

SF-LWR Extension Framework : Avalara Tax Integration

[Functional Overview](#)

[Technical Details](#)

[Sequence Diagram](#)

[Configuration details](#)

[Record - Avalara \(Tax Provider \)](#)

[Record - Avalara Tax Calc Service \(HTTP Service\)](#)

[Source Code](#)

Functional Overview

This document provides details specific to integrating with Avalara.

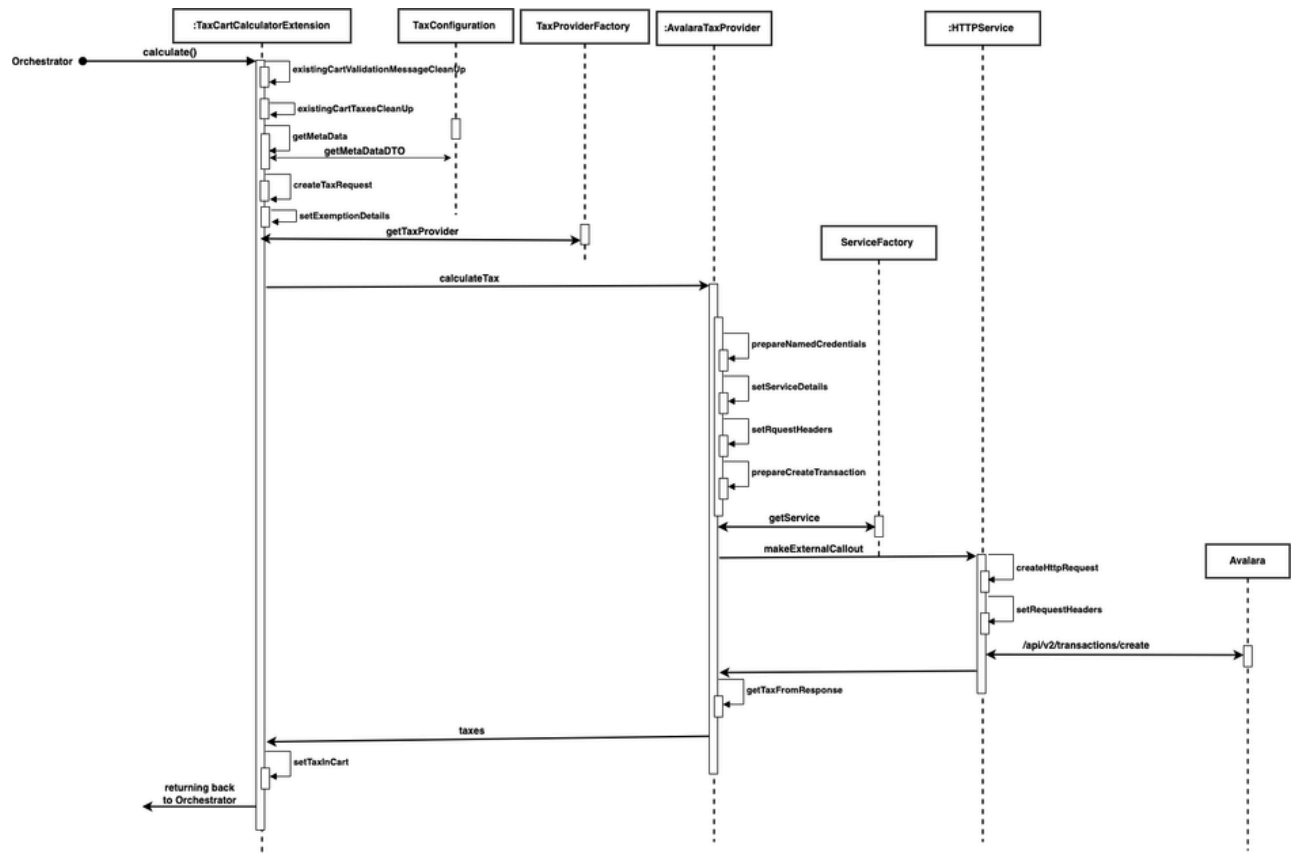
For common details you can refer to parent doc [here](#).

Technical Details

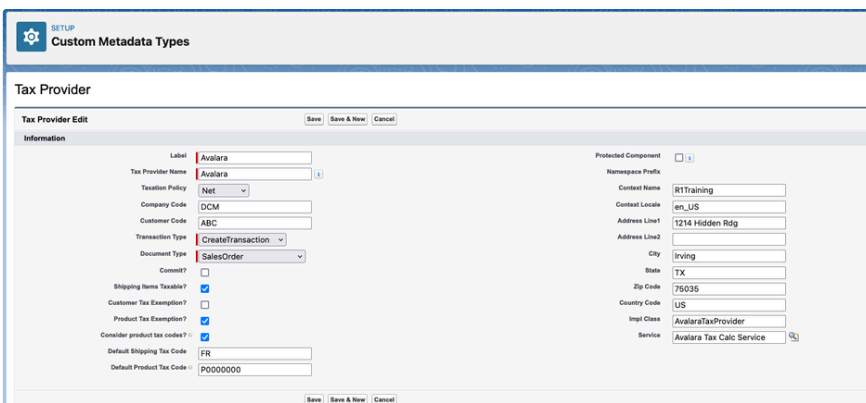
This integration implementation has two parts :

- **Configuration**
 - Custom Meta Data Records
 - Avalara
 - Avalara Tax Calc Service
 - Named Credentials
 - Avalara Credentials (Legacy)
- **Source Code**
 - AvalaraTaxProvider
 - AvalaraTaxRequest
 - AvalaraTaxResponse

Sequence Diagram



Configuration details

	Section	Task	Details
1	Salesforce Org	Tax Provider - Create a record for Avalara	<p>Create a record Avalara for Custom Meta Data Type “Tax Provider”</p> 
2	Salesforce Org	HTTP Service - Create a record for Avalara	<p>Create a record - Avalara Tax Calc Service of Custom Meta Data Type “HTTP Service”</p>

3	Salesforce Org	Create Named Credentials	<p>Create a Named Credentials(legacy) for storing Avalara API details</p> <p>Set URL - https://sandbox-rest.avatax.com</p> <p>https://rest.avatax.com</p> <p>Turn on - Generate Authorization Header</p>

Record - Avalara (Tax Provider)

Field Label	Field Value
Context Name	<Webstore name> e.g. R1Training
Context Locale	<locale> e.g. en_US
Impl Class	AvalaraTaxProvider
Address Line1	75 New Hampshire Ave
Address Line2	
City	Portsmouth
State	NH
Zip Code	03801
Country Code	US
Company Code	DCM

Customer Code	ABC
Taxation Policy	Net Gross
Transaction Type	CreateTransaction AdjustTransaction CommitTransaction RefundTransaction
Document Type	SalesOrder SalesInvoice PurchaseOrder PurchaseInvoice ReturnOrder ReturnInvoice InventoryTransferOrder InventoryTransferInvoice
Commit?	True/False
Shipping Items Taxable?	True/False
Consider product tax codes?	True/False
Customer Tax Exemption?	True/False
Product Tax Exemption?	True/False
Default Product Tax Code	P0000000
Default Shipping Tax Code	FR
Request Template	1
Service	Avalara Tax Calc Service

Record - Avalara Tax Calc Service (HTTP Service)

Field Label	Field Value
End Point	/api/v2/transactions/create
HTTP Method	POST
Impl Class	<i>This is optional so could be left blank</i>
Named Credentials	Avalara
Service Mode	Live Mocked
Timeout	5000 (ms)

Success Codes	200,201
Error Codes	400,403,404,500,503
Auth Codes	401
Mocked Response	

