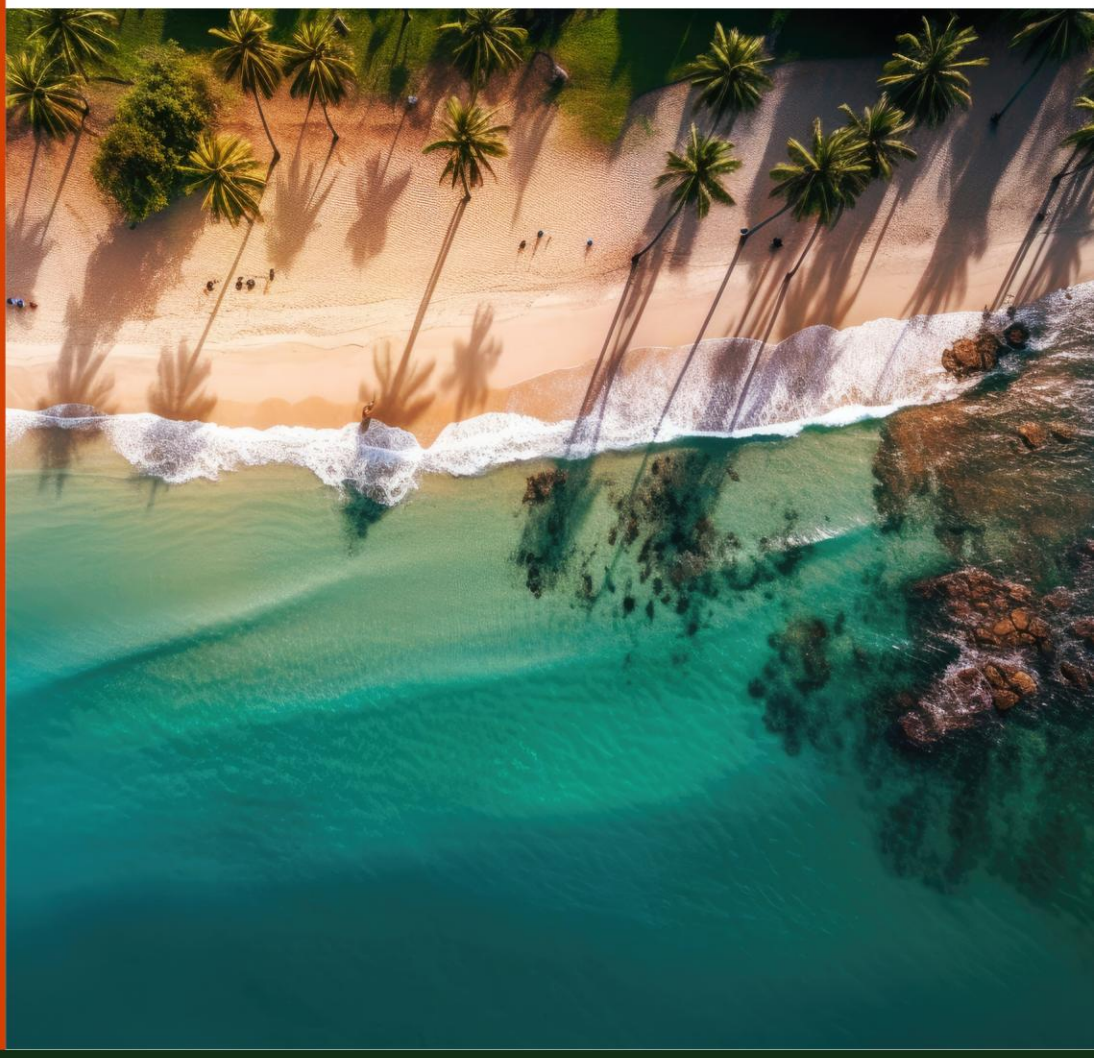
Thomson Reuters™

ONESOURCE IDT SAP Commerce Cloud Integration

SETUP, CONFIGURATION AND USER GUIDE

2.0.0.1



2026 Thomson Reuters/ONESOURCE. All Rights All Rights Reserved. Proprietary and confidential information of Thomson Reuters. Disclosure, use, or reproduction without the written authorization of TR /S is prohibited. In compliance with the license agreements for the Open Source Libraries leveraged by Thomson Reuters, our customers can obtain copies of these libraries by contacting Customer Support at https://tax.thomsonreuters.com/support/onesource/customer-center/.

DOCUMENT HISTORY

VERSION NUMBER	VERSION DATE	SUMMARY
V1	November 2020	Initial release
V2	March 2021	Updates to the guide
V3	June 2021	Updates to the guide
V4	October 2021	Updates to the guide
V5	December 2021	Updates to the guide
V6	February 2022	Updates to the guide
V7	March 2023	Updates to the guide
V8	January 2026	SAP Commerce Cloud version 2211-jdk21 upgrade
V9	June 2026	Minor corrections
V10	September 2026	Minor corrections

TABLE OF CONTENTS

INTRODUCTION ---------- **1**

WELCOME TO ONESOURCE INTEGRATION FOR SAP COMMERCE CLOUD ---------- 1
WHO SHOULD READ THIS GUIDE? ---------- 1
RESOURCES ---------- 1
PREREQUISITES AND STEPS ---------- 2
SUPPORTED SAP COMMERCE CLOUD VERSIONS ---------- 2

ARCHITECTURE AND DESIGN OVERVIEW ---------- **3**

SAP COMMERCE CLOUD EXTENSIONS ---------- 3
CONNECTION METHOD – REST VS SOAP ---------- 5
ASSUMPTIONS AND KNOWN LIMITATIONS ---------- 5

INSTALLATION ---------- **7**

LOCAL ENVIRONMENT INSTALLATION ---------- 7
Activating External Tax Calls ---------- *7*
SAP PUBLIC CLOUD ENVIRONMENT INSTALLATION ---------- 8
SETTING UP THE AUTHENTICATION ---------- 8
REST ---------- *8*
SOAP ---------- *9*

BUSINESS PROCESS DESCRIPTION ---------- **10**

MAIN BUSINESS PROCESSES ---------- 10
Checkout ---------- *10*
Order Processing ---------- *11*
Returns ---------- *12*
OTHER FEATURES ---------- 13
Net vs Gross Pricing ---------- *13*
Taxes for the Delivery Cost ---------- *13*
Summarization Options for Tax Authority Details ---------- *15*
Buyer and Seller Registrations ---------- *16*
Sending INCOTERMS ---------- *16*
Audit Calls in Orders ---------- *17*
Order Cancellations ---------- *18*
Recalculation of Cart/Order Totals ---------- *18*
Returns Process ---------- *19*
Refunding the Delivery Cost in Returns ---------- *20*
Backoffice Enhancements ---------- *21*
Reconciliation Report ---------- *22*

CONFIGURATION OPTIONS ---------- **25**

BASE STORE SPECIFIC CONFIGURATION ---------- 32

DEVELOPMENT AND EXTENSIBILITY ---------- **33**

CUSTOM ATTRIBUTES ---------- 33
SUPPORT FOR SPARTACUS UI AND OMNI COMMERCE CONNECT (OCC) REST API ---------- 35

LOGGING AND MONITORING ---------- **36**

CONSOLE LOGGING ---------- 36
KIBANA DASHBOARDS ---------- 37

Errors Dashboard ---------- *37*
Requests and Responses Dashboards ---------- *38*
AVS Dashboard ---------- *39*

TROUBLESHOOTING SCENARIOS ---------- 41

CLASSIFICATION OF HIGHER-LEVEL EXCEPTIONS ---------- 41
REQUEST PREPARATION EXCEPTIONS ---------- 41
CONNECTION EXCEPTIONS ---------- 42
DETERMINATION ERRORS ---------- 44
RESPONSE PROCESSING EXCEPTIONS ---------- 44
OTHER PROBLEMS ---------- 45
SYSTEM BEHAVIOR ON INTEGRATION FAILURES ---------- 47
Common Behavior ---------- *47*
Specific Cases (After Unsuccessful Trials) ---------- *47*

ADDRESS VALIDATION SERVICE (AVS) ---------- 48

AVS BUSINESS PROCESS DESCRIPTION ---------- 48
Checkout ---------- *48*
My Account / Address Book ---------- *49*
AVS SERVICE CONNECTION METHODS – SOAP VS REST ---------- 49
AVS CONFIGURATION OPTIONS ---------- 49
AVS UI MESSAGES ---------- 52
Accelerator Based UI ---------- *52*
Spartacus UI ---------- *53*
AVS LOGGING AND MONITORING ---------- 53

APPENDIX 1: TAX DATA MAPPING – REST ---------- 55

APPENDIX 2: TAX DATA MAPPING – SOAP ---------- 61

APPENDIX 3: AVS DATA MAPPING – REST ---------- 68

APPENDIX 4: AVS DATA MAPPING – SOAP ---------- 69

INTRODUCTION

WELCOME TO ONESOURCE INTEGRATION FOR SAP COMMERCE CLOUD

This is an overview of the tax calculation integration between SAP Commerce Cloud and ONESOURCE Indirect Tax Determination (Determination). Thomson Reuters ONESOURCE™ Indirect Tax offers a comprehensive, cloud-based transaction tax management solution that seamlessly integrates for accurate sales tax calculation, easy document management, effortless filing and remittance.

This integration will support the following events from the SAP Commerce Platform.

- Estimate Tax Calculation Call for Carts and Orders
- Audited Tax Calculation Call for Orders
- Estimate and Audited Tax Calls for Return Requests

The solution supports the following countries for tax integration:

- United States
- Canada
- European Union (VAT) region

The ONESOURCE Integration for SAP Commerce Cloud has been validated and certified by SAP to be placed on the SAP Certified Solutions Directory. Details can be found at https://www.sap.com/products/crm/partners/thomson-reuters-tax-accounting-inc-onesource-indirect-tax-integration-for-sap-commerce-cloud.html?countryCode=US

WHO SHOULD READ THIS GUIDE?

If you are responsible for overseeing setting up ONESOURCE Integration for SAP Commerce, you will need to coordinate with the following people:

- SAP Commerce Business Analyst
- SAP Commerce Solutions Architect
- Other Technical Resources
- Tax Professionals

Make this guide available to each of these contributors to ensure you have a successful installation.

RESOURCES

RESOURCE	DESCRIPTION
Customer Support	Look for answers in the Knowledge Base, or to open a support ticket.

