        taxMetaDataDTO.mockedResponse = taxProviderMDT.Service_
        taxMetaDataDTO.serviceMode = taxProviderMDT.Service__r.
        taxMetaDataDTO.serviceTimeout = taxProviderMDT.Service_
        taxMetaDataDTO.serviceImplClass = taxProviderMDT.Servic
        taxMetaDataDTO.namedCredentials = taxProviderMDT.Servic
        taxProviders.put(dataKey, taxMetaDataDTO);
      }
    }
    return taxMetaDataDTO;
  }

  public static Tax_Provider__mdt getMetaDataObject() {
    return taxProviderMDT;
  }
}

```

<table>
<tr><td></td><td></td><td></td></tr>
<tr>
<td>TaxProviderFactory</td>
<td>This class works as a factory to create shipping provider singletons as per <strong>Impl Class</strong> configured in custom meta data record</td>
<td>

```
public with sharing class TaxProviderFactory {
  private static TaxProvider taxProvider;
  private static Map<String, TaxProvider> taxProviders = new Ma
  private TaxProviderFactory() {
  }
  public static TaxProvider getTaxProvider(TaxMetaData taxMetaD
    if (taxProviders.containsKey(taxMetaData.providerImplClass)
      TaxProvider = taxProviders.get(taxMetaData.providerImplCl
    } else {
      if (taxMetaData.providerImplClass != null) {
        Type t = Type.forName(taxMetaData.providerImplClass);
        TaxProvider = (TaxProvider) t.newInstance();
        TaxProviders.put(taxMetaData.providerImplClass, taxProv
      }
    }
    return taxProvider;
  }
}

```

</td>
</tr>
<tr>
<td>TaxProvider</td>
<td>This class provides common functionality to prepare a shipping request and should be further extended by a specific tax providers like Avalara , Vertex.</td>
<td>

You must extend this class & override following abstract methods

<strong>prepareCreateTransaction</strong>
<strong>prepareCommitTransaction</strong>
<strong>getTaxFromResponse</strong>

As per need, overriding following methods is optional

<strong>calculateTax()</strong>
<strong>prepareNamedCredentials()</strong>
<strong>setServiceDetails()</strong>
<strong>setRequestHeaders()</strong>

```
public abstract class TaxProvider {
  public TaxProvider() {
  }

  public virtual Map<String, String> prepareNamedCredentials(
    TaxProviderRequest taxRequest
  ) {
    Map<String, String> callOutRequest = new Map<String, Strin
    String endPoint =
      Constants.CALLOUT +
      taxRequest.taxMetaData.namedCredentials +
      taxRequest.taxMetaData.endPoint;
    callOutRequest.put(Constants.END_POINT, endPoint);
    // callOutRequest.put(Constants.USERNAME , Constants.CREDE
    // callOutRequest.put(Constants.PASSWORD , Constants.CREDE
    return callOutRequest;
  }
  public virtual Map<String, String> setServiceDetails(
    TaxProviderRequest taxRequest
  ) {
    Map<String, String> serviceDetails = new Map<String, Strin
    serviceDetails.put(
      Constants.HTTP_METHOD,
      taxRequest.taxMetaData.httpMethod
```

</td>
</tr>
</table>

```
    );
    serviceDetails.put(
      Constants.SERVICE_TIMEOUT,
      taxRequest.taxMetaData.serviceTimeout
    );
    serviceDetails.put(
      Constants.SERVICE_MODE,
      taxRequest.taxMetaData.serviceMode
    );
    serviceDetails.put(
      Constants.MOCKED_RESPONSE,
      taxRequest.taxMetaData.mockedResponse
    );
    return serviceDetails;
  }

  public virtual Map<String, String> setRequestHeaders(
    TaxProviderRequest taxRequest
  ) {
    Map<String, String> htttpRequestDetails = new Map<String,
    return htttpRequestDetails;
  }

  public virtual TaxProviderResponse calculateTax(
    TaxProviderRequest taxRequest
  ) {
    TaxProviderResponse taxes;
    Map<String, String> responseMap;
    Boolean calloutSuccess = true;
    try {
      Map<String, String> callOutRequest = new Map<String, Str
      callOutRequest.putAll(prepareNamedCredentials(taxRequest
      callOutRequest.putAll(setServiceDetails(taxRequest));
      prepareCreateTransaction(taxRequest, callOutRequest);