Source Code

Class	Details	Code
AvalaraTaxProvider	<p>This class extends ShippingProvider class & implements following methods -</p> <p>prepareCreateTransaction</p> <p>prepareCommitTransaction</p> <p>getTaxFromResponse</p>	<pre> 1 public without sharing class AvalaraTaxProvider extends TaxPro 2 public AvalaraTaxProvider() { 3 } 4 5 public override void prepareCreateTransaction(6 TaxProviderRequest taxRequest, 7 Map<String, String> callOutRequest 8) { 9 Boolean hasMultipleShipments = taxRequest.hasMultipleShipm 10 taxRequest.hasMultipleShipments == true 11 ? true 12 : false; 13 14 AvalaraTaxRequest avalaraRequest = new AvalaraTaxRequest() 15 avalaraRequest.type = taxRequest.taxTransacionType; 16 avalaraRequest.currencyCode = taxRequest.currencyCode; 17 avalaraRequest.commit_Z = taxRequest.taxMetaData.isCommit; 18 // if (19 // String.isNotBlank(avalaraRequest.type) && 20 // avalaraRequest.type.contains('Invoice') 21 //) { 22 // avalaraRequest.commit_Z = true; 23 // }else{ 24 // avalaraRequest.commit_Z = false; 25 // } 26 27 if (String.isNotBlank(taxRequest.taxMetaData.companyCode)) 28 avalaraRequest.companyCode = taxRequest.taxMetaData.comp 29 } 30 31 if (String.isNotBlank(taxRequest.taxMetaData.customerCode)) 32 avalaraRequest.customerCode = taxRequest.taxMetaData.cus 33 } 34 35 if (String.isNotBlank(taxRequest.customerTaxId)) { 36 avalaraRequest.businessIdentificationNo = taxRequest.cus 37 } 38 39 if (String.isNotBlank(taxRequest.customerExemptionCode)) { 40 avalaraRequest.exemptionNo = taxRequest.customerExemptio 41 } 42 43 if (String.isNotBlank(taxRequest.entityUseCode)) { 44 avalaraRequest.entityUseCode = taxRequest.entityUseCode; 45 } 46 </pre>

```

47     avalaraRequest.date_Z = System.now() + '';
48     //taxRequest.taxableCartItems = null;
49     if (
50         taxRequest.taxableCartItems != null &&
51         taxRequest.taxableCartItems.keySet().size() > 0
52     ) {
53         avalaraRequest.lines = prepareRequestFromExistingData(
54             taxRequest,
55             hasMultipleShipments
56         );
57     } else {
58         avalaraRequest.lines = prepareRequestFromDB(
59             taxRequest,
60             hasMultipleShipments
61         );
62     }
63
64     if (!hasMultipleShipments) {
65         AvalaraTaxRequest.Addresses addresses = new AvalaraTaxRe
66         addresses.shipFrom = prepareShipFromAddress(taxRequest);
67         addresses.shipTo = prepareShipToAddress(
68             taxRequest.street,
69             taxRequest.city,
70             taxRequest.state,
71             taxRequest.postalCode,
72             taxRequest.country
73         );
74         avalaraRequest.addresses = addresses;
75     }
76
77     String avalaraRequestBody = JSON.serialize(avalaraRequest,
78     avalaraRequestBody = avalaraRequestBody.replaceAll('_Z', '
79     callOutRequest.put(Constants.SERVICE_REQUEST_BODY, avalara
80     System.debug('===== avalaraRequest : '+avalaraRequestBody
81 }
82
83 public virtual override TaxProviderResponse getTaxFromRespon
84     String strAvalaraResponseBody,
85     TaxProviderRequest tpRequest
86 ) {
87     TaxProviderResponse tpResponse = new TaxProviderResponse()
88     Map<String, LineItemTaxDetails> taxes = new Map<String, Li
89     Map<String, TaxableCartItem> taxableCartItems = tpRequest.
90
91     TaxProviderResponse shippingResponse;
92     System.debug('===== strAvalaraResponseBody : '+strAvalara
93     if (String.isNotBlank(strAvalaraResponseBody)) {
94         AvalaraTaxResponse avalaraTaxResponse = AvalaraTaxRespon
95         strAvalaraResponseBody
96     );
97     if (
98         avalaraTaxResponse.lines != null &
99         avalaraTaxResponse.lines.size() > 0
100     ) {
101         for (AvalaraTaxResponse.Lines line : avalaraTaxRespons
102             String lineNumber = line.lineNumber;
103             LineItemTaxDetails liTaxDetails = new LineItemTaxDet
104             liTaxDetails.tax = 0;
105             liTaxDetails.rate = 0;
106             if (line.tax > 0) {

```

```

107         liTaxDetails.tax = line.tax;
108     }
109     if (line.taxableAmount > 0) {
110         liTaxDetails.rate = liTaxDetails.tax / line.taxabl
111     }
112     taxableCartItems.get(lineNumber).lineItemTexas.add(1
113     }
114     tpResponse.taxableCartItems = taxableCartItems;
115 }
116 }
117 return tpResponse;
118 }
119
120 private List<AvalaraTaxRequest.Lines> prepareRequestFromExis
121     TaxProviderRequest taxRequest,
122     Boolean hasMultipleShipments
123 ) {
124     List<AvalaraTaxRequest.Lines> lines = new List<AvalaraTaxR
125     for (String id : taxRequest.taxableCartItems.keySet()) {
126         TaxableCartItem cartItem = taxRequest.taxableCartItems.g
127         AvalaraTaxRequest.Lines line = prepareLine(
128             id,
129             cartItem.sku,
130             cartItem.amount,
131             null,
132             cartItem.taxClassId,
133             cartItem.productExemptionCode,
134             cartItem.entityUseCode,
135             taxRequest,
136             false
137         );
138
139         if (hasMultipleShipments) {
140             AvalaraTaxRequest.Addresses addresses = new AvalaraTax
141             addresses.shipFrom = prepareShipFromAddress(taxRequest
142             addresses.shipTo = prepareShipToAddress(
143                 cartItem.street,
144                 cartItem.city,
145                 cartItem.state,
146                 cartItem.postalCode,
147                 cartItem.country
148             );
149             line.addresses = addresses;
150         }
151         lines.add(line);
152     }
153     return lines;
154 }
155
156 private List<AvalaraTaxRequest.Lines> prepareRequestFromDB(
157     TaxProviderRequest taxRequest,
158     Boolean hasMultipleShipments
159 ) {
160
161     String cartId = taxRequest.cartId;
162     taxRequest.taxableCartItems = new Map<String, TaxableCartI
163     List<AvalaraTaxRequest.Lines> lines = new List<AvalaraTaxR
164     String query = 'SELECT Id, Sku, TotalLineAmount, Quantity,
165     if (hasMultipleShipments) {
166         query += ', CartDeliveryGroup.Id, CartDeliveryGroup.Delive

```

```

167     }
168     if(taxRequest.taxMetaData.useProductTaxCodes){
169         query += ',Product2.Tax_Class_Id__c';
170     }
171     if(taxRequest.taxMetaData.productTaxExemption){
172         query += ',Product2.Taxable__c,Product2.Entity_Use_Code__c';
173     }
174     query += ' FROM CartItem WHERE cartId=:cartId';
175     if (!taxRequest.taxMetaData.shippingItemsTaxable) {
176         query += ' AND Type = \'Product\'';
177     }
178     for (CartItem cartItem : Database.query(query)) {
179         AvalaraTaxRequest.Lines line = prepareLine(
180             cartItem.Id,
181             cartItem.SKU !=null ? cartItem.SKU : cartItem.Product2.SKU,
182             cartItem.TotalLineAmount,
183             cartItem.Type,
184             taxRequest.taxMetaData.useProductTaxCodes && cartItem.Product2.Taxable,
185             taxRequest.taxMetaData.productTaxExemption ? cartItem.Product2.Taxable : false,
186             taxRequest.taxMetaData.productTaxExemption ? cartItem.Product2.Entity_Use_Code : null,
187             taxRequest,
188             true
189         );
190         if (hasMultipleShipments) {
191             AvalaraTaxRequest.Addresses addresses = new AvalaraTaxRequest.Addresses();
192             addresses.shipFrom = prepareShipFromAddress(taxRequest, cartItem);
193             addresses.shipTo = prepareShipToAddress(
194                 cartItem.CartDeliveryGroup.DeliverToStreet,
195                 cartItem.CartDeliveryGroup.DeliverToCity,
196                 cartItem.CartDeliveryGroup.DeliverToState,
197                 cartItem.CartDeliveryGroup.DeliverToPostalCode,
198                 cartItem.CartDeliveryGroup.DeliverToCountry
199             );
200             line.addresses = addresses;
201         }
202         lines.add(line);
203     }
204     return lines;
205 }

206
207 private AvalaraTaxRequest.Lines prepareLine(
208     ID id,
209     String sku,
210     Decimal taxableAmount,
211     String type,
212     String taxClassId,
213     String exemptionCode,
214     String entityUseCode,
215     TaxProviderRequest taxRequest,
216     Boolean setInRequestToo
217 ) {
218     AvalaraTaxRequest.Lines line = new AvalaraTaxRequest.Lines();
219     line.number_Z = id;
220     line.itemCode = sku;
221     line.amount = taxableAmount;
222     if (String.isNotBlank(taxRequest.taxMetaData.taxationPolicy))
223         line.taxIncluded = taxRequest.taxMetaData.taxationPolicy;
224     else
225         line.taxIncluded = 'net';
226     return line;
}