SUPPORT PROTOCOL

The ONESOURCE Integration for SAP Commerce Cloud is built, maintained, and owned by Thomson Reuters Tax & Accounting Indirect Tax. The business unit has a dedicated group of SAP Commerce Business Systems Analysts, Java Programmers, and Quality Assurance staff who built this product. We follow SAP's best practices, development standards, and strive to minimize the impact this solution will have on your SAP environment. In the case of an issue with the ONESOURCE Integration for SAP Commerce please follow these simple steps to open a support ticket with Thomson Reuters:

1. Identify the potential issue and gather all necessary facts (log files, scenarios, configurations, screen prints).
2. Provide steps to reproduce the scenario leading to the issue.
3. Provide system environment information such as your SAP Commerce Cloud release version.
4. Open a support ticket with Indirect Tax at https://tax.thomsonreuters.com/support/onesource/indirect-tax/.

PREREQUISITES AND STEPS

For the successful implementation of the ONESOURCE IDT Integration with SAP Commerce, ensure you have procured the necessary packages and followed the implementation steps mentioned below:

PLATFORM	COMPONENT	DESCRIPTION
SAP Commerce Cloud	Local and Cloud environment	• ONESOURCE IDT Integration extensions should be added to the system setup. • Building and updating the local system to activate the extensions. • Maintenance of system configuration for various integration options including connection details and authentication.
ONESOURCE Determination	Configuration	• The External Company ID setup that uniquely identifies the tax request within the Thomson Reuters solution for audit, reporting, and returns purposes. • Authentication details are required to set up the integration

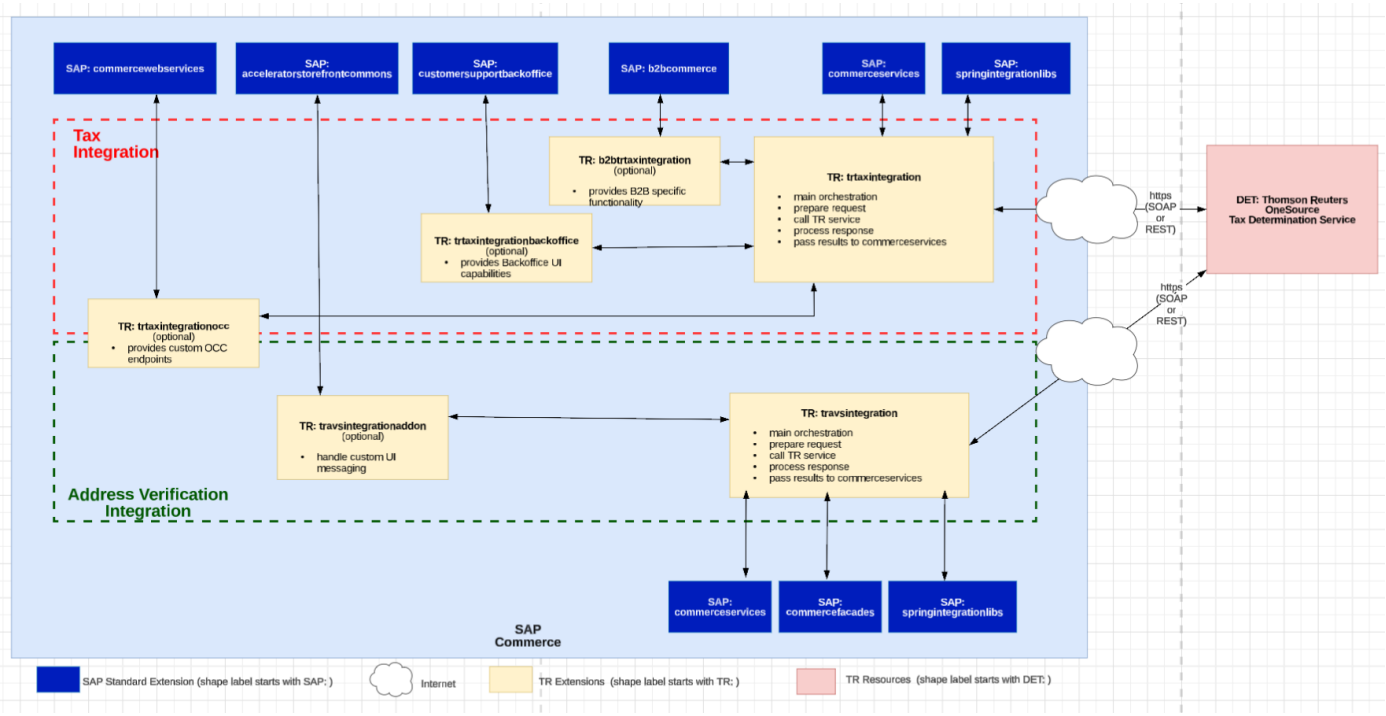
SUPPORTED SAP COMMERCE CLOUD VERSIONS

SAP Commerce Cloud 2211-jdk21.4

ARCHITECTURE AND DESIGN OVERVIEW

SAP COMMERCE CLOUD EXTENSIONS

The following diagrams depict the touch-points between the extensions within SAP Commerce and their integration with and ONESOURCE Tax.



To enable external tax calls to ONESOURCE IDT, Thomson Reuters provides following extensions to be included in your SAP Commerce setup.

For tax integration:

- **trtaxintegration**: This extension is mandatory for the tax services integration and it supports both B2C and B2B scenarios. It includes following functionality:
    - Implements SAP's External Tax Service interface.
    - Responsible to populate request details common for B2C and B2B
    - Responsible to make the call to the external service.
    - Responsible to convert the response to SAP's data structures.
    - Requires and auto-loads following SAP extension: commerceservices and springintegrationlibs.
- **b2btrtaxintegration**: This extension is optional and should only be used if you have a B2B Storefront. It includes following functionality:

  - Responsible to populate request details for B2B processes.
  - Requires trtaxintegration as a prerequisite.
  - Requires and auto loads following SAP extension: b2bcommerce.

- **trtaxintegrationbackoffice**: This extension is optional and should only be used if you would like to use Backoffice UI enhancements provided within the package. Includes following functionality:
  - Action buttons to retrigger tax integrations for orders and return requests.
  - Action buttons to recalculate totals and call the external tax service for carts and orders.
  - Requires trtaxintegration as a prerequisite.
  - Requires and auto loads following SAP extension: customersupportbackoffice
- **trtaxintegrationocc**: This extension is optional and should only be used if you would like to use the custom OCC endpoints provided within the package.
  - Provides REST end-points to get tax details for a cart, order and return request.
  - Provides an alternative REST end-point that extends the SAP standard OCC end-point and provides extra information regarding the reason for an invalid address verification attempt.
  - Requires trtaxintegration as a prerequisite.
  - Requires and auto loads following SAP extension: commercewebservices.

In addition, if you plan to activate address verification service integration, the following extensions should also be considered.

- **travsintegration**: This extension is mandatory for the address verification integration and it supports both B2C and B2B scenarios.
  - Implements SAP's External Address Verification Service interface.
  - Responsible to populate request details.
  - Responsible to make the call to the external service.
  - Responsible to convert the response into SAP's data structures.
  - Requires and auto loads following SAP extensions: commerceservices, commercefacades and springintegrationlibs.
- **travsintegrationaddon**: This extension is optional and should only be used if you have Accelerator-based storefronts on the UI layer. The extension is not needed for Spartacus UI Storefronts. (trtaxintegrationocc extension provides the same feature for Spartacus)
  - Provides extra information regarding the reason for an invalid address.

- Requires and auto loads following SAP extension: acceleratorstorefrontcommons, addonsupport

For all integration touchpoints, the call is always triggered and managed by the backend layer (SAP Commerce application servers). The solution provides no frontend (Javascript) integration with ONESOURCE IDT system.

CONNECTION METHOD – REST VS SOAP

ONESOURCE Determination tax service provides two different connection methods: SOAP and REST.

SAP Commerce Integration for ONESOURCE Determination extensions support both options. Clients can choose either of the method depending on their requirements. Please note that only one option can be active at a specific time.

"**thomsonreuters.idt.rest.active**" system property determines which method is active on the system. Maintain this item in your local.properties or on manifest.json file for cloud deployments. If this configuration is true, REST connection will be active, otherwise the tax call will be made in SOAP.

Keep in mind that each option has a different set of configuration items for connection and authentication. Please check "Setting up the Authentication" sections for the details.

Since the structure of the request / response messages of these options are different, the system behavior may also differ for certain cases. Please refer the relevant section of this document for the details.

One major difference between two options is about the logging of request and response messages. For REST method the logs are in JSON format whereas for SOAP the logs will be in XML.

ASSUMPTIONS AND KNOWN LIMITATIONS

- ONESOURCE IDT Integration for SAP Commerce is only supported for SAP Public Cloud Deployments. Although it may be assumed that the extensions provided would work in an on-premise environment, the functionality has not been tested in an on-premise cluster.
- On storefront, the tax integration is only limited to the checkout pages. No tax calls will be triggered on other pages of the storefront.
- The tax integration supports having multiple delivery addresses in a cart/order. Since this functionality is not supported by SAP in the out-of-the-box storefronts, it is assumed that the checkout steps are customized by the client to allow the user to enter multiple delivery addresses on the cart. Details can be found in the following sections.
- For Address Verification Integration, with SOAP connections, only United States addresses can be validated. On the other hand, through REST connections, it is possible to validate US, Canada and International addresses.
- The address verification integration is only limited to the storefront. There is no such integration in Backoffice UI as part of this integration package.

- Integration version 2.* only supports SAP Commerce Cloud 2211jdk21 versions. For earlier 2211 versions, please use the integration version 1.4.0.0.

INSTALLATION

LOCAL ENVIRONMENT INSTALLATION

- Decide which extensions are required for your setup. Please refer to the “Architecture and Design Overview” section of this guide.

- Extract the relevant extension folders into the “$HYBRIS_HOME/bin/custom” folder.

- Add your extensions to the localextensions.xml file. Example:
```
<extension name='trtaxintegration' />
<extension name='b2btrtaxintegration' />
<extension name='trtaxintegrationbackoffice' />
<extension name='trtaxintegrationocc' />
```

- For address verification, add the following lines to your localextensions.xml file. ’*travsintegration*’ is mandatory for the integration while the ’*travsintegrationaddon’* is optional.
```
<extension name='travsintegration' />
<extension name='travsintegrationaddon' />
```

- If you decided to add *travsintegrationaddon, r*un the following ant command to install the addon for the required storefront extension.
  Ex: ant addoninstall -Daddonnames="travsintegrationaddon" -DaddonStorefront.yacceleratorstorefront="{{myextension}}" (Replace {{myextension}} with the name of the storefront extension in your setup.)

- Build your system with “ant all".

- Update your system with “ant updatesystem”.

- ONESOURCE IDT Integration for SAP Commerce is only supported for SAP Public Cloud Deployments. Although it may be assumed that the extensions provided would work in an on-premise environment, the functionality has not been tested in an on-premise cluster.

