LineItemResponse Class

Response class that stores details of a list of one or more line items on which the tax engine has calculated tax.

Namespace

CommerceTax

Example

This example uses a `LineItemResponse` list to store information about each line item that was processed as part of the request. For simplicity, the sample code uses a static value of 1 for the tax rate. However, most integrations typically have a more complex process for determining a tax rate. Most integrations also build a `TaxDetailsResponse` list to store the actual tax value information that they assign to each line item in the `LineItemResponse` list.

```
Double totalTax = 0.0;
            Double totalAmount = 0.0;
            List<commercetax.LineItemResponse> lineItemResponses = new List<commercetax.L
            for(Commercetax.TaxLineItemRequest lineItem : request.lineItems){
                commercetax.AddressesResponse addressesRes = new commercetax.AddressesRes
                if(request.DocumentCode == 'SetsNullForResponseWithoutException'){
                    addressesRes.setShipFrom(null);
                    addressesRes.setShipTO(null);
                    addressesRes.setSoldTo(null);
                }else{
                    commercetax.AddressResponse addRes = new commercetax.AddressResponse(
                    addRes.setLocationCode('locationCode');
                    addressesRes.setShipFrom(addRes);
                    addressesRes.setShipTO(addRes);
                    addressesRes.setSoldTo(addRes);
                }
                commercetax.LineItemResponse lineItemResponse = new commercetax.LineItemR
                Double totalLineTax = 0;
                List<commercetax.TaxDetailsResponse> taxDetailsResponses = new List<comme
                for(integer i =0;i<1;i++){
                    Integer rate  = 1;
                    Double taxableAmount = lineItem.amount;
                    commercetax.TaxDetailsResponse taxDetailsResponse = new commercetax.T
                    taxDetailsResponse.setRate(Double.valueOf(rate));
                    taxDetailsResponse.setTaxableAmount(taxableAmount);
                    Double tax = taxableAmount*rate;
                    totalLineTax+=tax;
                    taxDetailsResponse.setTax(taxableAmount*rate);
                    taxDetailsResponse.setExemptAmount(0);
                    taxDetailsResponse.setExemptReason('exemptReason');
                    taxDetailsResponse.setTaxRegionId('taxRegionId');
                    taxDetailsResponse.setTaxId(String.valueOf(getRandomInteger(0,2323233
                    taxDetailsResponse.setSerCode('serCode');
                    taxDetailsResponse.setTaxAuthorityTypeId('taxAuthorityTypeId');
                    if(request.DocumentCode == 'SetsNullForResponseWithoutException'){
                        taxDetailsResponse.setImposition(null);
                    }else{
                        commercetax.ImpositionResponse imposition = new commercetax.Impos
                        imposition.setSubType('subtype');
                        imposition.setType('type');
                        taxDetailsResponse.setImposition(imposition);
```

```
                    jurisdiction.setCountry('country');
                    jurisdiction.setRegion('region');
                    jurisdiction.setName('name');
                    jurisdiction.setStateAssignedNumber('stateAssignedNo');
                    jurisdiction.setId('id');
                    jurisdiction.setLevel('level');
                    taxDetailsResponse.setJurisdiction(jurisdiction);
                }

                taxDetailsResponses.add(taxDetailsResponse);
            }
            lineItemResponse.setTaxes(taxDetailsResponses);
            totalTax +=totalLineTax;
            totalAmount+=lineItem.amount;
```

- **LineItemResponse Methods**
  Learn more about the available methods with the `LineItemResponse` class.

LineItemResponse Methods

Learn more about the available methods with the `LineItemResponse` class.

The `LineItemResponse` class includes these methods.

- **setAddresses(addresses)**
  Sets the Addresses field on the `LineItemResponse` using an instance of `AddressesResponse` class.
- **setAmountDetails(amountDetails)**
  Sets the Amount Details field on the `LineItemResponse` using an instance of `AmountDetails`.
- **setEffectiveDate(effectiveDate)**
  Sets the EffectiveDate field on the `LineItemResponse` class. Effective Date fields are optional fields that store the date that a transaction takes effect. We provide these fields only for recordkeeping purposes – for example, if you must report an effective date to an external general ledger system. Salesforce doesn't use them to calculate any tax or payment values.
- **setIsTaxable(isTaxable)**
  Sets the IsTaxable field on the `LineItemResponse` class.
- **setLineNumber(lineNumber)**
  Sets the LineNumber field on the `LineItemResponse` class.
- **setProductCode(productCode)**
  Sets the ProductCode field on the `LineItemResponse` class.
- **setQuantity(quantity)**
  Sets the Quantity field on the `LineItemResponse` class.
- **setTaxCode(taxCode)**
  Sets the TaxCode field on the `LineItemResponse`.
- **setTaxes(taxes)**
  Sets the Taxes field on a `LineItemResponse`.

setAddresses(addresses)

Sets the Addresses field on the `LineItemResponse` using an instance of `AddressesResponse` class.

Signature

```
global void setAddresses(commercetax.AddressesResponse addresses)
```

Parameters

addresses

Type: void

setAmountDetails(amountDetails)

Sets the Amount Details field on the `LineItemResponse` using an instance of `AmountDetails`.

Signature

```
global void setAmountDetails(commercetax.AmountDetailsResponse amountDetails)
```

Parameters

amountDetails

Type: AmountDetailsResponse

Class that contains methods to set the tax amount, total amount with tax, total amount, and exempt amount.

Return Value

Type: void

setEffectiveDate(effectiveDate)

Sets the EffectiveDate field on the `LineItemResponse` class. Effective Date fields are optional fields that store the date that a transaction takes effect. We provide these fields only for recordkeeping purposes – for example, if you must report an effective date to an external general ledger system. Salesforce doesn't use them to calculate any tax or payment values.

Signature

```
global void setEffectiveDate(Datetime effectiveDate)
```

Parameters

effectiveDate

Type: Datetime

Optional field that stores the date that a transaction takes effect.

Return Value

Type: void

setIsTaxable(isTaxable)

Sets the IsTaxable field on the `LineItemResponse` class.

Signature

```
global void setIsTaxable(Boolean isTaxable)
```

Parameters

isTaxable

Type: Boolean

Whether line items were taxed as part of the tax calculation request.

Return Value

Type: void

```
global void setLineNumber(String lineNumber)
```

Parameters

lineNumber

Type: String

User-defined number used to identify a line item.

Return Value

Type: void

setProductCode(productCode)

Sets the ProductCode field on the `LineItemResponse` class.

Signature

```
global void setProductCode(String productCode)
```

Parameters

productCode

Type: String

Code for the product that a line item represents.

Return Value

Type: void

setQuantity(quantity)

Sets the Quantity field on the `LineItemResponse` class.

Signature

```
global void setQuantity(Double quantity)
```

Parameters

quantity

Type: Double

Quantity of a line item.

Return Value

Type: void

setTaxCode(taxCode)

Sets the TaxCode field on the `LineItemResponse`.

Signature

```
global void setTaxCode(String taxCode)
```

Parameters

taxCode

Return Value

Type: void

setTaxes(taxes)

Sets the Taxes field on a `LineItemResponse`.

Signature

```
global void setTaxes(List<commercetax.TaxDetailsResponse> taxes)
```

Parameters

taxes

Type: List<TaxDetailsResponse>

Tax values applied to a line item in the `LineItemResponse` list. This information is stored in a list of `TaxDetailsResponses`, which contains values such as tax, taxable amount, and tax rate.

Return Value

Type: void