```


		<pre> 227 : true; 228 } 229 line.taxCode = taxClassId; 230 line.exemptionCode = exemptionCode; 231 line.entityUseCode = entityUseCode; 232 if (setInRequestToo) { 233 TaxableCartItem tcItem = new TaxableCartItem(); 234 tcItem.id = id; 235 tcItem.amount = taxableAmount; 236 tcItem.sku = sku; 237 tcItem.lineItemType = type; 238 taxRequest.taxableCartItems.put(id, tcItem); 239 } 240 241 return line; 242 } 243 244 public override void prepareCommitTransaction(TaxProviderReq 245 246 } 247 AvalaraTaxRequest.ShipFrom prepareShipFromAddress(248 TaxProviderRequest taxRequest 249) { 250 AvalaraTaxRequest.ShipFrom address = new AvalaraTaxRequest 251 252 address.line1 = taxRequest.taxMetaData.shipFromLine1; 253 address.city = taxRequest.taxMetaData.shipFromCity; 254 address.region = taxRequest.taxMetaData.shipFromState; 255 address.postalCode = taxRequest.taxMetaData.shipFromZipCod 256 address.country = taxRequest.taxMetaData.shipFromCountry; 257 258 return address; 259 } 260 261 AvalaraTaxRequest.ShipFrom prepareShipToAddress(262 String street, 263 String city, 264 String state, 265 String postalCode, 266 String country 267) { 268 AvalaraTaxRequest.ShipFrom address = new AvalaraTaxRequest 269 address.line1 = street; 270 address.city = city; 271 address.region = state; 272 address.postalCode = postalCode; 273 address.country = country; 274 return address; 275 } 276 } 277 </pre>
AvalaraTaxRequest	This class is a DTO used to prepare Avalara tax calculation request which is posted to UPS to get shipping details	<pre> 1 public class AvalaraTaxRequest { 2 public class Addresses { 3 public ShipFrom shipFrom { get; set; } 4 public ShipFrom shipTo { get; set; } 5 public Addresses() { 6 } 7 public Addresses(JSONParser parser) { 8 while (parser.nextToken() != System.JSONToken.END_OBJECT </pre>

```

9         if (parser.getCurrentToken() == System.JSONToken.FIELD
10             String text = parser.getText();
11             if (parser.nextToken() != System.JSONToken.VALUE_NULL) {
12                 if (text == 'shipFrom') {
13                     shipFrom = new ShipFrom(parser);
14                 } else if (text == 'shipTo') {
15                     shipTo = new ShipFrom(parser);
16                 } else {
17                     System.debug(
18                         LoggingLevel.WARN,
19                         'Addresses consuming unrecognized property: '
20                     );
21                     consumeObject(parser);
22                 }
23             }
24         }
25     }
26 }
27 }
28
29 public List<Lines> lines { get; set; }
30 public Addresses addresses { get; set; }
31 public String code { get; set; }
32 public String type { get; set; }
33 public String companyCode { get; set; }
34 public String date_Z { get; set; } // in json: date
35 public String salespersonCode { get; set; }
36 public String customerCode { get; set; }
37 public String customerUsageType { get; set; }
38 public String entityUseCode { get; set; }
39 public Integer discount { get; set; }
40 public String exemptionNo { get; set; }
41 public String reportingLocationCode { get; set; }
42 public String purchaseOrderNo { get; set; }
43 public String currencyCode { get; set; }
44 public String description { get; set; }
45 public String referenceCode { get; set; }
46 public Boolean commit_Z { get; set; } // in json: commit
47 public String batchCode { get; set; }
48 public String serviceMode { get; set; }
49 public Integer exchangeRate { get; set; }
50 public String exchangeRateEffectiveDate { get; set; }
51 public String exchangeRateCurrencyCode { get; set; }
52 public String posLaneCode { get; set; }
53 public String businessIdentificationNo { get; set; }
54 public Boolean isSellerImporterOfRecord { get; set; }
55 public String email { get; set; }
56 public String debugLevel { get; set; }
57 public String customerSupplierName { get; set; }
58 public Integer dataSourceId { get; set; }
59 public String deliveryTerms { get; set; }
60 public TaxOverride taxOverride { get; set; }
61 public List<Parameters> parameters { get; set; }
62 public List<UserDefinedFields> userDefinedFields { get; set; }
63
64 public AvalaraTaxRequest() {
65 }
66 public AvalaraTaxRequest(JSONParser parser) {
67     while (parser.nextToken() != System.JSONToken.END_OBJECT)
68         if (parser.getCurrentToken() == System.JSONToken.FIELD_N

```

```
69     String text = parser.getText();
70     if (parser.nextToken() != System.JSONToken.VALUE_NULL)
71         if (text == 'lines') {
72             lines = arrayOfLines(parser);
73         } else if (text == 'addresses') {
74             addresses = new Addresses(parser);
75         } else if (text == 'code') {
76             code = parser.getText();
77         } else if (text == 'type') {
78             type = parser.getText();
79         } else if (text == 'companyCode') {
80             companyCode = parser.getText();
81         } else if (text == 'date') {
82             date_Z = parser.getText();
83         } else if (text == 'salespersonCode') {
84             salespersonCode = parser.getText();
85         } else if (text == 'customerCode') {
86             customerCode = parser.getText();
87         } else if (text == 'customerUsageType') {
88             customerUsageType = parser.getText();
89         } else if (text == 'entityUseCode') {
90             entityUseCode = parser.getText();
91         } else if (text == 'discount') {
92             discount = parser.getIntegerValue();
93         } else if (text == 'exemptionNo') {
94             exemptionNo = parser.getText();
95         } else if (text == 'reportingLocationCode') {
96             reportingLocationCode = parser.getText();
97         } else if (text == 'purchaseOrderNo') {
98             purchaseOrderNo = parser.getText();
99         } else if (text == 'currencyCode') {
100             currencyCode = parser.getText();
101         } else if (text == 'description') {
102             description = parser.getText();
103         } else if (text == 'referenceCode') {
104             referenceCode = parser.getText();
105         } else if (text == 'commit') {
106             commit_Z = parser.getBooleanValue();
107         } else if (text == 'batchCode') {
108             batchCode = parser.getText();
109         } else if (text == 'serviceMode') {
110             serviceMode = parser.getText();
111         } else if (text == 'exchangeRate') {
112             exchangeRate = parser.getIntegerValue();
113         } else if (text == 'exchangeRateEffectiveDate') {
114             exchangeRateEffectiveDate = parser.getText();
115         } else if (text == 'exchangeRateCurrencyCode') {
116             exchangeRateCurrencyCode = parser.getText();
117         } else if (text == 'posLaneCode') {
118             posLaneCode = parser.getText();
119         } else if (text == 'businessIdentificationNo') {
120             businessIdentificationNo = parser.getText();
121         } else if (text == 'isSellerImporterOfRecord') {
122             isSellerImporterOfRecord = parser.getBooleanValue();
123         } else if (text == 'email') {
124             email = parser.getText();
125         } else if (text == 'debugLevel') {
126             debugLevel = parser.getText();
127         } else if (text == 'customerSupplierName') {
128             customerSupplierName = parser.getText();
```

```

129         } else if (text == 'dataSourceId') {
130             dataSourceId = parser.getIntegerValue();
131         } else if (text == 'deliveryTerms') {
132             deliveryTerms = parser.getText();
133         } else if (text == 'taxOverride') {
134             taxOverride = new TaxOverride(parser);
135         } else if (text == 'parameters') {
136             parameters = arrayOfParameters(parser);
137         } else if (text == 'userDefinedFields') {
138             userDefinedFields = arrayOfUserDefinedFields(parser);
139         } else {
140             System.debug(
141                 LoggingLevel.WARN,
142                 'AvalaraTaxRequest consuming unrecognized proper
143             );
144             consumeObject(parser);
145         }
146     }
147 }
148 }
149 }
150
151 public class TaxAmountByTaxTypes {
152     public String taxTypeId { get; set; }
153     public Integer taxAmount { get; set; }
154
155     public TaxAmountByTaxTypes() {
156     }
157     public TaxAmountByTaxTypes(JSONParser parser) {
158         while (parser.nextToken() != System.JSONToken.END_OBJECT
159             if (parser.getCurrentToken() == System.JSONToken.FIELD
160                 String text = parser.getText();
161                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
162                     if (text == 'taxTypeId') {
163                         taxTypeId = parser.getText();
164                     } else if (text == 'taxAmount') {
165                         taxAmount = parser.getIntegerValue();
166                     } else {
167                         System.debug(
168                             LoggingLevel.WARN,
169                             'TaxAmountByTaxTypes consuming unrecognized pr
170                         );
171                         consumeObject(parser);
172                     }
173                 }
174             }
175         }
176     }
177 }
178
179 public class TaxOverride {
180     public String type { get; set; }
181     public Integer taxAmount { get; set; }
182     public String taxDate { get; set; }
183     public String reason { get; set; }
184     public List<TaxAmountByTaxTypes> taxAmountByTaxTypes { get
185
186     public TaxOverride() {
187     }
188     public TaxOverride(JSONParser parser) {