Activating External Tax Calls

For ONESOURCE IDT external tax call to ONESOURCE IDT to be triggered in the system, the following prerequisites should be met:

- External Tax Calling should be activated for the base store (BaseStore.externalTaxEnabled should be true)

- External Company ID should be defined in system configuration. Check “Base Store Specific Configuration” section for details on how to differentiate External Company ID for each Base Store.

- The cart should have at least a delivery address and a delivery mode for the tax call to be triggered. In case of a pickup in-store scenario where delivery addresses are not relevant, a point of service address should already be maintained.

- Decide which extensions are required for your setup. Please refer to the “Architecture and Design Overview” section of this guide.

SAP PUBLIC CLOUD ENVIRONMENT INSTALLATION

- Make sure you committed extension folders in your code repository.
- Update your manifest.json file to include the extensions in your setup. This file should be located in $HYBRIS_HOME folder. Also, make sure that your configuration properties described in section “Configure SAP Commerce for ONESOURCE IDT Integration” are included in your setup.

SETTING UP THE AUTHENTICATION

REST

Thomson Reuters implements the OAuth2 protocol for ONESOURCE APIs that funnels requests to APIs through an Authorization server on ONESOURCE Platform.

As an admin user, you can login to the ONESOURCE portal and use the API Administrator tile to create a machine application and generate the credentials that will be used to authenticate the integration calls.

Starting with version 2.0.0.0, SAP Commerce Integration supports **Client ID and Secret Authentication** method only. The client assertion method, which was the default method in previous versions, is deprecated and is not supported anymore. Customers who are upgrading from the previous version need to generate a new machine user with client ID and secret in both UAT and PROD environments. Please visit ONESOURCE Portal, API Administration tile for machine users.

In the machine application, you can generate the Consumer Key (Client ID) and Consumer Secret (Client secret) and use these credentials to maintain **thomsonreuters.idt.rest.url, thomsonreuters.idt.rest.clientid**, and **thomsonreuters.idt.rest.clientsecret** properties in local.properties file. For cloud deployments you can provide these properties in SAP Cloud Portal.

Start During application start-up, these properties are detected by the trtaxintegration extension and respective beans will be created. For that reason, if you need to change these values, please make sure that the application is restarted for the changes to be in effect.

SOAP

For setting up authentication for SOAP connections, you'll need certain details such as URL for the connection, username and password. Please contact Thomson Reuters support in case you don't know any of these details.

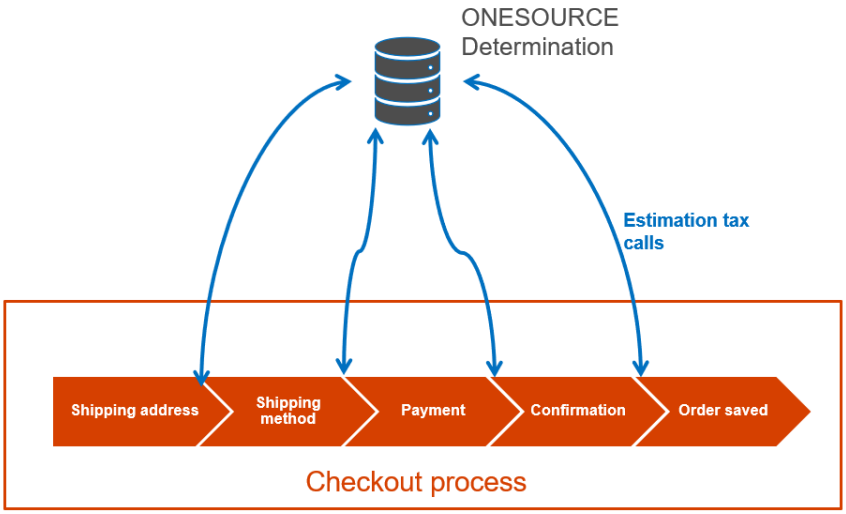
Once you have your username and password, you need to maintain three configuration parameters either in your local.properties file or in SAP Commerce Portal for cloud installations:

- thomsonreuters.idt.soap.username
- thomsonreuters.idt.soap.password
- thomsonreuters.idt.soap.url

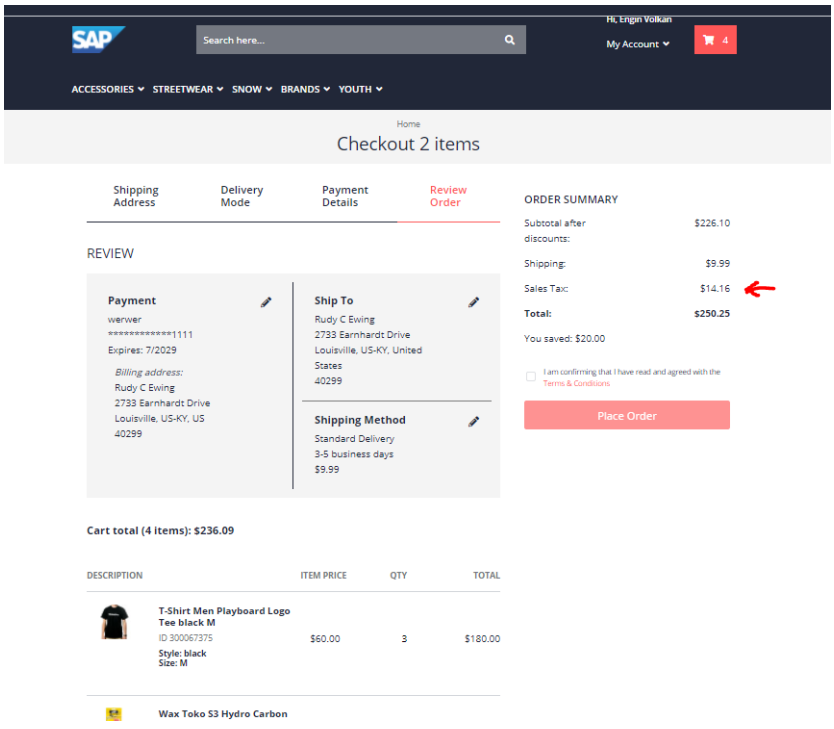
BUSINESS PROCESS DESCRIPTION

MAIN BUSINESS PROCESSES

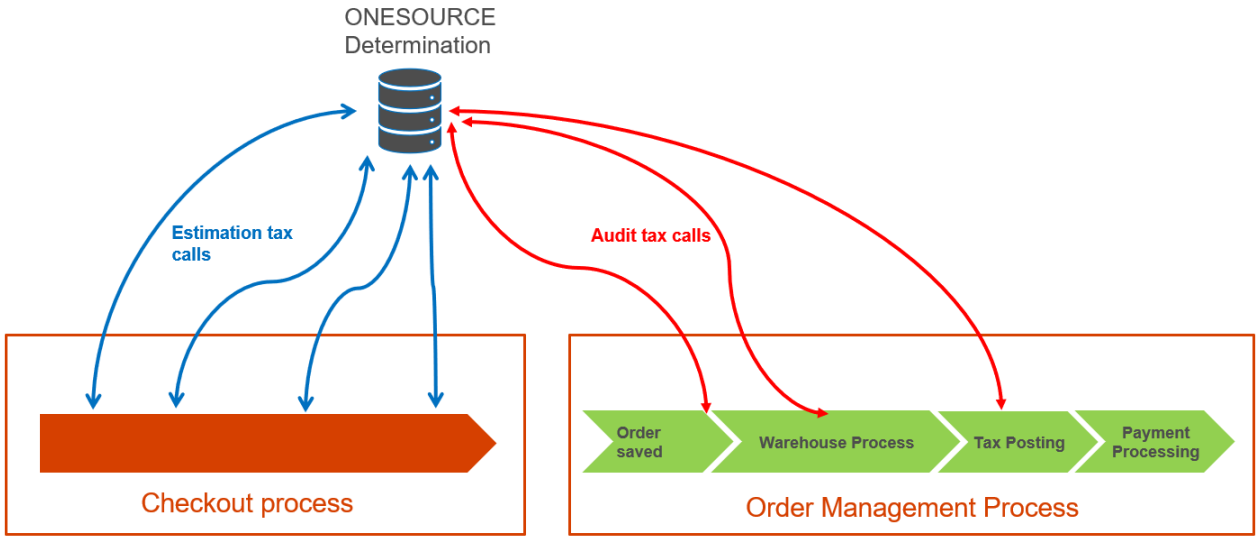
Checkout



During a checkout process, both in B2C and B2B storefronts, several estimation tax requests are sent to the Determination Engine to calculate the exact sales tax. After each call, Determination response is read, converted to SAP's format and persisted on the cart object. The result for the total tax value is displayed in the "Order Summary" section of the next checkout step.



Order Processing



If the order fulfillment process is managed by SAP Commerce, it is possible to do the tax auditing for ONESOURCE Determination. On the other hand, if the order is sent to an external system for processing, this step may not be necessary.

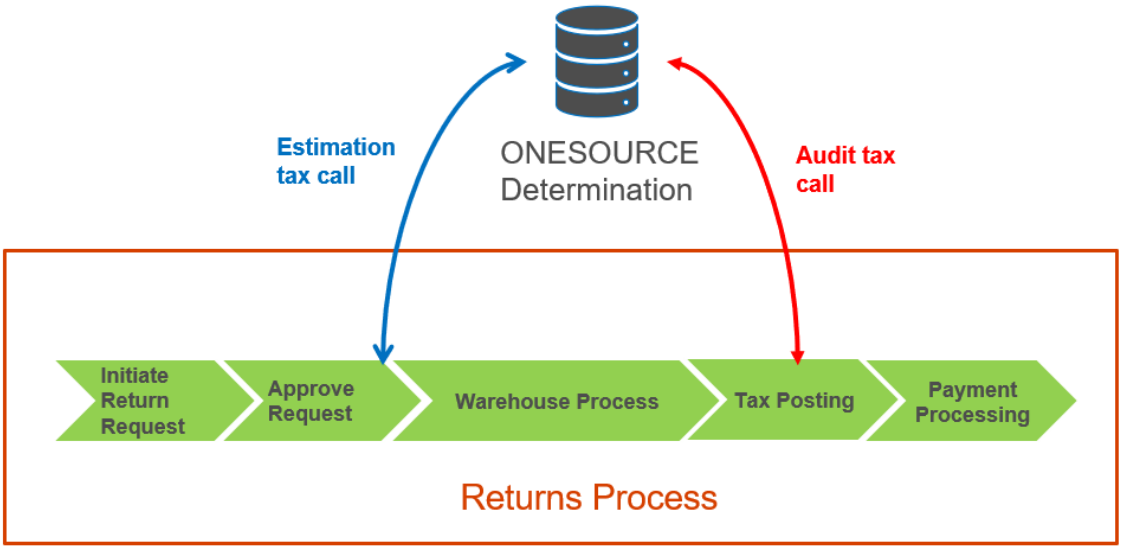
Once the order is saved in SAP Commerce, an Order Business Process is triggered associated with this order which orchestrates all the tasks that need to be carried out for fulfilling the order. The major steps include e-mail notifications, assignment of orders to warehouses, warehouse process, tax postings and payment processing.

