TaxEngineAdapter Interface

Retrieves information from the tax engine and evaluates the information to define tax details.

Namespace

CommerceTax

- **TaxEngineAdapter Methods**
  Learn more about the available methods with the `TaxEngineAdapter` class.
- **TaxEngineAdapter Example Implementation**
  Refer to the example implementation of the `TaxEngineAdapter` interface to accept information from a tax engine and evaluate the information to define tax details.

TaxEngineAdapter Methods

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

The `TaxEngineAdapter` class includes these methods.

- **processRequest(requestType)**
  The `processRequest` method takes an instance of `TaxEngineContext` class and returns a response with the calculated tax details through the `TaxDetailsResponse` class or an error response through the `ErrorResponse` class.

processRequest(requestType)

The `processRequest` method takes an instance of `TaxEngineContext` class and returns a response with the calculated tax details through the `TaxDetailsResponse` class or an error response through the `ErrorResponse` class.

Signature

```
global commercetax.TaxEngineResponse processRequest(commercetax.TaxEngineContext var1)
```

Parameters

var1

Type: TaxEngineContext

Wrapper class that stores information about the type of a tax calculation request.

Return Value

Type: TaxEngineResponse

Generic interface representing a response from a tax engine.

TaxEngineAdapter Example Implementation

Refer to the example implementation of the `TaxEngineAdapter` interface to accept information from a tax engine and evaluate the information to define tax details.

Usage

The `TaxEngineAdapter` interface accepts information from the tax engine through the `TaxEngineContext` class. The interface evaluates the information to define tax in the response with details, such as tax amount and addresses. The response is used to update and create entities in the Salesforce org.

Use these steps to build a sample tax adapter implementation. Each tax adapter implementation varies based on your implementation requirements. Customize this example to suit your business requirements.

Example

- The custom adapter class implements the `TaxEngineAdapter` interface. The `processRequest` method takes an instance of `TaxEngineContext` class and returns a response with the calculated tax details through the `TaxDetailsResponse` class or an error response through the `ErrorResponse` class.

```
global virtual class AvalaraAdapter implements commercetax.TaxEngineAdapter {
    global commercetax.TaxEngineResponse processRequest(commercetax.TaxEngineContex
        commercetax.RequestType requestType = taxEngineContext.getRequestType();
        if(requestType == commercetax.RequestType.CalculateTax){
            return CalculateTaxService.getTax(taxEngineContext);
        }
        else
            return null;
    }
}
```

