Avalara User Guide

The key components involved in Avalara Integration module are

* AvalaraConnectionUtility – This component will provide an option to configure the connection details like Account Number, Company code, License key, URL.
* AvalaraAddrvalidationService – This component will provide an option to configure list of countries and an option to enable/disable address validation. This component is responsible for address validation from Avalara API.
* AvalaraTaxCalculationService - This component will have the mappings of customer address and an option to enable/disable tax calculation. This component is responsible for tax calculation of orders.
* CSRAvalaraTaxCalculationService - This component will have the mappings of customer address and an option to enable/disable tax calculation. This component is responsible for tax calculation of orders from the CSC application.

# Initial Setup

Create an account in Avalara, note down license key, account number and company code to be used by the ATG application. Please refer to Avalara documentation for the same.

# Avalara Connection Configuration

Define following properties with appropriate values to make connection with Avalara API into the com/avalara/tax/csrintegration/utils/AvalaraConnectionUtility component.

url=https://development.avalara.net/

accountNumber=1100088509

licenseKey=8A92FA83E23DB16F

companyCode=NVIZ

(Note: The data provided is only for testing account.)

This component is responsible for the connection to Avalara, any ATG customer of Avalara can easily change these values.

# Address Validation

To validate the customer address from ATG, make a call to Avalara API by setting address validation property to true.

### Implementation

Steps involved in making a call to this integration API from ATG web commerce pipeline processor is as below,

* Define a property of AvalaraConnectionUtility component in AvalaraAddrvalidationService component.

avataxConnUtil=/com/avalara/tax/integration/utils/AvalaraConnectionUtility

* Define the addrValidationRequired property in AvalaraAddrvalidationService component and set the value to true to enable address validation.
* Define countryList property in component and set comma separated values of countries.
* Define a property AvalaraAddrvalidationService component in AvalaraAddressVerificationProcessor component.
* Make a call to AvalaraAddrvalidationService component validateAddress() method to avalara API from our ATG component AvalaraAddressVerificationProcessor which is mentioned in /atg/commerce/order/processor/VerifyOrderAddresses processor.

### Testing

* Enter Shipping and Billing address while placing an order.
* If an address in invalid it will redirect to billing.jsp page otherwise it will proceed to order conformation page.

# Tax Calculation

Tax calculation is done by setting taxCalculationReqd property to true. Also set the originAddress.

### Implementation

Steps involved in making a call to this integration API from ATG are as below,

* Define a property of AvalaraConnectionUtility component in AvalaraTaxCalculationService component.

avataxConnUtil=/com/avalara/tax/integration/utils/AvalaraConnectionUtility

* Define a property taxCalculationReqd in AvalaraTaxCalculationService component and set the value to true.

Define a property originAddress in AvalaraTaxCalculationService component and set the value as map.

originAddress=\

AddressCode=Origin,\

Line1=Avalara,\

Line2=900 Winslow Way,\

Line3=Suite 100,\

City=Bainbridge Island,\

Region=WA,\

PostalCode=98110,\

Country=USA

(Note: The data provided is only for Test purpose.)

* Define a property productCatalog in AvalaraTaxCalculationService component.

productCatalog=/atg/commerce/catalog/ProductCatalog

* Define a property profileTools in AvalaraTaxCalculationService component.

profileTools=/atg/userprofiling/ProfileTools

* Define a property for freight items.

shippingAmtTaxCode=FR

* Override the /atg/commerce/pricing/calculators/TaxProcessorTaxCalculator component.

$class= com.avalara.tax.integration.AvalaraTaxProcessTaxCalculator

Define a property taxProcessor and set a value as following

taxProcessor=/atg/commerce/payment/AvalaraTaxProcessor

* Define a property AvalaraTaxCalculationService component in AvalaraTaxProcessor component and AvalaraCommitTaxSink component.
* Make a call to AvalaraTaxCalculationService component getTaxForOrder() method, this will internally connect to Avalara API and gets the tax for the order and the transaction is made in to Avalara whose state is uncommitted.
* For committing the transaction in to Avalara , order has to be placed by the user in the ATG system.
* For committing, make a call to AvalaraTaxCalculationService component commitTaxForOrder() method Asynchronously.
* Send JMS message from the runProcess() method of ProcCommitTax processor, and this will be received by AvalaraCommitTaxSink and from receiveMessage() method make a call to AvalaraTaxCalculationService component commitTaxForOrder() method for committing the transaction in to Avalara.