In SAP's standard process flow, tax posting happens right after the warehouse process is completed and shipment is confirmed. This is also the default and the preferred method in ONESOURCE IDT integration. On the other hand, a modified process flow is provided as a part of the integration package which makes it possible to shift tax posting right after the order save and send updates in case any changes occur further in the process flow. Please check section "Audit Calls in Orders" of this document for details.

Compared to the calls in the checkout process, at the tax posting step, the following differences should or might be observed in the tax calls related with an order:

- The tax call is audited, which means the document is persisted on the ONESOURCE Determination database.
- The tax call request is prepared with the last version of the order. Any change that might happen on the order from the first save is considered such as partial cancellations, order quantity updates or pricing / shipment cost changes.

Returns



If the return process is managed by SAP Commerce, it is possible to do the tax auditing for ONESOURCE Determination. On the other hand, if the returns process is not activated in SAP Commerce or the return is transferred to an external system for processing, this step may not be necessary.

Initiating the return request may happen in two channels: Storefront via returns self-service page or Customer Support agent using Customer Service Backoffice role. In both cases, a Return Request document is created in SAP Commerce and a Return Business Process is started in the background.

As part of the integration package a custom return process is provided which allows you to make an estimation call to ONESOURCE Determination once the return is approved by customer service. As the process advances, the goods arrive to the warehouse and once the acceptance of the goods is completed in the warehouse, this time an audit call is sent to post the realized taxes and a record for the return request is also created in the Determination Database.

At both estimation and audit calls, the taxes are calculated and persisted on the return request document.

Please note that in Self-Service Returns process, the external tax calculation is not triggered by the SAP's standard flow and no tax value is presented to the user at the time of return request creation. For this reason, all the tax calculations are happening in the business process which runs as a background process. On the other hand, if you are using Spartacus UI for your storefront, you can make use of the custom OCC endpoint provided as a part of this integration package to present already calculated tax values in your Return Request Details page under My Account section of your site. Please check the details in "Support for Spartacus UI and Omni Commerce Connect (OCC) REST API" section of this document.

OTHER FEATURES

Net vs Gross Pricing

The ONESOURCE Determination integration supports both net pricing and gross pricing options in documents.”net” field on AbstractOrderModel is considered while preparing the tax request.

In the REST request:

- processingOptions / documentAmountType field is set either as GrossAmount or GrossPlusTaxAmount depending on the net/gross settings of the cart/order.

  Here is an example section from a request JSON for a net cart:

```
"processingOptions": {
    "chargeIncludedInAmounts": false,
    "chargeResponse": "SeparateAuthority",
    "responseSummary": "SummaryByAuthorityAndZone",
    "documentAmountType": "GrossAmount"
}
```

  All the monetary values in the request JSON will be interpreted accordingly by Determination Engine.

In the SOAP request:

- If the cart/order runs on net pricing
  - <CALCULATION_DIRECTION> is set as “F – Forward”
  - <GROSS_AMOUNT> tag is used for the item amount or discount
- If the cart/order runs on gross pricing
  - <CALCULATION_DIRECTION> is set as “R – Reverse”
  - <GROSS_PLUS_TAX > tag is used for the item amount or discount

Taxes for the Delivery Cost

The ONESOURCE Determination integration supports tax calculation for delivery costs. There two possible options. “thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems” configuration item is used to determine the system behavior (default is false).

- **Header Delivery Cost:** For each tax request, delivery cost is assumed to be valid for the whole cart/order
  - AbstractOrderModel.deliveryCost field is used as the delivery cost.

- The product/commodity code for the delivery cost item is determined by the "thomsonreuters.idt.request.shipping.productcode" system property.
- The tax results are kept as TaxValue's in cart/order header (AbstractOrderModel.totalTaxValues)

- **Distributed delivery cost per item:** A separate charge section is added for each delivery related item in cart/order, an additional item is inserted in the request.
  - AbstractOrderModel.deliveryCost field is distributed to each delivery item proportionally based on their item value.
  - The product/commodity code and quantities are maintained the same as the first method.
  - Pickup in store items are excluded from this logic since they are not delivery related. No delivery item is created for them and they are not taken into account in delivery cost distribution.
  - Although the costs are sent in the item, the tax results are kept as TaxValues in header rather than in the item (AbstractOrderModel.totalTaxValues). A prefix is added to the tax code (Ex: ForItem-0-) to differentiate the delivery cost tax values resulting from different items. The summarization options valid throughout the system also apply for the delivery cost taxes.

Header delivery cost method is the default method in delivery cost tax calculation and should be preferred in most cases. On the other hand, the distributed delivery cost method allows more accurate tax calculation but comes with its own complexity with increased number of TaxValue's in items.

These are the expected differences in requests on REST and SOAP methods:

For REST:

- **Header Cost Option:** For each tax request a charges section is added to the header

  Example:
```
"charges": [
    {
        "type": "Delivery Cost",
        "amount": 9.99,
        "productCode": "FREIGHT"
    }
],
```

- **Distributed delivery cost per item option:** A separate charge section is added for each delivery related item in cart/order, an additional item is inserted in the request.

  Example from a request JSON:
```
"lines": [
{
        "discountAmount": 0.0,
        "lineNumber": "0",
        "partNumber": "300067375",
        "productCode": "0012",
        "quantities": [
            {
                "amount": 1
            }
```

```
            ],
            "charges": [
                {
                    "type": "Delivery Cost",
                    "amount": 5.36,
                    "productCode": "FREIGHT"
                }
            ],
            "unitOfMeasure": "pieces",
            "amount": 60.0
        }
    ]
```


For SOAP:

This can happen in two different ways:

- **Header Cost Option:** For each tax request an additional item is added to the order that represents the delivery cost.
    - The delivery cost for the cart/order is used as the amount for this item.
    - Quantity will always be 1.
- **Distributed delivery cost per item option:** For each delivery related item in cart/order, an additional item is inserted in the request.
    - The distributed delivery cost is assigned as the item value.
    - The delivery tax items are associated with their original items with a <RELATED_LINE_NUMBER> tag so that the same tax calculation logic is applied both to the cart/order item and its respective delivery.

Summarization Options for Tax Authority Details

The ONESOURCE Determination response for a typical tax call includes details of all the Tax Authorities (Tax Blocks) associated with a line. Keeping this level of detail in SAP Commerce documents may result with having several tax value lines for each item in the cart/order and in the header (in case delivery cost exists). For this reason, it is possible to apply a summarization operation to these tax values and group them by using one of the available criteria, hence reducing the number of tax value lines in carts/orders in SAP Commerce.

“thomsonreuters.idt.response.summarization” system property has to be maintained to activate summarization. There are three different possible options for summarization:

- SummaryByErpCode:
- SummaryByAuthorityAndZone
- FullDetails (No Summarization)

For REST Connections:

The summarization is performed by the Determination. The summarization parameter is sent as a part of the request for an input the Determination summarization. The response already includes the summarized tax values. These results are directly mapped to SAP Commerce tax values and persisted on the document header / items.

For SOAP Connections:

The summarization operation is performed in SAP Commerce. The Determination sends all tax details in the response with no grouping. Depending on the “thomsonreuters.idt.response.summarization” system property, summarization may be applied to group the results while converting the results to SAP Commerce tax values. The results are persisted just like the REST method.

Buyer and Seller Registrations

For installations supporting Canada and European Union, it is necessary to maintain and send buyer and seller registration numbers in tax calls.

- Seller Registration Numbers: “thomsonreuters.idt.registration.seller” system configuration has to be set as true. Multiple registration numbers can be defined for a base store in Backoffice. For this reason, a custom field is introduced to the Base Store model (tridtTaxRegistrationNumbers).
- Buyer Registration Numbers: “thomsonreuters.idt.registration.buyer” system configuration has to be set as true. Multiple registration numbers can be defined for a B2B Unit in Backoffice. For this reason, a custom field is introduced to the B2BUnit model (tridtTaxRegistrationNumbers).

For REST Connections:

Registrations section is added on the header. Ex:

```
"registrations": [
    {
        "type": "SellerRole",
        "registrationNumber": [
            " GB123456718",
            " DE655894849",
        ]
    }
],
```

For SOAP Connections

REGISTRATIONS tag is added to the header:

```
<ns2:REGISTRATIONS>
    <ns2:SELLER_ROLE>GB123456718</ns2:SELLER_ROLE>
    <ns2:SELLER_ROLE>DE655894849</ns2:SELLER_ROLE>
</ns2:REGISTRATIONS>
```

Sending INCOTERMS

If b2btrtaxintegration extension is active, it is possible to send INCOTERMS in a tax request. There are two prerequisites for this feature:

- “thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms” configuration item should be set as true.
- A new field is created on DeliveryMode model (tridtIncoterms). By using this field, an INCOTERM value can be assigned to each delivery mode. The value for this field cannot be longer than 10 characters. If this field is empty, first 10 characters of the DeliveryMode.code field is used.

For REST Connections:

If the prerequisites are met, “deliveryTerm” section is added to the request JSON header. Ex:

```
"deliveryTerm": {
    "type": "Incoterms",
    "term": "CIF"
},
```

For SOAP Connections

REGISTRATIONS tag is added to the header:

```
<ns2:DELIVERY_TERMS>CIF</ns2:DELIVERY_TERMS>
```

Audit Calls in Orders

You are provided with two options for the timing of the ONESOURCE Determination tax audit call in the order process.

- **Auditing After Shipment:** This is the default setting which is also supported by the SAP’s out-of-the-box order business process (order-process). In this approach, no audit tax call is made right after the order is saved on the database (after a successful checkout) and the audit tax call is postponed after the warehouse process is complete. When no open consignment remains for an order and at least one consignment is in SHIPPED status, the “Tax Posting" business process step is activated and a request is prepared just like the ones already sent in the checkout process, but this time with the following differences:
    - For REST requests, a CommitRequest is sent instead of a CalculateRequest.
    - For SOAP requests, <IS_AUDITED> tag is sent as TRUE.
    - Always the last version of the order is taken as reference when preparing the request. This means if there are any partial cancellations happened in the order lifecycle, updated quantities and amounts are sent to the Determination Engine. For this reason, the calculated tax value in the audit call may differ from the estimation calls already happened in the checkout.
    - To be consistent with the estimation calculations in checkout, the document date value (<INVOICE_DATE> for SOAP, documentDate for REST), which is the basis for the tax calculation date, is set as the order date (AbstractOrderModel.date field), not the actual posting date.
    - To have this feature activated, you should keep “thomsonreuters.idt.request.audit.auditafterdelivery” system configuration as true which is the default value.
- **Auditing After Order Placement:** If your business and accounting practices requires making an audited tax call immediately after order, you can achieve this feature by setting

"thomsonreuters.idt.request.audit.auditafterdelivery" system configuration as false.