- This example shows the `CalculateTaxService` class.

```
global class CalculateTaxService {
    // ===========================================================================
    // CONSTANT
    // ===========================================================================
    private static final String AVALARA_ENDPOINT_URL_SANDBOX = 'https://sandbox-res
    // Avalara Endpoint URL Production
    private static final String AVALARA_ENDPOINT_URL_PRODUCTION = 'https://rest.ava
    private static final String TEST_REQUEST_BODY = '{  "id": -1,  "code": "0000013

    private static String getTestResponseString(){

     List<String> jsonResponse = new List<String> {
                                    '"id": 0',
                                    '"code": "testDocCode1231245984"',
                                    '"companyId": 468039',
                                    '"date": "2020-07-15"',
                                    '"paymentDate": "2020-07-15"',
                                    '"status": "Temporary"',
                                    '"type": "SalesOrder"',
                                    '"customerVendorCode": "testDocCode1234"',
                                    '"customerCode": "testDocCode1234"',
                                    '"reconciled": false',
                                    '"totalAmount": 232',
                                    '"totalExempt": 0',
                                    '"totalDiscount": 0',
                                    '"totalTax": 23.43',
                                    '"totalTaxable": 232',
                                    '"totalTaxCalculated": 23.43',
                                    '"adjustmentReason": "NotAdjusted"',
                                    '"locked": false',
                                    '"version": 1',
                                    '"exchangeRateEffectiveDate": "2020-07-15"',
                                    '"summary": [{"country": "US","region": "WA","ju
                                };
            return '{' + String.join(jsonResponse, ',') + '}';
        }

    public static commercetax.TaxEngineResponse getTax(commercetax.TaxEngineContext
    {
        commercetax.CalculateTaxRequest request = (commercetax.CalculateTaxRequest)
        commercetax.calculatetaxtype requestType = request.taxtype;
        string referenceEntity = request.ReferenceEntityId;
        try{
            List<commercetax.TaxLineItemRequest> listOfLines = request.lineItems;
            if(!listOfLines.isEmpty()){
                HttpService sendHttpRequest = new HttpService();
                sendHttpRequest.addHeader('Content-type', 'application/json');
                String requestBody = AvalaraJSONBuilder.getInstance().frameJsonForGe
                sendHttpRequest.post('/transactions/create',requestBody);
                //system.debug('Request '+requestBody);
                String responseString = '';
                if(Test.isRunningTest()){
                    responseString = getTestResponseString();
                } else{
                    responseString = sendHttpRequest.getResponse().getBody();
                }
                //system.debug(sendHttpRequest.getResponse());
                //system.debug('response'+responseString);
                //responseString = TEST_REQUEST_BODY;
                system.debug('Heap size used ' +Limits.getHeapSize());

                if(!responseString.contains('error'))
                {
                    commercetax.CalculateTaxResponse response = new commercetax.Calc
                    JsonSuccessParser jsonSuccessParserClass = JsonSuccessParser.par
                    response.setTaxTransactionType(request.taxTransactionType);
                    response.setDocumentCode(jsonSuccessParserClass.code);
                    response.setReferenceDocumentCode(jsonSuccessParserClass.referen
                    if(jsonSuccessParserClass.status == 'Temporary')  {
                        response.setStatus(commercetax.TaxTransactionStatus.Uncommit
                    }
                    if(jsonSuccessParserClass.status == 'Committed') {
                        response.setStatus(commercetax.TaxTransactionStatus.Committe
                    }
                    response.setTaxType(requestType);
                    commercetax.AmountDetailsResponse headerAmountResponse = new com
                    headerAmountResponse.setTotalAmountWithTax(jsonSuccessParserCla
                    headerAmountResponse.setExemptAmount(jsonSuccessParserClass.tota