### Testing

* Create an order from ATG and go to Billing page. On right hand side of page Tax can be seen.
* Log in to <https://admin-development.avalara.net/login.aspx>. Open Transaction tab and check the record is generated with status “uncommitted” in Avalara panel for an order.
* Place an order through ATG.
* Log in to <https://admin-development.avalara.net/login.aspx>. Open Transaction tab and check the record is generated with status “committed” in Avalara panel for an order.

# Tax Calculation for Return Orders

Tax Calculation for Return orders (i.e. ATG CSC) is as same as tax calculation in CRS with some more properties added.

### Implementation

* Define a property of AvalaraConnectionUtility in CSRAvalaraTaxCalculationService component.

avataxConnUtil=/com/avalara/tax/csrintegration/utils/AvalaraConnectionUtility

* Override ReturnManager Component with following class.

$class=com.avalara.tax.csrintegration.core.AvalaraReturnManager

* Define a property of ReturnManger in CSRAvalaraTaxCalculationService component

returnManager=/atg/commerce/custsvc/returns/ReturnManager

* Define other properties like productCatalog,profileTools, taxCalculationReqd as same as defined in AvalaraTaxCalculationService.
* Define a property CSRAvalaraTaxCalculationService component in CSRAvalaraTaxProcessor component and CSRAvalaraCommitTaxSink component.
* Make a call to CSRAvalaraTaxCalculationService component getTaxForOrder() method, this will internally connect to Avalara API and gets the tax for the return order and the transaction is made in to Avalara whose state is uncommitted with the version associated with the original order id.
* For committing the transaction in to Avalara , return order has to be submitted by the ATG CSC user.
* For committing, make a call to CSRAvalaraTaxCalculationService component commitTaxForOrder() method Asynchronously.
* Send JMS message from the createRequestItem()method of AvalaraReturnManager class, and this will be received by CSRAvalaraCommitTaxSink and from receiveMessage() method make a call to CSRAvalaraTaxCalculationService component commitTaxForOrder() method for committing the transaction in to Avalara.

### Testing

* Login to ATG CSC and select an item/all items from an order to return a. Click to return. Make sure an order has status “Fulfilled” before return process.
* Login to <https://admin-development.avalara.net/login.aspx>. Select Transaction tab and check the record is generated with status “uncommitted” in Avalara panel for an order. The DocType of the record would be ReturnInvoice since it is Return order.
* The document code will be generated in sequential manner. Like orginalorderid.1, originalorderid.2 etc.
* Now submit the return order in ATG CSC.
* Now again Login to <https://admin-development.avalara.net/login.aspx> and check the record is generated with status “committed” in Avalara panel for an order.

# Tax Calculation for Cancel Order

### Implementation

* Override handleModifyOrder process chain in fulfillmentpipeline.xml and append AvalaraCancelTax processor.

<pipelinelink transaction=*"TX\_MANDATORY"* name=*"updateOrderRepository1"* xml-combine=*"replace"*>

<processor jndi=*"/atg/commerce/fulfillment/processor/UpdateOrderRepository"*/>

<transition returnvalue=*"1"* link=*"cancelTax"*/>

</pipelinelink>

<pipelinelink transaction=*"TX\_MANDATORY"* name=*"cancelTax"*>

<processor jndi=*"/com/avalara/tax/csrintegration/processor/AvalaraCancelTax"*/>

<transition returnvalue=*"1"* link=*"sendModifyOrderNotification1"*/>

</pipelinelink>

* This is appended to cancel the being processed state Orders from ATG CSC.
* Define a property CSRAvalaraTaxCalculationService component, in AvalaraCancelTax component.
* For committing, make a call to CSRAvalaraTaxCalculationService component cancelTaxForOrder() method Asynchronously.
* Send JMS message from the AvalaraCancelTax component runProcess() method, this message will be received by CSRAvalaraCancelTaxSink component and from receiveMessage() method make a call to CSRAvalaraTaxCalculationService component cancelTaxForOrder() method for void the transaction in to Avalara.