- Although the postTaxes action after successful completion of "verifyOrderCompletion" step is not necessary in this scenario, you can keep the process as is, since the code will decide to trigger a request based on the "thomsonreuters.idt.request.audit.auditafterdelivery" value.

Order Cancellations

- In the standard SAP order business process, order cancellations do not trigger a new tax call. In the standard flow, since the cancellations happen before completion of the warehouse process, tax posting is done with the updated quantities and amounts for partial cancellations and not done at all for full cancellations.
- If you would like to update your order document after a cancellation or if you decide to go with the audit after order option in tax auditing, you will need to customize your system and call the external tax service during the cancellation process.
- There are at least two possible options for such a customization:
  - **Implementing OrderCancelPaymentServiceAdapter interface** in basecommerce extension. As of SAP Commerce version 2105, this interface is not implemented by SAP and expected to be implemented by each client according to their own business logic.
    - This implementation was not provided as a part of our product since payment processing is beyond our focus of interest.
    - In this class, A simple call to the ExternalTaxesService interface (calculateExternalTaxes method) would be enough to trigger a tax call and update the order document.
    - This method supports both immediate and warehouse cancellations.
  - **Extending the business process** to add extra steps after partial and full cancellations.
    - In the standard order process definition XML (order-process), look for actions that have "transition name='CANCELLED'" or wait nodes where "choice id='cancelled'" and make sure the flow is directed to the new steps after those transitions.
    - The bean necessary for defining such a step is already provided in the package. Use bean refreshTaxesAfterCancellationAction (class TridtOrderCancelTaxRefreshAction) in your process definition to enable tax refresh calls inside your business process step.

Recalculation of Cart/Order Totals

In the lifecycle of an order, recalculation of the totals, including total tax, might be performed several times starting with the checkout process, until the completion of the order process. Typical events that trigger a recalculation might be adding a product to cart, changing quantities, applying promotions or a partial cancellation that results a refund action. Technically speaking, DefaultCalculationService is the default class which is responsible for executing the business logic for this action. In this business logic, there are two important factors that need to be considered from tax point of view:

- The underlying assumption in this class is that the taxes are determined internally from TaxRows and for tax calculation. For this reason, it doesn't call the ExternalTaxService interface which is responsible to trigger ONESOURCE Determination integration. In checkout process this problem is handled by calling the external tax service immediately after recalculation and overriding the total tax value afterwards. But in other parts of the system, for example the "Recalculate Orders" button in Orders page of Backoffice UI, this additional external tax call functionality is missing.

- In each recalculation attempt, cart/order header tax values (AbstractOrderModel.totalTaxValuesInternal and totalTaxValues fields) are reconstructed by gathering all item tax values and finding subtotals of each tax code. This logic contradicts with SAP's external tax logic where totalTaxValuesInternal field is used to keep delivery related tax values (Check DefaultApplyExternalTaxesStrategy interface for details). For this reason, if a recalculation happens with calling external tax service, the header tax values in cart/order will be corrupted.

To cope with these challenges, following functionality is provided as part of this integration package:

- A custom implementation for CalculationService interface, TridtCalculationService, is provided that deactivates the unwanted effect of accumulation of item tax values in the header. Please keep in mind that, the service will not be active immediately since spring bean definitions are missing. This is intentional, since CalculationService is a core platform functionality which might already be enhanced. For this reason, the logic in TridtCalculationService should be manually merged to your implementation.

- Two custom backoffice actions are provided in trtaxintegrationbackoffice extension that replace system standard recalculation buttons in Orders page. Please check "Backoffice Enhancements" section for details.

In your custom enhancements, if you must call CalculationService interface, please consider calling ExternalTaxesService right after to update tax information for the new version. The interface will decide to make an estimation or an audit call depending on the status of the order and update Determination records if necessary. OrderCancelPaymentServiceAdapter is a good example for such a custom implementation, where clients usually apply custom business logic to handle the refund process after a cancellation and might require order recalculation.

Returns Process

To enable the tax calculation calls in the return process in SAP Commerce, return business process definition has to be customized.

- An example definition (tridt-return-process) is already provided in this package. You can find the definition in an impex format in trtaxintegration extension under resources / business_processes folder in tridt-return-business-process.impex file.

- This impex will not be automatically uploaded during system update / initialization and should be manually imported if necessary. Please note that this definition is not intended to be directly used in production environments and were provided just as an example. You should adapt this definition to your own return business process keeping in mind the following points:

    - There is an extra process step called taxCalculateAction, which is responsible to make an estimation tax call for an approved return request. The estimation results are kept in the Return

Request document for reference.

  - Compared to original process definition, tridt-return-process executes the tax posting (taxReverseAction step) before the payment refund process but right after the goods acceptance. This way the calculated tax values can be used when refunding the return in the payment processing step. The tax call at this step is an audit call which allows new records to be created in ONESOURCE Determination.

- If you created a new business process for this purpose, don't forget to update your Base Store settings in Backoffice to activate the new definition (BaseStoreModel. createReturnProcessCode field).

The data model for Return Requests were extended with custom fields in order to be able to hold detailed information just like carts and orders. Here are the custom fields in ReturnRequestModel:

- tridtTaxValuesInternal: Just like the counterpart in AbstractOrderModel (totalTaxValuesInternal), this field is not visible in Backoffice UI and it keeps the string representations of TaxValue objects assigned to ReturnRequest header. These tax values are the tax calculated for the delivery cost.

- tridtTaxValues: Just like the counterpart in AbstractOrderModel (totalTaxValues), this is a dynamic field that shows the tridtTaxValuesInternal as a Collection in Backoffice UI.

- tridtTaxPostingDate: If the value of this field is used as the base date for tax calculation. The value of this field is determined according to "thomsonreuters.idt.returns.usegoodsacceptancedate" system configuration property. If this property is true, the field is set at the time when taxReverseAction step is executed which corresponds to the goods issue processing date. If property is false, the field is kept empty and the associated order's date (AbstractOrderModel.date) is used as the base date for tax calculation.

Here are the custom fields in RefundEntryModel:

- tridtTaxValuesInternal: Just like the counterpart in AbstractOrderEntryModel (taxValuesInternal), this field is not visible in Backoffice UI and it keeps the string representations of TaxValue objects assigned to ReturnRequest header. These tax values are the tax calculated for the delivery cost.

- tridtTaxValues: Just like the counterpart in AbstractOrderEntryModel (taxValues), this is a dynamic field that shows the tridtTaxValuesInternal as a Collection in Backoffice UI.

Summarization options that are valid for the order's base store is also valid for the return request document. Please check "Summarization Options for Tax Authority Details" section for details.

SAP standard refund amount calculation logic does not include return request tax values into the total refund amount (class CaptureRefundAction). This implementation is not included in this package since the payment domain is out of this product's scope. It is advised that the clients who would like to use this functionality, extend refund calculation logic to take taxes into account when calculating the totals.

Refunding the Delivery Cost in Returns

As per standard SAP Commerce logic, it is possible to refund the delivery cost in a return (Refund Delivery Cost field on Return Request). In this case, the tax calculation needs to be carried out for the delivery cost. In this process:

- Whatever delivery cost taxing method is chosen for carts and orders (Header charges vs item charges), the same method applies for the return request documents. Please check “Taxes for the Delivery Cost” section for details.

- In case of a partial return, items that are not being returned are also kept in the tax request but their quantities are set to zero. If delivery cost is being refunded and item distribution for delivery cost strategy is activated, these zero quantity items may be referred by delivery cost items in SOAP requests or carry charges in REST requests. This is necessary to calculate the exact delivery cost tax amount in a return request just like the cart/order.

Backoffice Enhancements

The enhancement in Backoffice UI within the ONESOURCE Determination Integration package is listed as follows:

Backoffice Administrator Role

- “Recalculate order totals” and “Calculate with promotions” action buttons in “Orders” page are deactivated if external taxing is active in the base store of the order.

- Two new buttons are introduced instead of the two actions above: “Recalculate order totals with external tax” and “Calculate with promotions with external tax”. These actions will call Determination and update tax values in the order and respective order entries after executing the recalculation logic.

Customer Support Agent Role

- The following actions existing on the Order Details page are deactivated as they are not relevant for this integration.
    - Manual Tax Void Action
    - Manual Tax Commit Action
    - Manual Tax Requote Action
    - Manual Delivery Cost Commit Action
- The following actions are added on the Order Details page by the ONESOURCE Determination integration package.
    - Refresh Taxes Action (It the order is audit relevant, the request will be audited in Determination, otherwise it will be an estimation call. All Tax Values in the order and respective order entries will be refreshed and the total tax field in the order header will be recalculated.)
- Following actions existing on the Return Request Details page are deactivated as they are not relevant for this integration.
    - Manual Tax Reverse Action
- The following actions are added on the Return Request page by the ONESOURCE Determination integration package.