```

```

189         while (parser.nextToken() != System.JSONToken.END_OBJECT
190             if (parser.getCurrentToken() == System.JSONToken.FIELD
191                 String text = parser.getText();
192                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
193                     if (text == 'type') {
194                         type = parser.getText();
195                     } else if (text == 'taxAmount') {
196                         taxAmount = parser.getIntegerValue();
197                     } else if (text == 'taxDate') {
198                         taxDate = parser.getText();
199                     } else if (text == 'reason') {
200                         reason = parser.getText();
201                     } else if (text == 'taxAmountByTaxTypes') {
202                         taxAmountByTaxTypes = arrayOfTaxAmountByTaxTypes
203                     } else {
204                         System.debug(
205                             LoggingLevel.WARN,
206                             'TaxOverride consuming unrecognized property:
207                         );
208                         consumeObject(parser);
209                     }
210                 }
211             }
212         }
213     }
214 }
215
216 public class Parameters {
217     public String name { get; set; }
218     public String value { get; set; }
219     public String unit { get; set; }
220
221     public Parameters() {
222     }
223     public Parameters(JSONParser parser) {
224         while (parser.nextToken() != System.JSONToken.END_OBJECT
225             if (parser.getCurrentToken() == System.JSONToken.FIELD
226                 String text = parser.getText();
227                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
228                     if (text == 'name') {
229                         name = parser.getText();
230                     } else if (text == 'value') {
231                         value = parser.getText();
232                     } else if (text == 'unit') {
233                         unit = parser.getText();
234                     } else {
235                         System.debug(
236                             LoggingLevel.WARN,
237                             'Parameters consuming unrecognized property: '
238                         );
239                         consumeObject(parser);
240                     }
241                 }
242             }
243         }
244     }
245 }
246
247 public class ShipFrom {
248     public String id { get; set; }

```

```

249 public Integer transactionId { get; set; }
250 public String boundaryLevel { get; set; }
251 public String line1 { get; set; }
252 public String city { get; set; }
253 public String region { get; set; }
254 public String postalCode { get; set; }
255 public String country { get; set; }
256 public Integer taxRegionId { get; set; }
257
258 public ShipFrom() {
259 }
260 public ShipFrom(JSONParser parser) {
261     while (parser.nextToken() != System.JSONToken.END_OBJECT
262         if (parser.getCurrentToken() == System.JSONToken.FIELD
263             String text = parser.getText();
264             if (parser.nextToken() != System.JSONToken.VALUE_NULL
265                 if (text == 'id') {
266                     id = parser.getText();
267                 } else if (text == 'transactionId') {
268                     transactionId = parser.getIntegerValue();
269                 } else if (text == 'boundaryLevel') {
270                     boundaryLevel = parser.getText();
271                 } else if (text == 'line1') {
272                     line1 = parser.getText();
273                 } else if (text == 'city') {
274                     city = parser.getText();
275                 } else if (text == 'region') {
276                     region = parser.getText();
277                 } else if (text == 'postalCode') {
278                     postalCode = parser.getText();
279                 } else if (text == 'country') {
280                     country = parser.getText();
281                 } else if (text == 'taxRegionId') {
282                     taxRegionId = parser.getIntegerValue();
283                 } else {
284                     System.debug(
285                         LoggingLevel.WARN,
286                         'ShipFrom consuming unrecognized property: ' +
287                     );
288                     consumeObject(parser);
289                 }
290             }
291         }
292     }
293 }
294 }
295
296 public class UserDefinedFields {
297     public String name { get; set; }
298     public String value { get; set; }
299
300     public UserDefinedFields() {
301     }
302     public UserDefinedFields(JSONParser parser) {
303         while (parser.nextToken() != System.JSONToken.END_OBJECT
304             if (parser.getCurrentToken() == System.JSONToken.FIELD
305                 String text = parser.getText();
306                 if (parser.nextToken() != System.JSONToken.VALUE_NULL
307                     if (text == 'name') {
308                         name = parser.getText();

```

```

309         } else if (text == 'value') {
310             value = parser.getText();
311         } else {
312             System.debug(
313                 LoggingLevel.WARN,
314                 'UserDefinedFields consuming unrecognized prop
315             );
316             consumeObject(parser);
317         }
318     }
319 }
320 }
321 }
322 }
323
324 public class Lines {
325     public String number_Z { get; set; } // in json: number
326     public Double quantity { get; set; }
327     public Double amount { get; set; }
328     public String taxCode { get; set; }
329     public String customerUsageType { get; set; }
330     public String entityUseCode { get; set; }
331     public String itemCode { get; set; }
332     public String exemptionCode { get; set; }
333     public Boolean discounted { get; set; }
334     public Boolean taxIncluded { get; set; }
335     public String revenueAccount { get; set; }
336     public String ref1 { get; set; }
337     public String ref2 { get; set; }
338     public String description { get; set; }
339     public String businessIdentificationNo { get; set; }
340     public TaxOverride taxOverride { get; set; }
341     public List<Parameters> parameters { get; set; }
342     public List<UserDefinedFields> userDefinedFields { get; se
343     public String hsCode { get; set; }
344     public Integer merchantSellerId { get; set; }
345     public String merchantSellerIdentifier { get; set; }
346     public String marketplaceLiabilityType { get; set; }
347     public String originationDocumentId { get; set; }
348     public String originationSite { get; set; }
349     public String category { get; set; }
350     public String summary { get; set; }
351     public Addresses addresses { get; set; }
352
353     public Lines() {
354     }
355     public Lines(JSONParser parser) {
356         while (parser.nextToken() != System.JSONToken.END_OBJECT
357             if (parser.getCurrentToken() == System.JSONToken.FIELD
358                 String text = parser.getText();
359                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
360                     if (text == 'number') {
361                         number_Z = parser.getText();
362                     } else if (text == 'quantity') {
363                         quantity = parser.getDoubleValue();
364                     } else if (text == 'amount') {
365                         amount = parser.getDoubleValue();
366                     } else if (text == 'taxCode') {
367                         taxCode = parser.getText();
368                     } else if (text == 'customerUsageType') {