                    headerAmountResponse.setTotalAmount(jsonSuccessParserClass.total
                    headerAmountResponse.setTaxAmount(jsonSuccessParserClass.totalTa
                    response.setAmountDetails(headerAmountResponse);
                    response.setStatusDescription(jsonSuccessParserClass.adjustmentR
                    response.setEffectiveDate(date.valueof(jsonSuccessParserClass.ta
                    response.setTransactionDate(date.valueof(jsonSuccessParserClass.
                    response.setReferenceEntityId(referenceEntity);
                    response.setTaxTransactionId(jsonSuccessParserClass.id);
                    response.setCurrencyIsoCode(request.currencyIsoCode);
                    List<commercetax.LineItemResponse> lineItemResponses = new List<
                    for(JsonSuccessParser.Lines linesToProcess: jsonSuccessParserCla
                    {
                        commercetax.LineItemResponse lineItemResponse = new commerce
                        Double rateCalculated = 0.0;
                        List<commercetax.TaxDetailsResponse> taxDetailsResponses = n
                        for(JsonSuccessParser.details linesDetails : linesToProcess.
                        {
                            commercetax.TaxDetailsResponse taxDetailsResponse = new
                            if(linesDetails.exemptAmount != 0){
                                taxDetailsResponse.setExemptAmount(linesDetails.exem
                                taxDetailsResponse.setExemptReason('Some reason we c
                            }
                                commercetax.ImpositionResponse imposition = new comm
                                    imposition.setSubType(linesDetails.taxName);
                                    imposition.setType(linesDetails.ratetype);
                                    imposition.setSubType(linesDetails.taxName);
```

```
                  jurisdiction.setId(linesDetails.jurisCode);
                  jurisdiction.setLevel(linesDetails.jurisType);
                  taxDetailsResponse.setJurisdiction(jurisdiction
                  rateCalculated += linesDetails.rate;
                taxDetailsResponse.setRate(rateCalculated);
                taxDetailsResponse.setTax(linesDetails.taxCalculate
                taxDetailsResponse.setTaxableAmount(linesDetails.ta
                taxDetailsResponse.setTaxAuthorityTypeId(String.val
                taxDetailsResponse.setTaxId(linesDetails.id);
                taxDetailsResponse.setTaxRegionId(linesDetails.regi
                taxDetailsResponses.add(taxDetailsResponse);

            }
              lineItemResponse.setTaxes(taxDetailsResponses);
              lineItemResponse.setEffectiveDate(date.valueof(linesToPr
              lineItemResponse.setIsTaxable(true);
                commercetax.AmountDetailsResponse amountResponse = 
                amountResponse.setTaxAmount(linesToProcess.taxCalcu
                amountResponse.setTotalAmount(linesToProcess.lineAmo
                amountResponse.setTotalAmountWithTax(linesToProcess
                amountResponse.setExemptAmount(linesToProcess.exempt
                lineItemResponse.setAmountDetails(amountResponse);
              lineItemResponse.setIsTaxable(linesToProcess.isItemTaxa
              lineItemResponse.setProductCode(linesToProcess.itemCode
              lineItemResponse.setTaxCode(linesToProcess.taxCode);
              lineItemResponse.setLineNumber(linesToProcess.lineNumber
              lineItemResponse.setQuantity(linesToProcess.quantity);
              lineItemResponses.add(lineItemResponse);
          }
          response.setLineItems(lineItemResponses);
          return response;
        }
        else
        {
          JsonErrorParser jsonErrorParserClass = JsonErrorParser.parse(re
          String message = null;
          if(String.isNotBlank(jsonErrorParserClass.error.message))
          {
             message=jsonErrorParserClass.error.message;
          }else{
                 String errorMessage = '';
                  for(JsonErrorParser.cls_details messageString : jsonErro
                  {
                      if(String.isNotBlank(messageString.message) )
                      {
                          errorMessage = messageString.message;
                      }
                  }
                  message = errorMessage;
              }
           return new commercetax.ErrorResponse(commercetax.resultcode.Ta

        }
      }else return null;
    }
    catch (Exception e)
    {
        throw e;
    }
  }
}
```