### Test

* Login to ATG CSC and Cancel an order. Make sure the order has status “Being Processed”
* Login to <https://admin-development.avalara.net/login.aspx>. Go to Transaction tab and check the status of the record for corresponding order is changed to void from committed.

# Tax Calculation Call for Delete order

### Implementation

* Override the /atg/commerce/custsvc/order/CancelOrderService component and set the class property

$class=com.avalara.tax.csrintegration.core.AvalaraCancelOrderService

* Define a property of CSRAvalaraTaxCalculation component and set the value

avaTaxService=/com/avalara/tax/csrintegration/service/CSRAvalaraTaxCalculationService

* This Component overrides the deleteOrder() method and the customize code will get executed which cancels the record from avalara.
* From this deleteOrder() method make a call to CSRAvalaraTaxCalculationService component cancelTaxForOrder() method with doctype as docDeleted.

### Test

* Login to ATG CSC and select an order to delete. Make sure an order has status “uncommitted”.
* Login to <https://admin-development.avalara.net/login.aspx>. Go to Transaction tab and check ­­the record for respected order is deleted from Avalara panel.

# Tax Code for the Items

The tax code can be set for a product or group of products. The tax code cane be set through Avalara Console panel. For the shipping items the tax code will be “FR” and it should be configurable.

### Implementation

* Append the atg/commerce/catalog/custom/customCatalog.xml to add the Tax code for the products.

<?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>

<!DOCTYPE gsa-template

PUBLIC "-//Art Technology Group, Inc.//DTD General SQL Adapter//EN"

"http://www.atg.com/dtds/gsa/gsa\_1.0.dtd">

<gsa-template>

<header>

<name>Dynamo Commerce Server Default Catalog</name>

<author>nvizion</author>

</header>

<item-descriptor name=*"product"* display-name-resource=*"itemDescriptorProduct"* context-membership-property=*"siteIds"*

sub-type-property=*"type"* display-property=*"displayName"* version-property=*"version"*

item-cache-size=*"1000"* query-cache-size=*"1000"* id-space-name=*"product"* xml-combine=*"append"*>

<table name=*"product\_taxcode\_map"* id-column-name=*"product\_id"* type=*"auxiliary"* >

<property name=*"taxCode"* data-type=*"string"* category = *"Tax Code Mapping"* column-name=*"tax\_code"* display-name-resource=*"Tax Code"*></property>

</table>

</item-descriptor>

</gsa-template>

### Test

* Open ACC panel (ATG Control Center). Select an Item and assign Tax Code.
* Place an order form CRS with those items who have tax code applied.
* Login to <https://admin-development.avalara.net/login.aspx>. Go to Transaction tab and check record for a placed order.
* Select the record and check the tax code is applied on item level.
* For the separate shipping line item the tax code should be “FR”, and this value is configured in our service component. It will not come from the product table taxCode property.

# Tax Calculation with Exemption Number

Exemption Number and entityUseCode can be set by System Administrator in ATG System. Both these properties are at profile level.

### Implementation

* Override atg/userprofiling/userProfile.xml and append following item-descriptor to add exemption code.

<item-descriptor name=*"user"* default=*"true"* sub-type-property=*"userType"* display-property=*"login"* display-name-resource=*"itemDescriptorUser"* xml-combine=*"append"*>

<table name=*"dps\_user\_tax\_exemption\_details"* id-column-name=*"id"* type=*"auxiliary"* >

<property name=*"entity\_use\_code"* data-type=*"string"* column-name=*"entity\_use\_code"* display-name-resource=*"Entity Use Code"*></property>

<property name=*"exemption\_number"* data-type=*"string"* column-name=*"exemption\_number"* display-name-resource=*"Exemption Number"*></property>

</table>

</item-descriptor>

### Test

* Login to the ACC (ATG control Center) and select customer profile to whom you need to set the Exemption Number.
* select the exemption\_number Field and assign excemption number to the profile.
* Select the entity\_use\_code and assign the entityuseCode to the profile.
* Place an order through ATG
* Login to <https://admin-development.avalara.net/login.aspx>. Go to Transaction tab and check record for a placed order.
* Check the record on item level. The amount is exempted from the tax and no tax is applied for that order.