```

```

369         customerUsageType = parser.getText();
370     } else if (text == 'entityUseCode') {
371         entityUseCode = parser.getText();
372     } else if (text == 'itemCode') {
373         itemCode = parser.getText();
374     } else if (text == 'exemptionCode') {
375         exemptionCode = parser.getText();
376     } else if (text == 'discounted') {
377         discounted = parser.getBooleanValue();
378     } else if (text == 'taxIncluded') {
379         taxIncluded = parser.getBooleanValue();
380     } else if (text == 'revenueAccount') {
381         revenueAccount = parser.getText();
382     } else if (text == 'ref1') {
383         ref1 = parser.getText();
384     } else if (text == 'ref2') {
385         ref2 = parser.getText();
386     } else if (text == 'description') {
387         description = parser.getText();
388     } else if (text == 'businessIdentificationNo') {
389         businessIdentificationNo = parser.getText();
390     } else if (text == 'taxOverride') {
391         taxOverride = new TaxOverride(parser);
392     } else if (text == 'parameters') {
393         parameters = arrayOfParameters(parser);
394     } else if (text == 'userDefinedFields') {
395         userDefinedFields = arrayOfUserDefinedFields(par
396     } else if (text == 'hsCode') {
397         hsCode = parser.getText();
398     } else if (text == 'merchantSellerId') {
399         merchantSellerId = parser.getIntegerValue();
400     } else if (text == 'merchantSellerIdentifier') {
401         merchantSellerIdentifier = parser.getText();
402     } else if (text == 'marketplaceLiabilityType') {
403         marketplaceLiabilityType = parser.getText();
404     } else if (text == 'originationDocumentId') {
405         originationDocumentId = parser.getText();
406     } else if (text == 'originationSite') {
407         originationSite = parser.getText();
408     } else if (text == 'category') {
409         category = parser.getText();
410     } else if (text == 'summary') {
411         summary = parser.getText();
412     } else if (text == 'addresses') {
413         addresses = new Addresses(parser);
414     } else {
415         System.debug(
416             LoggingLevel.WARN,
417             'Lines consuming unrecognized property: ' + te
418         );
419         consumeObject(parser);
420     }
421 }
422 }
423 }
424 }
425 }
426
427 public static AvalaraTaxRequest parse(String json) {
428     System.JSONParser parser = System.JSON.createParser(json);

```



```

429     return new AvalaraTaxRequest(parser);
430 }
431
432 public static void consumeObject(System.JSONParser parser) {
433     Integer depth = 0;
434     do {
435         System.JSONToken curr = parser.getCurrentToken();
436         if (
437             curr == System.JSONToken.START_OBJECT ||
438             curr == System.JSONToken.START_ARRAY
439         ) {
440             depth++;
441         } else if (
442             curr == System.JSONToken.END_OBJECT ||
443             curr == System.JSONToken.END_ARRAY
444         ) {
445             depth--;
446         }
447     } while (depth > 0 && parser.nextToken() != null);
448 }
449
450 private static List<TaxAmountByTaxTypes> arrayOfTaxAmountByT
451     System.JSONParser p
452 ) {
453     List<TaxAmountByTaxTypes> res = new List<TaxAmountByTaxTyp
454     if (p.getCurrentToken() == null)
455         p.nextToken();
456     while (p.nextToken() != System.JSONToken.END_ARRAY) {
457         res.add(new TaxAmountByTaxTypes(p));
458     }
459     return res;
460 }
461
462 private static List<UserDefinedFields> arrayOfUserDefinedFie
463     System.JSONParser p
464 ) {
465     List<UserDefinedFields> res = new List<UserDefinedFields>(
466     if (p.getCurrentToken() == null)
467         p.nextToken();
468     while (p.nextToken() != System.JSONToken.END_ARRAY) {
469         res.add(new UserDefinedFields(p));
470     }
471     return res;
472 }
473
474 private static List<Parameters> arrayOfParameters(System.JSO
475     List<Parameters> res = new List<Parameters>();
476     if (p.getCurrentToken() == null)
477         p.nextToken();
478     while (p.nextToken() != System.JSONToken.END_ARRAY) {
479         res.add(new Parameters(p));
480     }
481     return res;
482 }
483
484 private static List<Lines> arrayOfLines(System.JSONParser p)
485     List<Lines> res = new List<Lines>();
486     if (p.getCurrentToken() == null)
487         p.nextToken();
488     while (p.nextToken() != System.JSONToken.END_ARRAY) {

```

		<pre> 489 res.add(new Lines(p)); 490 } 491 return res; 492 } 493 } 494 </pre>
AvalaraTaxResponse	This class is a DTO used to parse tax response returned by Avalara	<pre> 1 public class AvalaraTaxResponse { 2 public class Addresses { 3 public String id { get; set; } 4 public String transactionId { get; set; } 5 public String boundaryLevel { get; set; } 6 public String line1 { get; set; } 7 public String line2 { get; set; } 8 public String line3 { get; set; } 9 public String city { get; set; } 10 public String region { get; set; } 11 public String postalCode { get; set; } 12 public String country { get; set; } 13 public String taxRegionId { get; set; } 14 public String latitude { get; set; } 15 public String longitude { get; set; } 16 17 public Addresses() { 18 } 19 public Addresses(JSONParser parser) { 20 while (parser.nextToken() != System.JSONToken.END_OBJECT 21 if (parser.getCurrentToken() == System.JSONToken.FIELD 22 String text = parser.getText(); 23 if (parser.nextToken() != System.JSONToken.VALUE_NULL 24 if (text == 'id') { 25 id = parser.getText(); 26 } else if (text == 'transactionId') { 27 transactionId = parser.getText(); 28 } else if (text == 'boundaryLevel') { 29 boundaryLevel = parser.getText(); 30 } else if (text == 'line1') { 31 line1 = parser.getText(); 32 } else if (text == 'line2') { 33 line2 = parser.getText(); 34 } else if (text == 'line3') { 35 line3 = parser.getText(); 36 } else if (text == 'city') { 37 city = parser.getText(); 38 } else if (text == 'region') { 39 region = parser.getText(); 40 } else if (text == 'postalCode') { 41 postalCode = parser.getText(); 42 } else if (text == 'country') { 43 country = parser.getText(); 44 } else if (text == 'taxRegionId') { 45 taxRegionId = parser.getText(); 46 } else if (text == 'latitude') { 47 latitude = parser.getText(); 48 } else if (text == 'longitude') { 49 longitude = parser.getText(); 50 } else { 51 System.debug(52 LoggingLevel.WARN, 53 'Addresses consuming unrecognized property: ' </pre>

```

54         );
55         consumeObject(parser);
56     }
57 }
58 }
59 }
60 }
61 }
62
63 public class Details {
64     public String id { get; set; }
65     public String transactionLineId { get; set; }
66     public String transactionId { get; set; }
67     public String addressId { get; set; }
68     public String country { get; set; }
69     public String region { get; set; }
70     public String stateFIPS { get; set; }
71     public Decimal exemptAmount { get; set; }
72     public String exemptReasonId { get; set; }
73     public Boolean inState { get; set; }
74     public String jurisCode { get; set; }
75     public String jurisName { get; set; }
76     public String jurisdictionId { get; set; }
77     public String signatureCode { get; set; }
78     public String stateAssignedNo { get; set; }
79     public String jurisType { get; set; }
80     public Decimal nonTaxableAmount { get; set; }
81     public String nonTaxableRuleId { get; set; }
82     public String nonTaxableType { get; set; }
83     public Double rate { get; set; }
84     public String rateRuleId { get; set; }
85     public String rateSourceId { get; set; }
86     public String serCode { get; set; }
87     public String sourcing { get; set; }
88     public Double tax { get; set; }
89     public Decimal taxableAmount { get; set; }
90     public String taxType { get; set; }
91     public String taxName { get; set; }
92     public String taxAuthorityTypeId { get; set; }
93     public String taxRegionId { get; set; }
94     public Double taxCalculated { get; set; }
95     public String taxOverride { get; set; }
96     public String rateType { get; set; }
97     public Double taxableUnits { get; set; }
98     public Integer nonTaxableUnits { get; set; }
99     public Double exemptUnits { get; set; }
100    public Integer reportingTaxableUnits { get; set; }
101    public Integer reportingNonTaxableUnits { get; set; }
102    public Integer reportingExemptUnits { get; set; }
103    public Decimal reportingTax { get; set; }
104    public Decimal reportingTaxCalculated { get; set; }
105    public String jurisdictionType { get; set; }
106    public String taxSubTypeId { get; set; }
107    public String rateTypeCode { get; set; }
108    public String unitOfBasis { get; set; }
109    public Boolean isNonPassThru { get; set; }
110    public Boolean isFee { get; set; }
111    public String liabilityType { get; set; }
112    public String chargedTo { get; set; }
113