- In the `HttpService` class, replace the `test` value in the endpoint variable with the name of the `TaxTypedNamedCredential` record. This class contains the credentials that are required to access your Avalara account through Salesforce.

```
public with sharing class HttpService
{
```

```
private Map<String,String> mapOfHeaderParameter = new Map<String,String>();
private enum Method {GET, POST}

/**
* @name getInstance
* @description get an Instance of Service class
* @params NA
* @return Http Service Class Instance
*/
public static HttpService getInstance()
{
    if (NULL == httpServiceInstance)
    {
        httpServiceInstance =  new HttpService();
    }
    return httpServiceInstance;
}

/**
* @name get
* @description Get Method to get a HTTP request
*/
public void get(String endPoint)
{
    send(newRequest(Method.GET, endPoint));
}

/**
* @name post
* @description Post Method to Post a HTTP request
*/
public void post(String path, String requestBody)
{
   String endPoint = 'callout:commerce.tax.TaxTypedNamedCredential:test'+path;
    send(newRequest(Method.POST, endPoint, requestBody));
}

/**
* @name addHeader
* @description addHeader Methods to add all the defualt Header's required fo rth
*/
public void addHeader(String name, String value)
{
    mapOfHeaderParameter.put(name, value);
}

/**
* @name setHeader
* @description setHeader Methods to set setHeader for the request
*/
private void setHeader(HttpRequest request)
{
    for(String headerValue : mapOfHeaderParameter.keySet())
    {
        request.setHeader(headerValue, mapOfHeaderParameter.get(headerValue));
    }
}
/**
* @name newRequest
* @description newRequest Methods to make a new request
*/
private HttpRequest newRequest(Method method, String endPoint)
{
    return newRequest(method, endPoint, NULL);
}

/**
* @name newRequest
* @description newRequest Methods to make a new request
*/
private HttpRequest newRequest(Method method, String endPoint, String requestBo
{
```

```
                request.setBody(requestBody);
            }
            request.setTimeout(120000);
            return request;
        }

        /**
        * @name send
        * @description send Methods to send a request
        */
        private void send(HttpRequest request)
        {
            try
            {
                Http http = new Http();
                httpResponse = http.send(request);
            }
            catch(System.CalloutException e)
            {
                system.debug('callout exception happened' + e.getMessage());
            }
            catch(Exception e)
            {
                system.debug('callout did not happen' + e.getMessage());
            }
        }

        /**
        * @name getResponse
        * @description getResponse Method to get the Response
        */
        public HttpResponse getResponse()
        {
            return httpResponse;
        }

        /**
        * @name getResponseToString
        * @description getResponse Method to get the Responses
        */
        public String getResponseToString()
        {
            return getResponse().toString();
        }
    }
```