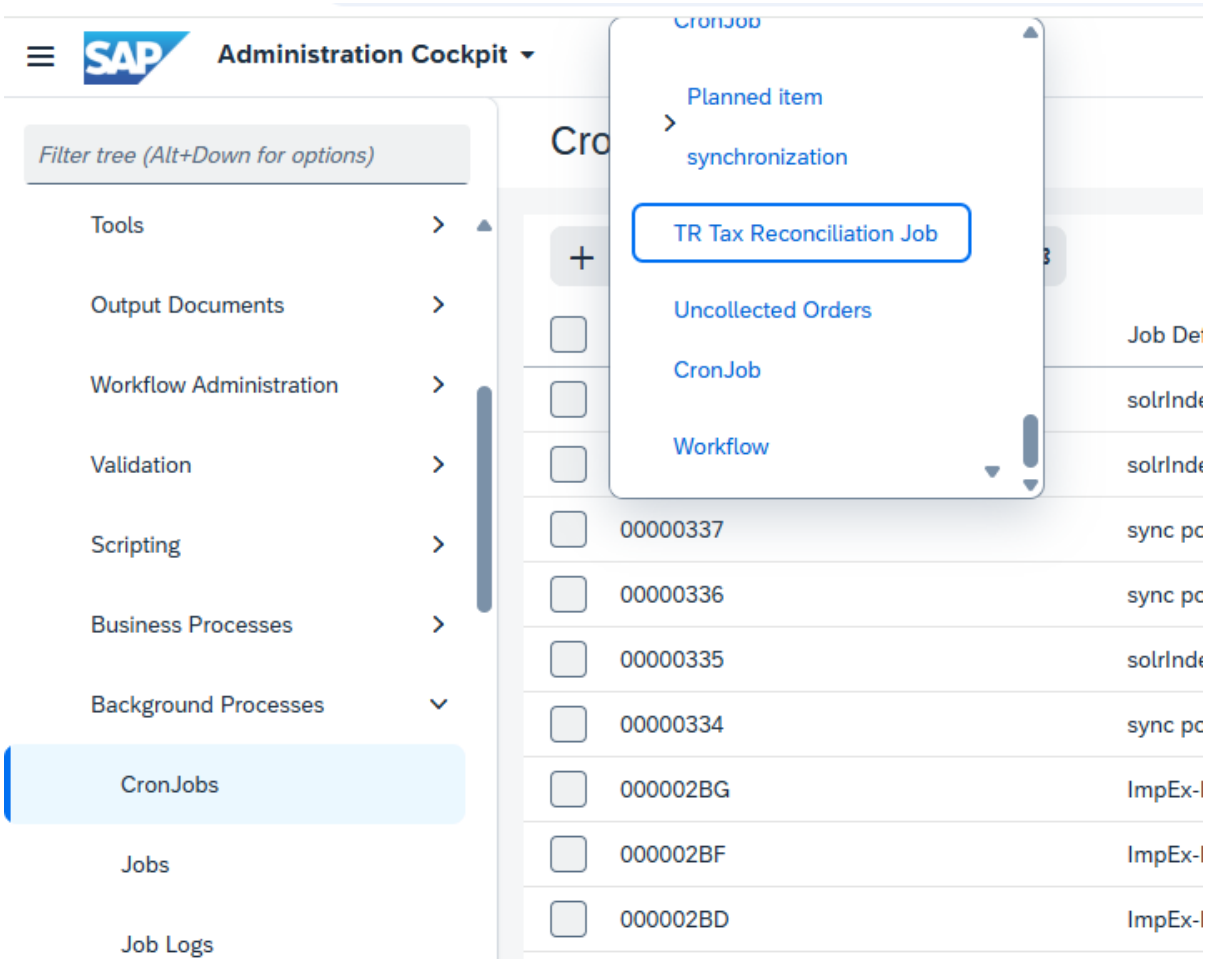
- Refresh Taxes Action (The system will send an audit call to the Determination System. Tax fields in the return request and refund entries will be recalculated.)

Reconciliation Report

The integration package provides a reconciliation report that can be used in month-end closing. By running a cronjob, users can download a list of audited orders and returns in a specified period for a specified base store and compare the results with ONESOURCE Determination reports to identify inconsistencies between two systems.

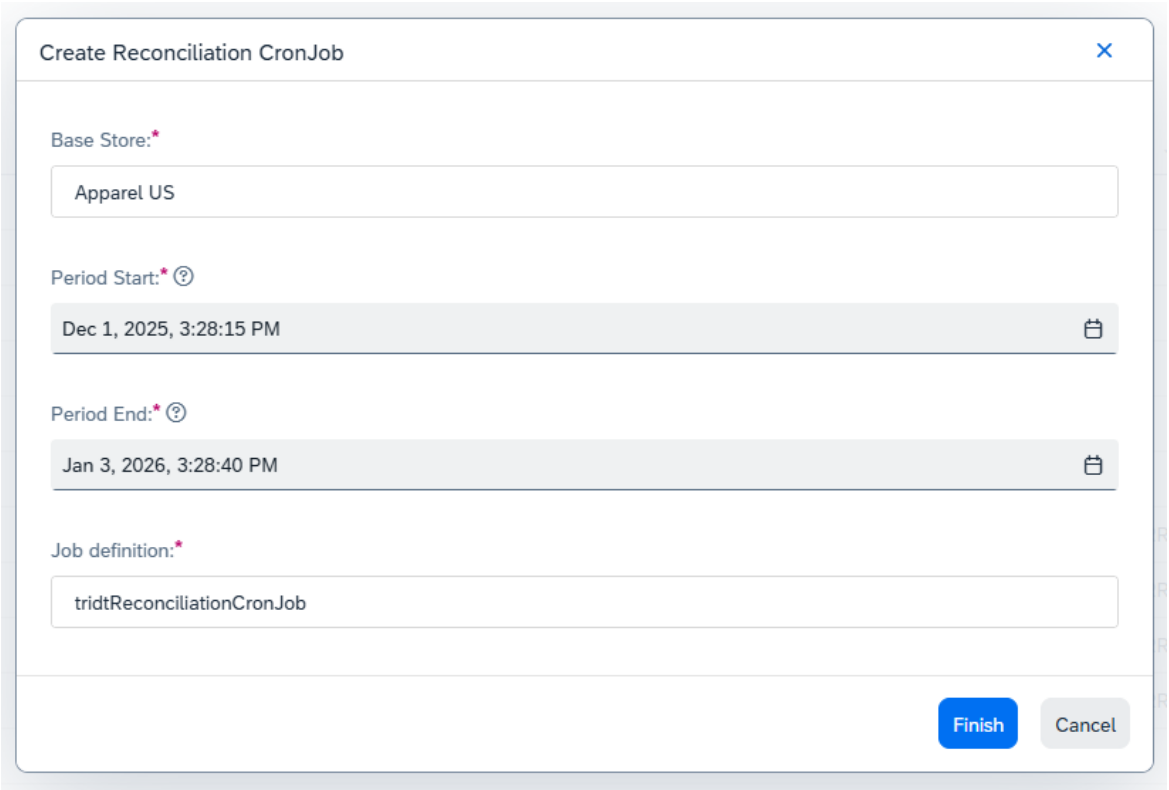
To run the cronjob:

- In Backoffice UI, go to System/ Background Processes / Cronjobs
- Create a new cronjob by opening up the sub-types of Cronjob and select TR Tax Reconciliation Job.

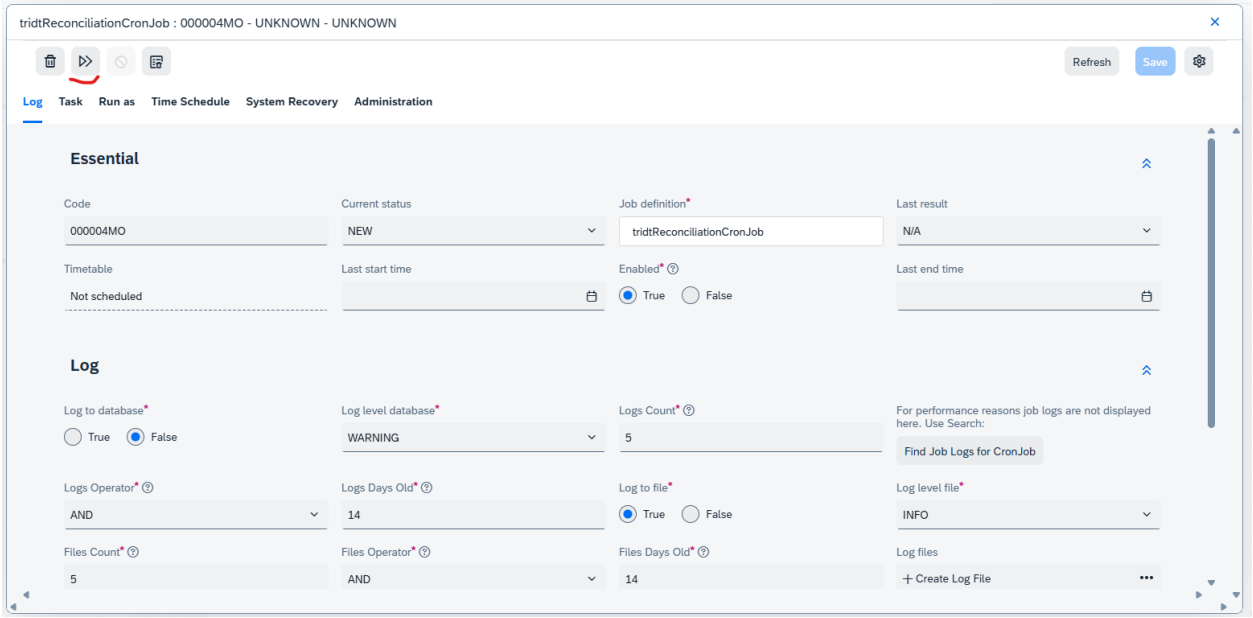


- Select a base store, period start and end dates for the interval. Then choose tridtReconciliationCronJob for job definition.
    - When selecting dates, please keep in mind that, the report will only consider the date portion of the selected value. The times will be adjusted to the start of the day for the period start and end of the day for the period end after the cronjob is started.

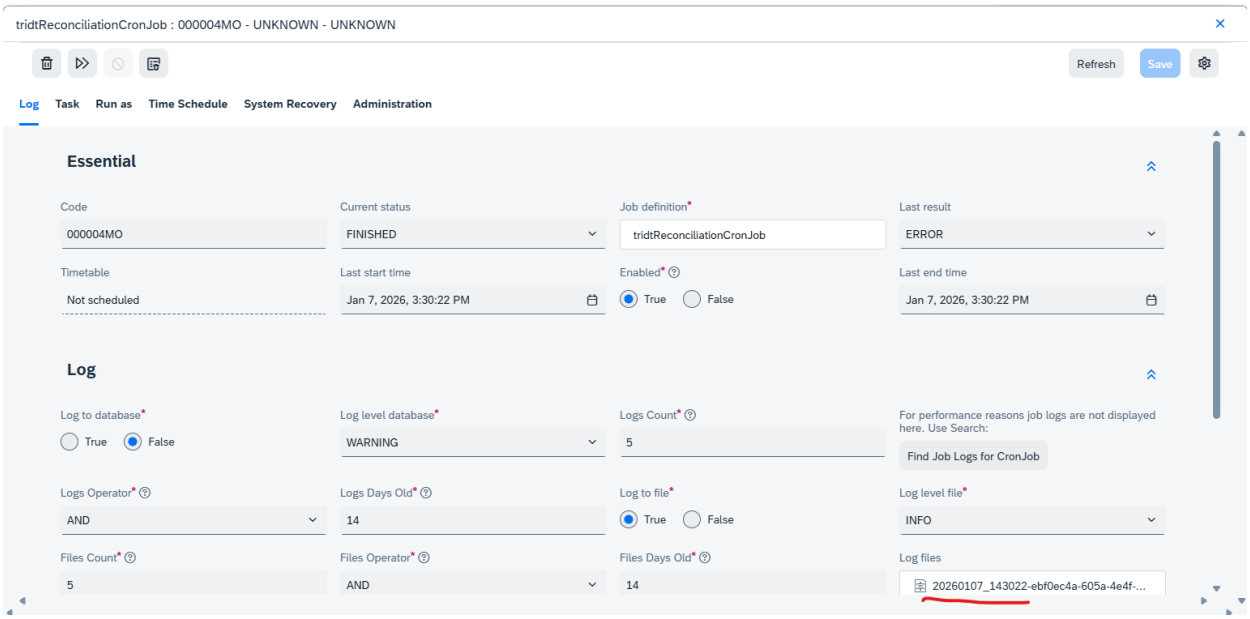
- Since the dates are kept in the system timezone in SAP Commerce, the date chosen by the user is assumed to be in the system timezone.
  Ex: For a user in EST (browser timezone) on a system in CST, Oct 1, 2020 10:49:16 AM EST is adjusted as Oct 1, 2020 12:00:00 AM CST for a period start date.