```

```
114     public Details() {
115     }
116     public Details(JSONParser parser) {
117         while (parser.nextToken() != System.JSONToken.END_OBJECT
118             if (parser.getCurrentToken() == System.JSONToken.FIELD
119                 String text = parser.getText();
120                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
121                     if (text == 'id') {
122                         id = parser.getText();
123                     } else if (text == 'transactionLineId') {
124                         transactionLineId = parser.getText();
125                     } else if (text == 'transactionId') {
126                         transactionId = parser.getText();
127                     } else if (text == 'addressId') {
128                         addressId = parser.getText();
129                     } else if (text == 'country') {
130                         country = parser.getText();
131                     } else if (text == 'region') {
132                         region = parser.getText();
133                     } else if (text == 'stateFIPS') {
134                         stateFIPS = parser.getText();
135                     } else if (text == 'exemptAmount') {
136                         exemptAmount = parser.getDecimalValue();
137                     } else if (text == 'exemptReasonId') {
138                         exemptReasonId = parser.getText();
139                     } else if (text == 'inState') {
140                         inState = parser.getBooleanValue();
141                     } else if (text == 'jurisCode') {
142                         jurisCode = parser.getText();
143                     } else if (text == 'jurisName') {
144                         jurisName = parser.getText();
145                     } else if (text == 'jurisdictionId') {
146                         jurisdictionId = parser.getText();
147                     } else if (text == 'signatureCode') {
148                         signatureCode = parser.getText();
149                     } else if (text == 'stateAssignedNo') {
150                         stateAssignedNo = parser.getText();
151                     } else if (text == 'jurisType') {
152                         jurisType = parser.getText();
153                     } else if (text == 'nonTaxableAmount') {
154                         nonTaxableAmount = parser.getDecimalValue();
155                     } else if (text == 'nonTaxableRuleId') {
156                         nonTaxableRuleId = parser.getText();
157                     } else if (text == 'nonTaxableType') {
158                         nonTaxableType = parser.getText();
159                     } else if (text == 'rate') {
160                         rate = parser.getDoubleValue();
161                     } else if (text == 'rateRuleId') {
162                         rateRuleId = parser.getText();
163                     } else if (text == 'rateSourceId') {
164                         rateSourceId = parser.getText();
165                     } else if (text == 'serCode') {
166                         serCode = parser.getText();
167                     } else if (text == 'sourcing') {
168                         sourcing = parser.getText();
169                     } else if (text == 'tax') {
170                         tax = parser.getDoubleValue();
171                     } else if (text == 'taxableAmount') {
172                         taxableAmount = parser.getDecimalValue();
173                     } else if (text == 'taxType') {
```

```

174         taxType = parser.getText();
175     } else if (text == 'taxName') {
176         taxName = parser.getText();
177     } else if (text == 'taxAuthorityTypeId') {
178         taxAuthorityTypeId = parser.getText();
179     } else if (text == 'taxRegionId') {
180         taxRegionId = parser.getText();
181     } else if (text == 'taxCalculated') {
182         taxCalculated = parser.getDoubleValue();
183     } else if (text == 'taxOverride') {
184         taxOverride = parser.getText();
185     } else if (text == 'rateType') {
186         rateType = parser.getText();
187     } else if (text == 'taxableUnits') {
188         taxableUnits = parser.getDoubleValue();
189     } else if (text == 'nonTaxableUnits') {
190         nonTaxableUnits = parser.getIntegerValue();
191     } else if (text == 'exemptUnits') {
192         exemptUnits = parser.getDoubleValue();
193     } else if (text == 'reportingTaxableUnits') {
194         reportingTaxableUnits = parser.getIntegerValue();
195     } else if (text == 'reportingNonTaxableUnits') {
196         reportingNonTaxableUnits = parser.getIntegerValue();
197     } else if (text == 'reportingExemptUnits') {
198         reportingExemptUnits = parser.getIntegerValue();
199     } else if (text == 'reportingTax') {
200         reportingTax = parser.getDecimalValue();
201     } else if (text == 'reportingTaxCalculated') {
202         reportingTaxCalculated = parser.getDecimalValue();
203     } else if (text == 'jurisdictionType') {
204         jurisdictionType = parser.getText();
205     } else if (text == 'taxSubTypeId') {
206         taxSubTypeId = parser.getText();
207     } else if (text == 'rateTypeCode') {
208         rateTypeCode = parser.getText();
209     } else if (text == 'unitOfBasis') {
210         unitOfBasis = parser.getText();
211     } else if (text == 'isNonPassThru') {
212         isNonPassThru = parser.getBooleanValue();
213     } else if (text == 'isFee') {
214         isFee = parser.getBooleanValue();
215     } else if (text == 'liabilityType') {
216         liabilityType = parser.getText();
217     } else if (text == 'chargedTo') {
218         chargedTo = parser.getText();
219     } else {
220         System.debug(
221             LoggingLevel.WARN,
222             'Details consuming unrecognized property: ' +
223             text);
224         consumeObject(parser);
225     }
226 }
227 }
228 }
229 }
230 }
231
232 public String id { get; set; }
233 public String code { get; set; }

```

```

234 public String companyId { get; set; }
235 public String date_Z { get; set; } // in json: date
236 public String paymentDate { get; set; }
237 public String status { get; set; }
238 public String type { get; set; }
239 public String batchCode { get; set; }
240 public String currencyCode { get; set; }
241 public String exchangeRateCurrencyCode { get; set; }
242 public String customerUsageType { get; set; }
243 public String entityUseCode { get; set; }
244 public String customerVendorCode { get; set; }
245 public String customerCode { get; set; }
246 public String exemptNo { get; set; }
247 public Boolean reconciled { get; set; }
248 public String locationCode { get; set; }
249 public String reportingLocationCode { get; set; }
250 public String purchaseOrderNo { get; set; }
251 public String referenceCode { get; set; }
252 public String salespersonCode { get; set; }
253 public Double totalAmount { get; set; }
254 public Double totalExempt { get; set; }
255 public Double totalDiscount { get; set; }
256 public Double totalTax { get; set; }
257 public Double totalTaxable { get; set; }
258 public Double totalTaxCalculated { get; set; }
259 public String adjustmentReason { get; set; }
260 public Boolean locked { get; set; }
261 public String version { get; set; }
262 public String exchangeRateEffectiveDate { get; set; }
263 public Double exchangeRate { get; set; }
264 public String description { get; set; }
265 public String email { get; set; }
266 public String modifiedDate { get; set; }
267 public String modifiedUserId { get; set; }
268 public String taxDate { get; set; }
269 public List<Lines> lines { get; set; }
270 public List<Addresses> addresses { get; set; }
271 public List<Summary> summary { get; set; }
272 public List<TaxDetailsByTaxType> taxDetailsByTaxType { get;
273
274 public AvalaraTaxResponse() {
275 }
276 public AvalaraTaxResponse(JSONParser parser) {
277     while (parser.nextToken() != System.JSONToken.END_OBJECT)
278         if (parser.getCurrentToken() == System.JSONToken.FIELD_N
279             String text = parser.getText();
280             if (parser.nextToken() != System.JSONToken.VALUE_NULL)
281                 if (text == 'id') {
282                     id = parser.getText();
283                 } else if (text == 'code') {
284                     code = parser.getText();
285                 } else if (text == 'companyId') {
286                     companyId = parser.getText();
287                 } else if (text == 'date') {
288                     date_Z = parser.getText();
289                 } else if (text == 'paymentDate') {
290                     paymentDate = parser.getText();
291                 } else if (text == 'status') {
292                     status = parser.getText();
293                 } else if (text == 'type') {