- Parse the `JsonSuccessParser` response object by using the `AvalaraJSONBuilder` class to build the response for your adapter.

  This example shows the `JsonSuccessParser` class.

```
global with sharing class JsonSuccessParser
{
  public static void consumeObject(JSONParser parser)
  {
    Integer depth = 0;
    do {
      JSONToken curr = parser.getCurrentToken();
      if (curr == JSONToken.START_OBJECT ||
        curr == JSONToken.START_ARRAY) {
        depth++;
      } else if (curr == JSONToken.END_OBJECT ||
        curr == JSONToken.END_ARRAY) {
        depth--;
      }
    } while (depth > 0 && parser.nextToken() != null);
  }
```

```
        public String region {get;set;}
        public String postalCode {get;set;}
        public String country {get;set;}
        public Integer taxRegionId {get;set;}

        public Addresses(JSONParser parser) {
            while (parser.nextToken() != JSONToken.END_OBJECT) {
                if (parser.getCurrentToken() == JSONToken.FIELD_NAME) {
                    String text = parser.getText();
                    if (parser.nextToken() != JSONToken.VALUE_NULL) {
                        if (text == 'id') {
                            id = parser.getText();
                        } else if (text == 'transactionId') {
                            transactionId = parser.getText();
                        } else if (text == 'boundaryLevel') {
                            boundaryLevel = parser.getText();
                        } else if (text == 'line1') {
                            line1 = parser.getText();
                        } else if (text == 'city') {
                            city = parser.getText();
                        } else if (text == 'region') {
                            region = parser.getText();
                        } else if (text == 'postalCode') {
                            postalCode = parser.getText();
                        } else if (text == 'country') {
                            country = parser.getText();
                        } else if (text == 'taxRegionId') {
                            taxRegionId = parser.getIntegerValue();
                        } else {
                            consumeObject(parser);
                        }
                    }
                }
            }
        }
    }

    public class Details {
        public String id {get;set;}
        public String transactionLineId {get;set;}
        public String transactionId {get;set;}
        public String country {get;set;}
        public String region {get;set;}
        public Integer exemptAmount {get;set;}
        public String jurisCode {get;set;}
        public String jurisName {get;set;}
        public String stateAssignedNo {get;set;}
        public String jurisType {get;set;}
        public Integer nonTaxableAmount {get;set;}
        public Double rate {get;set;}
        public Double tax {get;set;}
        public Integer taxableAmount {get;set;}
        public String taxType {get;set;}
        public String taxName {get;set;}
        public Integer taxAuthorityTypeId {get;set;}
        public Double taxCalculated {get;set;}
        public String rateType {get;set;}

        public Details(JSONParser parser) {
            while (parser.nextToken() != JSONToken.END_OBJECT) {
                if (parser.getCurrentToken() == JSONToken.FIELD_NAME) {
                    String text = parser.getText();
                    if (parser.nextToken() != JSONToken.VALUE_NULL) {
                        if (text == 'id') {
                            id = parser.getText();
                        } else if (text == 'transactionLineId') {
                            transactionLineId = parser.getText();
                        } else if (text == 'transactionId') {
                            transactionId = parser.getText();
                        } else if (text == 'country') {
                            country = parser.getText();
                        } else if (text == 'region') {
                            jurisName = parser.getText();
                        } else if (text == 'stateAssignedNo') {
                            stateAssignedNo = parser.getText();
                        } else if (text == 'jurisType') {
                            jurisType = parser.getText();
                        } else if (text == 'nonTaxableAmount') {
                            nonTaxableAmount = parser.getIntegerValue();
                        } else if (text == 'rate') {
                            rate = parser.getDoubleValue();
                        } else if (text == 'tax') {
                            tax = parser.getDoubleValue();
                        } else if (text == 'taxableAmount') {
                            taxableAmount = parser.getIntegerValue();
                        } else if (text == 'taxType') {
                            taxType = parser.getText();
                        } else if (text == 'taxName') {
                            taxName = parser.getText();
                        } else if (text == 'taxAuthorityTypeId') {
                            taxAuthorityTypeId = parser.getIntegerValue();
                        } else if (text == 'taxCalculated') {
                            taxCalculated = parser.getDoubleValue();
                        } else if (text == 'rateType') {
                            rateType = parser.getText();
                        } else {
                            consumeObject(parser);
                        }
                    }
                }
            }
        }
    }

    public class Messages {
        public String summary {get;set;}
        public String details {get;set;}
        public String refersTo {get;set;}
        public String severity {get;set;}
        public String source {get;set;}

        public Messages(JSONParser parser) {
            while (parser.nextToken() != JSONToken.END_OBJECT) {
                if (parser.getCurrentToken() == JSONToken.FIELD_NAME) {
                    String text = parser.getText();