      responseMap = ServiceFactory.getService(
          taxRequest.taxMetaData.serviceImplClass
        )
        .makeExternalCallout(callOutRequest, setRequestHeaders
      if (
        !'201'.equals(responseMap.get(Constants.RESPONSE_REASO
        responseMap.isEmpty()
      ) {
        calloutSuccess = false;
      }
      String response = responseMap.get(Constants.SERVICE_RESP
      if (calloutSuccess) {
        taxes = getTaxFromResponse(response, taxRequest);
      }
    } catch (Exception expObj) {
      System.debug(
        'Exception due to error ====' +
          expObj.getMessage() +
          'at Line Number ====' +
          expObj.getLineNumber()
      );
      // WK_Exception.log(expObj, applicationName, moduleName,
      // methodName, supportData);
    }
    return taxes;
  }

  public abstract TaxProviderResponse getTaxFromResponse(
    String response,
    TaxProviderRequest taxRequest
  );
  public abstract void prepareCreateTransaction(
    TaxProviderRequest taxRequest,
    Map<String, String> callOutRequest
  );

  public void commitTransaction(TaxProviderRequest taxRequest)
    System.debug(
      '========================= Inside TaxProvider#calculateT
    );

    TaxProviderResponse taxes;
    Map<String, String> responseMap;
    Boolean calloutSuccess = true;
    try {
      Map<String, String> callOutRequest = new Map<String, Str
      callOutRequest.putAll(prepareNamedCredentials(taxRequest
      callOutRequest.putAll(setServiceDetails(taxRequest));
      prepareCommitTransaction(taxRequest, callOutRequest);

      responseMap = ServiceFactory.getService(
          taxRequest.taxMetaData.serviceImplClass
        )
        .makeExternalCallout(callOutRequest, setRequestHeaders
      if (
        !'201'.equals(responseMap.get(Constants.RESPONSE_REASO
        responseMap.isEmpty()
      ) {
        calloutSuccess = false;
      }
      String response = responseMap.get(Constants.SERVICE_RESP
      if (calloutSuccess) {
        taxes = getTaxFromResponse(response, taxRequest);
      }
    } catch (Exception expObj) {
      System.debug(
        'Exception due to error ====' +
          expObj.getMessage() +
          'at Line Number ====' +
          expObj.getLineNumber()
      );
    }
  }
  public abstract void prepareCommitTransaction(
    TaxProviderRequest taxRequest,
    Map<String, String> callOutRequest
  );
}


```

TaxProviderRequest	This class is used as a DTO to transfer request	

```
public with sharing class TaxProviderRequest {
  public TaxProviderRequest() {
```

	specific data & meta data between classes	}
		
		  public String cartId { get; set; }
		  public String street { get; set; }
		  public String city { get; set; }
		  public String state { get; set; }
		  public String postalCode { get; set; }
		  public String country { get; set; }
		  public Boolean hasMultipleShipments { get; set; }
		  public String taxTransacionType { get; set; }
		  public String customerTaxId { get; set; }
		  public TaxMetaData taxMetaData { get; set; }
		  public String webStoreId { get; set; }
		  public String taxTransactionId { get; set; }
		  public String customerExemptionCode { get; set; }
		  public String entityUseCode { get; set; }
		  public String currencyCode { get; set; }
		  public Map<String, TaxableCartItem> taxableCartItems { get; s
		
		  private Map<String, Object> additionalData = new Map<String,
		
		  public Object getData(String key) {
		    return additionalData.get(key);
		  }
		
		  public void addData(String key, Object value) {
		    additionalData.put(key, value);
		  }
		}

TaxProviderResponse	This class used as a DTO to transfer shipping rates back to shipping extension class	public with sharing class TaxProviderResponse {
		  public TaxProviderResponse() {
		  }
		  public Map<String, TaxableCartItem> taxableCartItems { get; se
		}

ServiceFactory	This class works as a factory to create HTTP Service singletons as per **Impl Class** configured in custom meta data record for service. For now there is no shipping specific class so it returns base class itself i.e. HTTPService	

<table>
<tr>
<td></td>
<td></td>
<td>