```

```
294         type = parser.getText();
295     } else if (text == 'batchCode') {
296         batchCode = parser.getText();
297     } else if (text == 'currencyCode') {
298         currencyCode = parser.getText();
299     } else if (text == 'exchangeRateCurrencyCode') {
300         exchangeRateCurrencyCode = parser.getText();
301     } else if (text == 'customerUsageType') {
302         customerUsageType = parser.getText();
303     } else if (text == 'entityUseCode') {
304         entityUseCode = parser.getText();
305     } else if (text == 'customerVendorCode') {
306         customerVendorCode = parser.getText();
307     } else if (text == 'customerCode') {
308         customerCode = parser.getText();
309     } else if (text == 'exemptNo') {
310         exemptNo = parser.getText();
311     } else if (text == 'reconciled') {
312         reconciled = parser.getBooleanValue();
313     } else if (text == 'locationCode') {
314         locationCode = parser.getText();
315     } else if (text == 'reportingLocationCode') {
316         reportingLocationCode = parser.getText();
317     } else if (text == 'purchaseOrderNo') {
318         purchaseOrderNo = parser.getText();
319     } else if (text == 'referenceCode') {
320         referenceCode = parser.getText();
321     } else if (text == 'salespersonCode') {
322         salespersonCode = parser.getText();
323     } else if (text == 'totalAmount') {
324         totalAmount = parser.getDoubleValue();
325     } else if (text == 'totalExempt') {
326         totalExempt = parser.getDoubleValue();
327     } else if (text == 'totalDiscount') {
328         totalDiscount = parser.getDoubleValue();
329     } else if (text == 'totalTax') {
330         totalTax = parser.getDoubleValue();
331     } else if (text == 'totalTaxable') {
332         totalTaxable = parser.getDoubleValue();
333     } else if (text == 'totalTaxCalculated') {
334         totalTaxCalculated = parser.getDoubleValue();
335     } else if (text == 'adjustmentReason') {
336         adjustmentReason = parser.getText();
337     } else if (text == 'locked') {
338         locked = parser.getBooleanValue();
339     } else if (text == 'version') {
340         version = parser.getText();
341     } else if (text == 'exchangeRateEffectiveDate') {
342         exchangeRateEffectiveDate = parser.getText();
343     } else if (text == 'exchangeRate') {
344         exchangeRate = parser.getDoubleValue();
345     } else if (text == 'description') {
346         description = parser.getText();
347     } else if (text == 'email') {
348         email = parser.getText();
349     } else if (text == 'modifiedDate') {
350         modifiedDate = parser.getText();
351     } else if (text == 'modifiedUserId') {
352         modifiedUserId = parser.getText();
353     } else if (text == 'taxDate') {
```

```

354         taxDate = parser.getText();
355     } else if (text == 'lines') {
356         lines = arrayOfLines(parser);
357     } else if (text == 'addresses') {
358         addresses = arrayOfAddresses(parser);
359     } else if (text == 'summary') {
360         summary = arrayOfSummary(parser);
361     } else if (text == 'taxDetailsByTaxType') {
362         taxDetailsByTaxType = arrayOfTaxDetailsByTaxType(p
363     } else {
364         System.debug(
365             LoggingLevel.WARN,
366             'AvalaraTaxResponse consuming unrecognized prop
367         );
368         consumeObject(parser);
369     }
370 }
371 }
372 }
373 }
374
375 public class NonPassthroughDetails {
376     public NonPassthroughDetails() {
377     }
378     public NonPassthroughDetails(JSONParser parser) {
379         while (parser.nextToken() != System.JSONToken.END_OBJECT
380             if (parser.getCurrentToken() == System.JSONToken.FIELD
381                 String text = parser.getText();
382                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
383                     {
384                         System.debug(
385                             LoggingLevel.WARN,
386                             'NonPassthroughDetails consuming unrecognized
387                         );
388                         consumeObject(parser);
389                     }
390                 }
391             }
392         }
393     }
394 }
395
396 public class TaxDetailsByTaxType {
397     public String taxType { get; set; }
398     public Decimal totalTaxable { get; set; }
399     public Double totalExempt { get; set; }
400     public Decimal totalNonTaxable { get; set; }
401     public Double totalTax { get; set; }
402
403     public TaxDetailsByTaxType() {
404     }
405     public TaxDetailsByTaxType(JSONParser parser) {
406         while (parser.nextToken() != System.JSONToken.END_OBJECT
407             if (parser.getCurrentToken() == System.JSONToken.FIELD
408                 String text = parser.getText();
409                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
410                     if (text == 'taxType') {
411                         taxType = parser.getText();
412                     } else if (text == 'totalTaxable') {
413                         totalTaxable = parser.getDecimalValue();

```



```

414         } else if (text == 'totalExempt') {
415             totalExempt = parser.getDoubleValue();
416         } else if (text == 'totalNonTaxable') {
417             totalNonTaxable = parser.getDecimalValue();
418         } else if (text == 'totalTax') {
419             totalTax = parser.getDoubleValue();
420         } else {
421             System.debug(
422                 LoggingLevel.WARN,
423                 'TaxDetailsByTaxType consuming unrecognized pr
424             );
425             consumeObject(parser);
426         }
427     }
428 }
429 }
430 }
431 }
432
433 public class Summary {
434     public String country { get; set; }
435     public String region { get; set; }
436     public String jurisType { get; set; }
437     public String jurisCode { get; set; }
438     public String jurisName { get; set; }
439     public String taxAuthorityType { get; set; }
440     public String stateAssignedNo { get; set; }
441     public String taxType { get; set; }
442     public String taxSubType { get; set; }
443     public String taxName { get; set; }
444     public String rateType { get; set; }
445     public Double taxable { get; set; }
446     public Double rate { get; set; }
447     public Double tax { get; set; }
448     public Double taxCalculated { get; set; }
449     public Double nonTaxable { get; set; }
450     public Double exemption { get; set; }
451
452     public Summary() {
453     }
454
455     public Summary(JSONParser parser) {
456         while (parser.nextToken() != System.JSONToken.END_OBJECT
457             if (parser.getCurrentToken() == System.JSONToken.FIELD
458                 String text = parser.getText();
459                 if (parser.nextToken() != System.JSONToken.VALUE_NUL
460                     if (text == 'country') {
461                         country = parser.getText();
462                     } else if (text == 'region') {
463                         region = parser.getText();
464                     } else if (text == 'jurisType') {
465                         jurisType = parser.getText();
466                     } else if (text == 'jurisCode') {
467                         jurisCode = parser.getText();
468                     } else if (text == 'jurisName') {
469                         jurisName = parser.getText();
470                     } else if (text == 'taxAuthorityType') {
471                         taxAuthorityType = parser.getText();
472                     } else if (text == 'stateAssignedNo') {
473                         stateAssignedNo = parser.getText();
474                     } else if (text == 'taxType') {

```

```

474         taxType = parser.getText();
475     } else if (text == 'taxSubType') {
476         taxSubType = parser.getText();
477     } else if (text == 'taxName') {
478         taxName = parser.getText();
479     } else if (text == 'rateType') {
480         rateType = parser.getText();
481     } else if (text == 'taxable') {
482         taxable = parser.getDoubleValue();
483     } else if (text == 'rate') {
484         rate = parser.getDoubleValue();
485     } else if (text == 'tax') {
486         tax = parser.getDoubleValue();
487     } else if (text == 'taxCalculated') {
488         taxCalculated = parser.getDoubleValue();
489     } else if (text == 'nonTaxable') {
490         nonTaxable = parser.getDoubleValue();
491     } else if (text == 'exemption') {
492         exemption = parser.getDoubleValue();
493     } else {
494         System.debug(
495             LoggingLevel.WARN,
496             'Summary consuming unrecognized property: ' +
497             );
498         consumeObject(parser);
499     }
500 }
501 }
502 }
503 }
504 }
505
506 public class Lines {
507     public String id { get; set; }
508     public String transactionId { get; set; }
509     public String lineNumber { get; set; }
510     public String customerUsageType { get; set; }
511     public String entityUseCode { get; set; }
512     public String description { get; set; }
513     public Double discountAmount { get; set; }
514     public Double exemptAmount { get; set; }
515     public String exemptCertId { get; set; }
516     public String exemptNo { get; set; }
517     public Boolean isItemTaxable { get; set; }
518     public String itemCode { get; set; }
519     public Double lineAmount { get; set; }
520     public Double quantity { get; set; }
521     public String ref1 { get; set; }
522     public String ref2 { get; set; }
523     public String reportingDate { get; set; }
524     public Double tax { get; set; }
525     public Double taxableAmount { get; set; }
526     public Double taxCalculated { get; set; }
527     public String taxCode { get; set; }
528     public String taxCodeId { get; set; }
529     public String taxDate { get; set; }
530     public Boolean taxIncluded { get; set; }
531     public List<Details> details { get; set; }
532     public List<NonPassthroughDetails> nonPassthroughDetails {
533         public String hsCode { get; set; }