                    if (parser.nextToken() != JSONToken.VALUE_NULL) {
                        if (text == 'summary') {
                            summary = parser.getText();
                        } else if (text == 'details') {
                            details = parser.getText();
                        } else if (text == 'refersTo') {
                            refersTo = parser.getText();
                        } else if (text == 'severity') {
                            severity = parser.getText();
                        } else if (text == 'source') {
                            source = parser.getText();
                        } else {
                            consumeObject(parser);
                        }
                    }
                }
            }
        }
    }

    public String id {get;set;}
    public String code {get;set;}
    public String referenceCode {get;set;}
    public Integer companyId {get;set;}
    public String taxDate {get;set;}
    public String transactionDate {get;set;}
    public String status {get;set;}
    public String type_Z {get;set;} // in json: type
    public Boolean reconciled {get;set;}
```

```
public Boolean locked {get;set;}
public Integer version {get;set;}
public String modifiedDate {get;set;}
public Integer modifiedUserId {get;set;}
public List<Lines> lines {get;set;}
public List<Addresses> addresses {get;set;}
public List<Summary> summary {get;set;}
public List<Messages> messages {get;set;}

public JsonSuccessParser(JSONParser parser) {
    while (parser.nextToken() != JSONToken.END_OBJECT) {
        if (parser.getCurrentToken() == JSONToken.FIELD_NAME) {
            String text = parser.getText();
            if (parser.nextToken() != JSONToken.VALUE_NULL) {
                if (text == 'id') {
                    id = parser.getText();
                } else if (text == 'code') {
                    code = parser.getText();
                } else if (text == 'referenceCode'){
                    referenceCode = parser.getText();
                } else if (text == 'companyId') {
                    companyId = parser.getIntegerValue();
                } else if (text == 'taxDate') {
                    taxDate = parser.getText();
                } else if (text == 'date') {
                    transactionDate = parser.getText();
                } else if (text == 'status') {
                    status = parser.getText();
                } else if (text == 'type') {
                    type_Z = parser.getText();
                } else if (text == 'reconciled') {
                    reconciled = parser.getBooleanValue();
                } else if (text == 'totalAmount') {
                    totalAmount = parser.getIntegerValue();
                } else if (text == 'totalExempt') {
                    totalExempt = parser.getIntegerValue();
                } else if (text == 'totalTax') {
                    totalTax = parser.getDoubleValue();
                } else if (text == 'totalTaxable') {
                    totalTaxable = parser.getIntegerValue();
                } else if (text == 'totalTaxCalculated') {
                    totalTaxCalculated = parser.getDoubleValue();
                } else if (text == 'adjustmentReason') {
                    adjustmentReason = parser.getText();
                } else if (text == 'locked') {
                    locked = parser.getBooleanValue();
                } else if (text == 'version') {
                    version = parser.getIntegerValue();
                } else if (text == 'modifiedDate') {
                    modifiedDate = parser.getText();
                } else if (text == 'modifiedUserId') {
                    modifiedUserId = parser.getIntegerValue();
                } else if (text == 'lines') {
                    lines = new List<Lines>();
                    while (parser.nextToken() != JSONToken.END_ARRAY) {
                        lines.add(new Lines(parser));
                    }
                } else if (text == 'addresses') {
                    addresses = new List<Addresses>();
                    while (parser.nextToken() != JSONToken.END_ARRAY) {
                        addresses.add(new Addresses(parser));
                    }
                } else if (text == 'summary') {
                    summary = new List<Summary>();
                    while (parser.nextToken() != JSONToken.END_ARRAY) {
                        summary.add(new Summary(parser));
                    }
                } else if (text == 'messages') {
                    messages = new List<Messages>();
                    while (parser.nextToken() != JSONToken.END_ARRAY) {
                        messages.add(new Messages(parser));
                    }
}

public class Summary {
    public String country {get;set;}
    public String region {get;set;}
    public String jurisType {get;set;}
    public String jurisCode {get;set;}
    public String jurisName {get;set;}
    public Integer taxAuthorityType {get;set;}
    public String stateAssignedNo {get;set;}
    public String taxType {get;set;}
    public String taxName {get;set;}
    public String taxGroup {get;set;}
    public String rateType {get;set;}
    public Integer taxable {get;set;}
    public Double rate {get;set;}
    public Double tax {get;set;}
    public Double taxCalculated {get;set;}
    public Integer nonTaxable {get;set;}
    public Integer exemption {get;set;}

    public Summary(JSONParser parser) {
        while (parser.nextToken() != JSONToken.END_OBJECT) {
            if (parser.getCurrentToken() == JSONToken.FIELD_NAME) {
                String text = parser.getText();
                if (parser.nextToken() != JSONToken.VALUE_NULL) {
                    if (text == 'country') {
                        country = parser.getText();
                    } else if (text == 'region') {
                        region = parser.getText();
                    } else if (text == 'jurisType') {
                        jurisType = parser.getText();
                    } else if (text == 'jurisCode') {