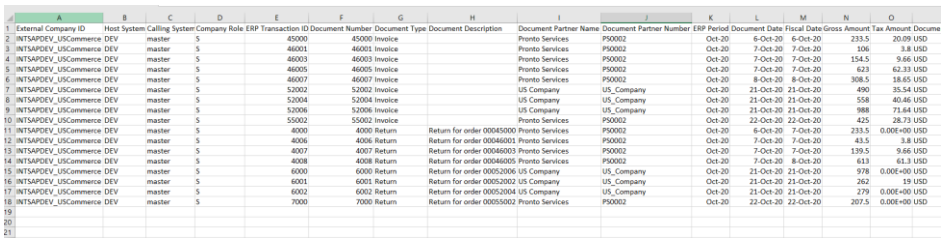
- After clicking on Done, a new cronjob will be created but not run yet. In order to run the job, find the newly created cronjob and start it.



- If the job is completed successfully, you'll find a csv file in Resulting CSV field under Administration tab



- You can double click on the file to download it to your local environment.



External Company ID	Host System	Calling System	Company Role	ERP Transaction ID	Document Number	Document Type	Document Description	Document Partner Name	Document Partner Number	ERP Period	Document Date	Fiscal Date	Gross Amount	Tax Amount	Docume
INTSAPDEV_USCommerce	DEV	master	S	45000	45000	Invoice		Pronto Services	PS0002	Oct-20	6-Oct-20	6-Oct-20	233.5	20.09	USD
INTSAPDEV_USCommerce	DEV	master	S	46001	46001	Invoice		Pronto Services	PS0002	Oct-20	7-Oct-20	7-Oct-20	106	3.8	USD
INTSAPDEV_USCommerce	DEV	master	S	46003	46003	Invoice		Pronto Services	PS0002	Oct-20	7-Oct-20	7-Oct-20	154.5	9.66	USD
INTSAPDEV_USCommerce	DEV	master	S	46005	46005	Invoice		Pronto Services	PS0002	Oct-20	7-Oct-20	7-Oct-20	623	62.33	USD
INTSAPDEV_USCommerce	DEV	master	S	46007	46007	Invoice		Pronto Services	PS0002	Oct-20	8-Oct-20	8-Oct-20	308.5	18.65	USD
INTSAPDEV_USCommerce	DEV	master	S	52002	52002	Invoice		US Company	US_Company	Oct-20	21-Oct-20	21-Oct-20	490	35.54	USD
INTSAPDEV_USCommerce	DEV	master	S	52004	52004	Invoice		US Company	US_Company	Oct-20	21-Oct-20	21-Oct-20	558	40.46	USD
INTSAPDEV_USCommerce	DEV	master	S	52006	52006	Invoice		US Company	US_Company	Oct-20	21-Oct-20	21-Oct-20	988	71.64	USD
INTSAPDEV_USCommerce	DEV	master	S	55002	55002	Invoice		Pronto Services	PS0002	Oct-20	22-Oct-20	22-Oct-20	425	28.73	USD
INTSAPDEV_USCommerce	DEV	master	S	4000	4000	Return	Return for order 00045000	Pronto Services	PS0002	Oct-20	6-Oct-20	7-Oct-20	233.5	0.00E+00	USD
INTSAPDEV_USCommerce	DEV	master	S	4006	4006	Return	Return for order 00046001	Pronto Services	PS0002	Oct-20	7-Oct-20	7-Oct-20	43.5	3.8	USD
INTSAPDEV_USCommerce	DEV	master	S	4007	4007	Return	Return for order 00046003	Pronto Services	PS0002	Oct-20	7-Oct-20	7-Oct-20	139.5	9.66	USD
INTSAPDEV_USCommerce	DEV	master	S	4008	4008	Return	Return for order 00046005	Pronto Services	PS0002	Oct-20	7-Oct-20	8-Oct-20	613	61.3	USD
INTSAPDEV_USCommerce	DEV	master	S	6000	6000	Return	Return for order 00052006	US Company	US_Company	Oct-20	21-Oct-20	21-Oct-20	978	0.00E+00	USD
INTSAPDEV_USCommerce	DEV	master	S	6001	6001	Return	Return for order 00052002	US Company	US_Company	Oct-20	21-Oct-20	21-Oct-20	262	19	USD
INTSAPDEV_USCommerce	DEV	master	S	6002	6002	Return	Return for order 00052004	US Company	US_Company	Oct-20	21-Oct-20	21-Oct-20	279	0.00E+00	USD
INTSAPDEV_USCommerce	DEV	master	S	7000	7000	Return	Return for order 00055002	Pronto Services	PS0002	Oct-20	22-Oct-20	22-Oct-20	207.5	0.00E+00	USD

- If the job can not find any orders or returns for the given parameters, the result of the cronjob will be set as ERROR.

CONFIGURATION OPTIONS

It is possible to change the default integration behavior by providing certain system properties in local.properties file for local installations and manifest.json file SAP Public Cloud Systems. Each option is described below in detail:

CONFIGURATION KEY	DESCRIPTION	CAN BE BASE STORE SPECIFIC ?
Connection - Common		
thomsonreuters.idt.externalcompanyid	External Company ID value used in your ONESOURCE Determination setup Default value: Empty	Yes
thomsonreuters.idt.hostsystem	String that represents the role of the system. Ex: UAT, PROD etc. Default value: UAT	No
thomsonreuters.idt.callingsystemnumber	String that represents the calling system sending the transaction. Can be differentiated in case of multiple tenants in a single system Default value: 01	No
Connection - REST		
thomsonreuters.idt.rest.active	When true, the connection method for integration will be REST When false, the connection method for integration will be SOAP Default value: Empty	No
thomsonreuters.idt.rest.url	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.idt.rest.clientid	Client ID If this property is set together with clientsecret, Client ID and Secret method is used for authenticating REST calls. Check Setting up Authentication section	No

	for details Default value: Empty	
thomsonreuters.idt.rest.clientsecret	If this property is set together with clientid, Client ID and Secret method is used for authenticating REST calls. Check Setting up Authentication section for details Default value: Empty	No
thomsonreuters.idt.rest.scope	Optional property that only needs to be set when connecting to EMEA or OCI Determination. Please contact Thomson Reuters for details Default value: urn:tr:onesource:auth:api:IndirectTaxDetermination which is for AMERS AWS. Use urn:tr:onesource:auth:api:IndirectTaxDeterminationII for AWS EMEA and urn:tr:onesource:auth:api:IndirectTaxDeterminationIV for OCI	No
thomsonreuters.idt.connection.retry.delay	Optional property for waiting duration between retry attempts in milliseconds Default value: 1000	No
thomsonreuters.idt.connection.retry.attempts	Optional property for number of retries in case of a connection error Default value: 3	No
Connection - SOAP		
thomsonreuters.idt.soap.url	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.idt.soap.username	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.idt.soap.password	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.idt.soap.connection.connectiontimeout	Timeout settings for SOAP connections	No

	Default value: 5000	
thomsonreuters.idt.soap.connection.readtimeout	Timeout settings for SOAP connections Default value: 5000	No
Logging		
thomsonreuters.idt.consolelogging.request	If set to true, the details of the request JSON will be outputted to the system console. Default value: false	No
thomsonreuters.idt.consolelogging.response	If set to true, the details of the response JSON will be outputted to the system console. Default value: false	No
Address		
thomsonreuters.idt.address.sendGeoCode	If it is true, geocode will be included in the request. The value that will be sent as geocode will depend on thomsonreuters.idt.address.postalcodeIncludesGeoCode and thomsonreuters.idt.address.geocode parameters. Default value: false	Yes
thomsonreuters.idt.address.postalcodeIncludesGeoCode	If it is true, XXXXX-YYYY format is expected on Address.postalCode field: where the first part will be used for the postal code field and the second part will be used for the geocode field in the request . If no hyphen is detected, the entire string will be used for the postal code and geocode will be left empty. If it is false Address.postalCode field is used for the postal code field. In this case, if thomsonreuters.idt.address.sendGeoCode is true, thomsonreuters.idt.address.geocode will be used to generate the geocode value. Default value: false	Yes
thomsonreuters.idt.address.geocode	Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property Default value: Empty	Yes

thomsonreuters.idt.address.county	Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property Default value: district	Yes
thomsonreuters.idt.address.city	Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property Default value: town	Yes
thomsonreuters.idt.address.district	Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property Default value: district	Yes
thomsonreuters.idt.request.address.sendorderacceptanceaddress	If set to true, address details will be determined based on the tridtOrderAcceptanceAddress field on the Base Store. If the field is empty, the ship-from address will be used as the order acceptance address. Default value: false	Yes
thomsonreuters.idt.request.address.sendoderoriginaddress	In B2C, if set to true, bill-to address will be copied to the order-origin address. In case bill-to is not set at the time of the tax call, ship-to address will be used instead. In B2B, if set to true, address details will be populated based on the B2BUnit.contactAddress field. If the field is empty, similar to B2C, bill-to or ship-to address will be copied to order origin address. Default value: false	Yes
thomsonreuters.idt.request.address.sendsellerprimary	If set to true, address details will be determined based on the tridtOrderAcceptanceAddress field on the Base Store. If the field is empty, the ship-from address will be used as the seller primary address. Default value: false	Yes
thomsonreuters.idt.request.address.sendbuyerprimary	In B2C, if set to true, bill-to address will be copied to the buyer primary address. In case bill-to is not set at the time of the tax call, ship-to address will be used instead	Yes