```

```

532 public Double costInsuranceFreight { get; set; }
535 public String vatCode { get; set; }
536 public String vatNumberTypeId { get; set; }
537 public String originAddressId { get; set; }
538 public String destinationAddressId { get; set; }
539
540 public Lines() {
541 }
542 public Lines(JSONParser parser) {
543     while (parser.nextToken() != System.JSONToken.END_OBJECT)
544         if (parser.getCurrentToken() == System.JSONToken.FIELD_NAME)
545             String text = parser.getText();
546             if (parser.nextToken() != System.JSONToken.VALUE_NULL)
547                 if (text == 'id') {
548                     id = parser.getText();
549                 } else if (text == 'transactionId') {
550                     transactionId = parser.getText();
551                 } else if (text == 'lineNumber') {
552                     lineNumber = parser.getText();
553                 } else if (text == 'customerUsageType') {
554                     customerUsageType = parser.getText();
555                 } else if (text == 'entityUseCode') {
556                     entityUseCode = parser.getText();
557                 } else if (text == 'description') {
558                     description = parser.getText();
559                 } else if (text == 'discountAmount') {
560                     discountAmount = parser.getDoubleValue();
561                 } else if (text == 'exemptAmount') {
562                     exemptAmount = parser.getDoubleValue();
563                 } else if (text == 'exemptCertId') {
564                     exemptCertId = parser.getText();
565                 } else if (text == 'exemptNo') {
566                     exemptNo = parser.getText();
567                 } else if (text == 'isItemTaxable') {
568                     isItemTaxable = parser.getBooleanValue();
569                 } else if (text == 'itemCode') {
570                     itemCode = parser.getText();
571                 } else if (text == 'lineAmount') {
572                     lineAmount = parser.getDoubleValue();
573                 } else if (text == 'quantity') {
574                     quantity = parser.getDoubleValue();
575                 } else if (text == 'ref1') {
576                     ref1 = parser.getText();
577                 } else if (text == 'ref2') {
578                     ref2 = parser.getText();
579                 } else if (text == 'reportingDate') {
580                     reportingDate = parser.getText();
581                 } else if (text == 'tax') {
582                     tax = parser.getDoubleValue();
583                 } else if (text == 'taxableAmount') {
584                     taxableAmount = parser.getDoubleValue();
585                 } else if (text == 'taxCalculated') {
586                     taxCalculated = parser.getDoubleValue();
587                 } else if (text == 'taxCode') {
588                     taxCode = parser.getText();
589                 } else if (text == 'taxCodeId') {
590                     taxCodeId = parser.getText();
591                 } else if (text == 'taxDate') {
592                     taxDate = parser.getText();
593                 } else if (text == 'taxIncluded') {

```

```






















594         taxIncluded = parser.getBooleanValue();
595     } else if (text == 'details') {
596         details = arrayOfDetails(parser);
597     } else if (text == 'nonPassthroughDetails') {
598         nonPassthroughDetails = arrayOfNonPassthroughDet
599     } else if (text == 'hsCode') {
600         hsCode = parser.getText();
601     } else if (text == 'costInsuranceFreight') {
602         costInsuranceFreight = parser.getDoubleValue();
603     } else if (text == 'vatCode') {
604         vatCode = parser.getText();
605     } else if (text == 'vatNumberTypeId') {
606         vatNumberTypeId = parser.getText();
607     } else if (text == 'originAddressId') {
608         originAddressId = parser.getText();
609     } else if (text == 'destinationAddressId') {
610         destinationAddressId = parser.getText();
611     } else {
612         System.debug(
613             LoggingLevel.WARN,
614             'Lines consuming unrecognized property: ' + te
615         );
616         consumeObject(parser);
617     }
618 }
619 }
620 }
621 }
622 }
623
624 public static AvalaraTaxResponse parse(String json) {
625     System.JSONParser parser = System.JSON.createParser(json);
626     return new AvalaraTaxResponse(parser);
627 }
628
629 public static void consumeObject(System.JSONParser parser) {
630     Integer depth = 0;
631     do {
632         System.JSONToken curr = parser.getCurrentToken();
633         if (
634             curr == System.JSONToken.START_OBJECT ||
635             curr == System.JSONToken.START_ARRAY
636         ) {
637             depth++;
638         } else if (
639             curr == System.JSONToken.END_OBJECT ||
640             curr == System.JSONToken.END_ARRAY
641         ) {
642             depth--;
643         }
644     } while (depth > 0 && parser.nextToken() != null);
645 }
646
647 private static List<Addresses> arrayOfAddresses(System.JSONP
648     List<Addresses> res = new List<Addresses>();
649     if (p.getCurrentToken() == null)
650         p.nextToken();
651     while (p.nextToken() != System.JSONToken.END_ARRAY) {
652         res.add(new Addresses(p));
653     }

```

```

654     return res;
655 }
656
657 private static List<Details> arrayOfDetails(System.JSONParse
658     List<Details> res = new List<Details>();
659     if (p.getCurrentToken() == null)
660         p.nextToken();
661     while (p.nextToken() != System.JSONToken.END_ARRAY) {
662         res.add(new Details(p));
663     }
664     return res;
665 }
666
667 private static List<Lines> arrayOfLines(System.JSONParser p)
668     List<Lines> res = new List<Lines>();
669     if (p.getCurrentToken() == null)
670         p.nextToken();
671     while (p.nextToken() != System.JSONToken.END_ARRAY) {
672         res.add(new Lines(p));
673     }
674     return res;
675 }
676
677 private static List<Summary> arrayOfSummary(System.JSONParse
678     List<Summary> res = new List<Summary>();
679     if (p.getCurrentToken() == null)
680         p.nextToken();
681     while (p.nextToken() != System.JSONToken.END_ARRAY) {
682         res.add(new Summary(p));
683     }
684     return res;
685 }
686
687 private static List<TaxDetailsByTaxType> arrayOfTaxDetailsBy
688     System.JSONParser p
689 ) {
690     List<TaxDetailsByTaxType> res = new List<TaxDetailsByTaxTy
691     if (p.getCurrentToken() == null)
692         p.nextToken();
693     while (p.nextToken() != System.JSONToken.END_ARRAY) {
694         res.add(new TaxDetailsByTaxType(p));
695     }
696     return res;
697 }
698
699 private static List<NonPassthroughDetails> arrayOfNonPassthr
700     System.JSONParser p
701 ) {
702     List<NonPassthroughDetails> res = new List<NonPassthroughD
703     if (p.getCurrentToken() == null)
704         p.nextToken();
705     while (p.nextToken() != System.JSONToken.END_ARRAY) {
706         res.add(new NonPassthroughDetails(p));
707     }
708     return res;
709 }
710 }
711
712

```

File	Modified
›  image-20240411-170839.png	yesterday at 10:29 PM by Manoj Tyagi
›  avalara_tax_calc_seq.drawio.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240404-231938.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240404-231809.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240404-230012.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240306-193010.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240306-193053.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240306-193410.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240306-193718.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240306-193853.png	yesterday at 10:29 PM by Manoj Tyagi
›  image-20240306-194511.png	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240306-194640.png	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240306-194749.png	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240228-041911.png	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240228-045132.png	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240228-041821.png	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240228-041538.png	yesterday at 10:30 PM by Manoj Tyagi
›  shipping_rates_sequence.drawio.png	yesterday at 10:30 PM by Manoj Tyagi
›  ups_shipping_rates_sequence.drawio.png	yesterday at 10:30 PM by Manoj Tyagi
›  ups_shipping.zip	yesterday at 10:30 PM by Manoj Tyagi
›  image-20240319-162328.png	yesterday at 10:30 PM by Manoj Tyagi

⬇ Drag and drop to upload or [browse for files](#) 🌀

⬇ [Download All](#)