                        jurisCode = parser.getText();
                    } else if (text == 'jurisName') {
                        jurisName = parser.getText();
                    } else if (text == 'taxAuthorityType') {
                        taxAuthorityType = parser.getIntegerValue();
                    } else if (text == 'stateAssignedNo') {
                        stateAssignedNo = parser.getText();
                    } else if (text == 'taxType') {
                        taxType = parser.getText();
                    } else if (text == 'taxName') {
                        taxName = parser.getText();
                    } else if (text == 'taxGroup') {
                        taxGroup = parser.getText();
                    } else if (text == 'rateType') {
                        rateType = parser.getText();
                    } else if (text == 'taxable') {
                        taxable = parser.getIntegerValue();
                    } else if (text == 'rate') {
                        rate = parser.getDoubleValue();
                    } else if (text == 'tax') {
                        tax = parser.getDoubleValue();
                    } else if (text == 'taxCalculated') {
                        taxCalculated = parser.getDoubleValue();
                    } else if (text == 'nonTaxable') {
                        nonTaxable = parser.getIntegerValue();
                    } else if (text == 'exemption') {
                        exemption = parser.getIntegerValue();
                    } else {
                        consumeObject(parser);
                    }
                }
            }
        }
    }
}

public class Lines {
    public String id {get;set;}
    public String transactionId {get;set;}
        public Double quantity {get;set;}
        public String reportingDate {get;set;}
        public Double tax {get;set;}
        public Integer taxableAmount {get;set;}
        public Double taxCalculated {get;set;}
        public String taxCode {get;set;}
        public String taxDate {get;set;}
        public Boolean taxIncluded {get;set;}
        public List<Details> details {get;set;}
        public String itemCode {get;set;}
        public Lines(JSONParser parser) {
            while (parser.nextToken() != JSONToken.END_OBJECT) {
                if (parser.getCurrentToken() == JSONToken.FIELD_NAME) {
                    String text = parser.getText();
                    if (parser.nextToken() != JSONToken.VALUE_NULL) {
                        if (text == 'id') {
                            id = parser.getText();
                        } else if (text == 'transactionId') {
                            transactionId = parser.getText();
                        }else if (text == 'itemCode') {
                            itemCode = parser.getText();
                        }else if (text == 'lineNumber') {
                            lineNumber = parser.getText();
                        } else if (text == 'discountAmount') {
                            discountAmount = parser.getIntegerValue();
                        } else if (text == 'exemptAmount') {
                            exemptAmount = parser.getIntegerValue();
                        } else if (text == 'exemptCertId') {
                            exemptCertId = parser.getIntegerValue();
                        } else if (text == 'isItemTaxable') {
                            isItemTaxable = parser.getBooleanValue();
                        } else if (text == 'lineAmount') {
                            lineAmount = parser.getIntegerValue();
                        } else if (text == 'quantity') {
                            quantity = parser.getDoubleValue();
                        } else if (text == 'reportingDate') {
                            reportingDate = parser.getText();
                        } else if (text == 'tax') {
                            tax = parser.getDoubleValue();
                        } else if (text == 'taxableAmount') {
                            taxableAmount = parser.getIntegerValue();
                        } else if (text == 'taxCalculated') {
                            taxCalculated = parser.getDoubleValue();
                        } else if (text == 'taxCode') {
                            taxCode = parser.getText();
                        } else if (text == 'taxDate') {
                            taxDate = parser.getText();
                        } else if (text == 'taxIncluded') {
                            taxIncluded = parser.getBooleanValue();
                        } else if (text == 'details') {
                            details = new List<Details>();
                            while (parser.nextToken() != JSONToken.END_ARRAY) {
                                details.add(new Details(parser));
                            }
                        } else {
                            consumeObject(parser);
                        }
                    }
                }
            }
        }
    }


    public static JsonSuccessParser parse(String json)
    {
        return new JsonSuccessParser(System.JSON.createParser(json));
    }
}
```

```
public with sharing class AvalaraJSONBuilder
{
    private static AvalaraJSONBuilder avalaraJSONBuilderInstance;

    public static AvalaraJSONBuilder getInstance()
    {
        if (NULL == avalaraJSONBuilderInstance)
        {
            avalaraJSONBuilderInstance = new AvalaraJSONBuilder();
        }
        return avalaraJSONBuilderInstance;
    }

    public String frameJsonForGetTaxOrderItem(commercetax.CalculateTaxRequest calcu
    {
        try
        {
            Id accountid  = null;
            if(calculateTaxRequest.CustomerDetails.AccountId != null &&  calculateTa
               accountid = Id.valueof(calculateTaxRequest.CustomerDetails.AccountId
            JSONGenerator jsonGeneratorInstance = JSON.createGenerator(true);
            jsonGeneratorInstance.writeStartObject();
            String type = null;
            if(calculateTaxRequest.taxtype == commercetax.CalculateTaxType.Actual)