```
public with sharing class ServiceFactory {
  private static HTTPService service;
  private ServiceFactory() {
  }
  public static HTTPService getService(String className) {
    if (service == null) {
      if (String.isNotEmpty(className)) {
        Type t = Type.forName(className);
        service = (HTTPService) t.newInstance();
      } else {
        service = new HTTPService();
      }
    }
    return service;
  }
}

```

</td>
</tr>
<tr>
<td>HTTPService</td>
<td>This class is used to make a REST based callout. Currently supporting only JSON as data format.</td>
<td>

As per need, you can extend this class and override following methods :

createHttpRequest
setRequestHeaders
makeExternalCallout

```
public virtual class HTTPService {
  public HTTPService() {
  }
  public virtual HttpRequest createHttpRequest(
    Map<String, String> requestDetails
  ) {
    HttpRequest req = new HttpRequest();
    req.setEndpoint(requestDetails.get(Constants.END_POINT));
    req.setMethod(
      String.isNotBlank(requestDetails.get(Constants.HTTP_METH
        ? requestDetails.get(Constants.HTTP_METHOD)
        : Constants.HTTP_POST
    );
    Integer timeout = String.isNotBlank(
        requestDetails.get(Constants.SERVICE_TIMEOUT)
      )
      ? Integer.valueOf(requestDetails.get(Constants.SERVICE_T
      : Constants.HTTP_DEFAULT_TIMEOUT;
    req.setTimeout(timeout);
    return req;
  }

  public virtual void setRequestHeaders(
    HttpRequest req,
    Map<String, String> requestHeaders
  ) {
    if (!requestHeaders.isEmpty()) {
      for (String key : requestHeaders.keySet()) {
        req.setHeader(key, requestHeaders.get(key));
      }
    }
    if (!requestHeaders.containsKey(Constants.HTTP_HEADER_CONT
      req.setHeader(
        Constants.HTTP_HEADER_CONTENT_TYPE,
```

</td>
</tr>
</table>

```
        Constants.HTTP_HEADER_CONTENT_TYPE_JSON
      );
    }
  }

  public virtual Map<String, String> makeExternalCallout(
    Map<String, String> calloutRequestDetails,
    Map<String, String> requestHeaders
  ) {
    Map<String, String> responseMap = new Map<String, String>(
    try {
      HttpRequest req = createHttpRequest(calloutRequestDetail
      setRequestHeaders(req, requestHeaders);
      Http http = new Http();
      HTTPResponse res = null;
      if (!Test.isRunningTest()) {
        if (
          calloutRequestDetails.get(Constants.SERVICE_MODE)
            .toUpperCase()
            .equals(Constants.SERVICE_MODE_LIVE)
        ) {
          req.setbody(
            calloutRequestDetails.get(Constants.SERVICE_REQUES
          );
          res = http.send(req);
        } else {
          res = new HTTPResponse();
          res.setStatusCode(Constants.HTTP_200);
          res.setBody(calloutRequestDetails.get(Constants.MOCK
        }
      } else {
        res = new HTTPResponse();
        res.setStatusCode(Constants.HTTP_200);
      }
      if (res.getStatusCode() == Constants.HTTP_200) {
        responseMap.put(
          Constants.HTTP_RESPONSE_STATUS,
          Constants.HTTP_RESPONSE_STATUS_SUCCESS
        );
        responseMap.put(
          Constants.RESPONSE_REASON_CODE,
          String.valueOf(res.getstatusCode())
        );
        responseMap.put(Constants.SERVICE_RESPONSE_BODY, res.g
      } else {
        responseMap.put(
          Constants.HTTP_RESPONSE_STATUS,
          Constants.HTTP_RESPONSE_STATUS_ERROR
        );
        responseMap.put(
          Constants.RESPONSE_REASON_CODE,
          String.valueOf(res.getstatusCode())
        );
        responseMap.put(Constants.SERVICE_RESPONSE_BODY, res.g
      }
    } catch (Exception expObj) {
      responseMap.put(
        Constants.HTTP_RESPONSE_STATUS,
        Constants.HTTP_RESPONSE_STATUS_ERROR
      );
      System.debug(
        'Exception due to error ====' +
          expObj.getMessage() +
          'at Line Number ====' +
          expObj.getLineNumber()
      );
      // WK_Exception.log(expObj, applicationName, moduleName,
      // methodName, supportData);
    }
    return responseMap;
  }

  public virtual void setAuth() {
  }
  public virtual void setAuthHeaders() {
  }
}

```