                type ='SalesInvoice';
                else type = 'SalesOrder';
            jsonGeneratorInstance.writeStringField('type', type);
            if(calculateTaxRequest.SellerDetails != null)
                jsonGeneratorInstance.writeStringField('companyCode', calculateTaxR
            else
                jsonGeneratorInstance.writeStringField('companyCode', 'billing2');
            if(calculateTaxRequest.isCommit != null) {
                jsonGeneratorInstance.writeBooleanField('commit', calculateTaxReque
            }
            if(calculateTaxRequest.documentcode != null){
                jsonGeneratorInstance.writeStringField('code', calculateTaxRequest.
            }else if(calculateTaxRequest.referenceEntityId != null) {
                jsonGeneratorInstance.writeStringField('code', calculateTaxRequest.
            }
            if(calculateTaxRequest.CustomerDetails.code == null && accountid !=null
                Account acc = [select id, name from account where id=:accountid];
                jsonGeneratorInstance.writeStringField('customerCode', acc.name);
            } else {
                jsonGeneratorInstance.writeStringField('customerCode', calculateTax
            }
            if(calculateTaxRequest.EffectiveDate == null)
                jsonGeneratorInstance.writeDateField('date', system.today());
            else
                jsonGeneratorInstance.writeDateTimeField('date', calculateTaxReques

            jsonGeneratorInstance.writeFieldName('lines');
            jsonGeneratorInstance.writeStartArray();
            for(integer i=0;i<1;i++){
                for(Commercetax.TaxLineItemRequest lineItem : calculateTaxRequest.L
                {
                    jsonGeneratorInstance.writeStartObject();
                    if(lineItem.linenumber != null){
                        jsonGeneratorInstance.writeStringField('number', lineItem.l
                    }
                    jsonGeneratorInstance.writeNumberField('quantity', lineItem.Qua
                    jsonGeneratorInstance.writeNumberField('amount', (lineItem.Amou
                    jsonGeneratorInstance.writeStringField('taxCode',lineItem.taxCo

                    jsonGeneratorInstance.writeFieldName('addresses');
                    jsonGeneratorInstance.writeStartObject();
                    jsonGeneratorInstance.writeFieldName('ShipFrom');
                    jsonGeneratorInstance.writeStartObject();
                    jsonGeneratorInstance.writeStringField('line1', lineItem.addres
                    jsonGeneratorInstance.writeStringField('line2', lineItem.addres
                    jsonGeneratorInstance.writeStringField('city', lineItem.address
                    jsonGeneratorInstance.writeStringField('region', lineItem.addre
                    jsonGeneratorInstance.writeStringField('country', lineItem.addr
                    jsonGeneratorInstance.writeStringField('line2', lineItem.addres
                    jsonGeneratorInstance.writeStringField('city', lineItem.address
                    jsonGeneratorInstance.writeStringField('region', lineItem.addre
                    jsonGeneratorInstance.writeStringField('country', lineItem.addr
                    jsonGeneratorInstance.writeStringField('postalCode',lineItem.ad
                    jsonGeneratorInstance.writeEndObject();

                    jsonGeneratorInstance.writeFieldName('pointOfOrderOrigin');
                    jsonGeneratorInstance.writeStartObject();
                    jsonGeneratorInstance.writeStringField('line1', lineItem.addres
                    jsonGeneratorInstance.writeStringField('line2', lineItem.addres
                    jsonGeneratorInstance.writeStringField('city', lineItem.address
                    jsonGeneratorInstance.writeStringField('region', lineItem.addre
                    jsonGeneratorInstance.writeStringField('country', lineItem.addr
                    jsonGeneratorInstance.writeStringField('postalCode',lineItem.ad
                    jsonGeneratorInstance.writeEndObject();


                    if(lineItem.effectiveDate != null)
                    {
                        jsonGeneratorInstance.writeFieldName('taxOverride');
                        jsonGeneratorInstance.writeStartObject();
                        jsonGeneratorInstance.writeDateTimeField('taxDate', lineIte
                        jsonGeneratorInstance.writeEndObject();
                    }
                    jsonGeneratorInstance.writeEndObject();
                    jsonGeneratorInstance.writeEndObject();
                }
            }
                jsonGeneratorInstance.writeEndArray();
            jsonGeneratorInstance.writeEndObject();
            return jsonGeneratorInstance.getAsString();
        }
        catch (Exception e)
        {
             throw e;
        }
    }
}
```

- Use the `JsonErrorParser` class to extract the error details, if any.

```
global with sharing class JsonErrorParser
{
    public cls_error error;

    public class cls_error
    {
        public String code;
        public String message;
        public String target;
        public cls_details[] details;
    }

    public class cls_details
    {
        public String code;
        public String message;
        public String description;
        public String faultCode;
        public String helpLink;
        public String severity;
    }
    public static JsonErrorParser parse(String json)
    {
        return (JsonErrorParser) System.JSON.deserialize(json, JsonErrorParser.clas
```

DID THIS ARTICLE SOLVE YOUR ISSUE?
Let us know so we can improve!

Share your feedback

DEVELOPER CENTERS
Heroku
MuleSoft
Tableau
Commerce Cloud
Lightning Design System
Einstein
Quip

POPULAR RESOURCES
Documentation
Component Library
APIs
Trailhead
Sample Apps
Podcasts
AppExchange

COMMUNITY
Trailblazer Community
Events and Calendar
Partner Community
Blog
Salesforce Admins
Salesforce Architects

Privacy Information Terms of Service Legal Use of Cookies Trust Cookie Preferences

Your Privacy Choices Responsible Disclosure Contact