	In B2B, if set to true, address details will be determined based on the B2BUnit.contactAddress field. If the field is empty, similar to B2C, bill-to or ship-to address will be copied to buyer primary address. Default value: false	
Product Code / Commodity Code Determination		
thomsonreuters.idt.request.item.sendCommodityCode	If set to true, commodity code will be sent for each line item. Product code field will be sent otherwise. Default value: false	Yes
thomsonreuters.idt.request.item.taxcode.enhancedsearch.producthierarchy.enabled	If set to true, an enhanced search will be triggered to find product/commodity code starting from the product in cart and going upper levels if not found. E.g. First check the size variant's product master for a product/commodity code, if not found check the associated color variant and if not found check the base model. Default value: false	Yes
thomsonreuters.idt.request.item.taxcode.enhancedsearch.categoryhierarchy.enabled	If set to true and if the search in product hierarchy doesn't provide any results, then an enhanced search will be triggered to find product/commodity code starting from the first level category and going upper levels if not found. E.g. First check the immediate categories assigned to the product in cart, if not found check the parent categories until a value is found. Default value: false	Yes
thomsonreuters.idt.request.item.taxcode.fallbackCode	If no value is found for product/commodity code, this text will be used as a fallback code. Default value: Empty	Yes
Delivery Cost		
thomsonreuters.idt.request.shipping.productcode	This value will be used as the default product/commodity code for the delivery cost item. Default value: FREIGHT	Yes
thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms	If set to true, an incoterm value will be added to the header and items (in case there is a specific delivery	Yes

	mode in the item). Please check Sending INCOTERMS section for details. Default value: false Only valid for B2B scenario	
thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems	If activated, header delivery cost value will be distributed to the items and a separate charges section will be included in the item details of the request. Please check Taxes for the Delivery Cost section of this document Default value: false	Yes
thomsonreuters.idt.response.shipping.prefix	This prefix will be used for naming every delivery cost related tax value if item charges scenario is activated. Please visit Taxes for the Delivery Cost section for details Default value: ForItem-	Yes
Tax Auditing		
thomsonreuters.idt.request.audit.auditafterdelivery	If set to true, tax audit call order delivery status should be SHIPPED as a prerequisite to trigger a tax audit call. If set to false, a tax audit call will be performed as soon as the cart is converted to an order. Default value: true	Yes
Canada and EU specific		
thomsonreuters.idt.registration.seller	If activated, any registration number maintained in the base store will be included in the request as seller registration numbers. Default value: false	Yes
thomsonreuters.idt.registration.buyer	If activated, any registration number maintained in the B2BUnit will be included in the request as buyer registration numbers. Default value: false Only valid for B2B scenario	Yes

Tax Response Summarization		
thomsonreuters.idt.response.summarization	The value should be either of these three values: - SummaryByErpCode - SummaryByAuthorityAndZone - FullDetails Details are described in section Summarization Options for Tax Authority Details Default value: SummaryByAuthorityAndZone	Yes
Returns		
thomsonreuters.idt.returns.usegoodsacceptancedate	If set to true, goods acceptance date is sent as the base date for return request tax calculation. If set to false, the date field for the associated order is used instead. Default value: false	Yes
Service Call Fallback		
thomsonreuters.idt.fallback.fallbackoncart	If activated, for cases where ONESOURCE Determination service is not reachable or integration call could not be completed due to a system error, a fallback rate will be used as an alternative to avoid an exception in checkout. Default value: false	Yes
thomsonreuters.idt.fallback.fixedpercentage	The fixed-rate for the fallback process. Default value: Empty	Yes

BASE STORE SPECIFIC CONFIGURATION

The configuration properties defined in the table above can be used directly to adjust the behavior system-wide. In case you need to define different behavior for each Base Store in your organization, you can add the base store code as a postfix and override the generic value. Note that not all configuration items support this feature. Please refer to the “Can be Base Store specific?” column in the table above for each item.

Example:

- Suppose you have two base stores running in parallel: storeA and storeB and you have the following items in your local.properties file:
    - thomsonreuters.idt.registration.seller=false
    - thomsonreuters.idt.registration.seller.storeA=true
- In this case, you should expect the following behavior:
    - Seller registrations should be sent for storeA but not for storeB.

DEVELOPMENT AND EXTENSIBILITY

CUSTOM ATTRIBUTES

ONESOURCE Determination enables companies to submit, and have returned, user-defined elements that are not directly supported in Determination. These elements can be used in conjunction with TransEditors (input filters) to enable custom tax calculations and are stored in the audit table for reporting purposes. Each user element structure contains a single name/value pair.

If you need to use these fields and send in the request payload, you need to extend the capabilities of trtaxintegration extension. Certain steps that need to be taken depends on the connection method that is active on the system. Here is a general outline for the enhancement steps:

1) Decide on the hierarchy. Customer attributes can be sent either in:
    a) Header
    b) Product item
    c) Delivery cost items (Only valid for SOAP connections)
2) In a custom extension add trtaxintegration as a required extension
3) In the custom extension, create a populator class implementing de.hybris.platform.converters.Populator with the following source and target parameters:
    a) For header attributes:
        1. source: de.hybris.platform.core.model.order.AbstractOrderModel
        2. target: com.thomsonreuters.idt.client.model.CalculateRequest for REST or taxcalculationservice.wsdl.IndataType for SOAP
    b) For product item attributes:
        1. source: com.thomsonreuters.idt.integrations.sapcommerce.models.TridtDiscountedOrderEntry
        2. target: com.thomsonreuters.idt.client.model.CalculateLineRequest for REST or taxcalculationservice.wsdl.IndataLineType for SOAP
    c) For delivery cost item attributes: (Only valid for SOAP)
        1. source: de.hybris.platform.core.model.order.AbstractOrderModel
        2. target: taxcalculationservice.wsdl.IndataLineType
4) Implement the business logic for each class. Add name/value pairs to the respective structure, in com.thomsonreuters.idt.client.model.UserAttribute type for REST or taxcalculationservice.wsdl.UserElementType for SOAP. Currently restricted to the values ATTRIBUTE2 through ATTRIBUTE50. Note that ATTRIBUTE1 is already used in the standard logic in trtaxintegration extension (Only for SOAP).

5) Manage dependency injection in {{extension}}-spring.xml file

a) Add a new bean for the populator. Ex:

```
<alias name="{{populator bean ID}}" alias="{{populator bean ID}}" />
<bean id="{{populator bean ID}}" class="{{Populator class path}}"/>
```

b) Add the populator bean to the converter bean as a dependency

For header attributes:

For REST:

```
<alias name="{{header converter bean ID}}" alias="tridtOrderConverter" />
<bean id="{{header converter bean ID}}" parent="abstractPopulatingConverter">
   <property name="targetClass" value="
com.thomsonreuters.idt.client.model.CalculateRequest" />
   <property name="populators">
      <list>
         <ref bean="tridtCommonOrderHeaderPopulator" />
         <ref bean="tridtB2COrderHeaderPopulator" />
         <!-- Optional for B2B -->
          <!-- <ref bean="tridtB2BOrderHeaderPopulator" /> -->
         <ref bean="{{populator bean ID}}" />
      </list>
   </property>
</bean>
```

For SOAP:

```
<alias name="{{header converter bean ID}}" alias="tridtOrderConverter" />
<bean id="{{header converter bean ID}}" parent="abstractPopulatingConverter">
   <property name="targetClass" value="taxcalculationservice.wsdl.IndataType" />
   <property name="populators">
      <list>
         <ref bean="tridtCommonOrderHeaderPopulator" />
         <ref bean="tridtB2COrderHeaderPopulator" />
         <!-- Optional for B2B -->
          <!-- <ref bean="tridtB2BOrderHeaderPopulator" /> -->
         <ref bean="{{populator bean ID}}" />
      </list>
   </property>
</bean>
```

For item attributes:

For REST:

```
<alias name="{{item converter bean ID}}" alias="tridtOrderItemConverter" />
<bean id="{{item converter bean ID}}" parent="abstractPopulatingConverter">
   <property name="targetClass" value="
com.thomsonreuters.idt.client.model.CalculateLineRequest" />
   <property name="populators">
      <list>
         <ref bean="tridtOrderItemPopulator" />
         <ref bean="{{populator bean ID}}" />
       </list>
    </property>
</bean>
```

For SOAP:

```
<alias name="{{item converter bean ID}}" alias="tridtOrderItemConverter" />
<bean id="{{item converter bean ID}}" parent="abstractPopulatingConverter">
   <property name="targetClass" value="taxcalculationservice.wsdl.IndataLineType" />
   <property name="populators">
      <list>
         <ref bean="tridtOrderItemPopulator" />
         <ref bean="{{populator bean ID}}" />
       </list>
    </property>
</bean>
```

For delivery cost item attributes (Only valid for SOAP):

```
<alias name="{{delivery cost item converter bean ID}}"
alias="tridtShippingItemConverter" />
<bean id="{{delivery cost item converter bean ID}}"
parent="abstractPopulatingConverter">
  <property name="targetClass" value="java.util.ArrayList" />
  <property name="populators">
    <list>
       <ref bean="tridtShippingItemPopulator" />
       <ref bean="{{populator bean ID}}" />
    </list>
   </property>
</bean>
```

SUPPORT FOR SPARTACUS UI AND OMNI COMMERCE CONNECT (OCC) REST API

The SAP Commerce ONESOURCE integration (estimation calls during checkout) can also be triggered by Omni Commerce Connect (OCC) web service calls. When an update on the cart is received via an OCC call, the system decides whether a new tax call is necessary and performs the call automatically.

Here are some examples that can trigger an external tax call:

- POST - /{baseSiteId}/users/{userId}/carts/{cartId}/addresses/delivery
- PUT - /{baseSiteId}/users/{userId}/carts/{cartId}/deliverymode
- POST/PUT - /{baseSiteId}/users/{userId}/carts/{cartId}/paymentdetails

This feature makes ONESOURCE Determination integration available for the new Spartacus UI.

In addition to that following custom endpoints are added to the available OCC endpoints if you include trtaxintegrationocc extension in your setup. These endpoints provide extra tax summary information and can be used in case you decide to customize your checkout, order details and return details pages to provide details about your calculated tax, such as tax value details.

- GET - /{baseSiteId}/{userId}/tax/cart/{cartId}
- GET - /{baseSiteId}/{userId}/tax/order/{orderId}
- GET - /{baseSiteId}/{userId}/tax/returnRequest/{returnRequestId}

LOGGING AND MONITORING

CONSOLE LOGGING

In the lifetime of a single tax call, several log messages are generated by the system and those can be displayed in the standard application log in SAP Commerce. The table below summarizes the logs that should be observed in the logs for an error-free tax call process. Please note that the messages are slightly varies according to the connection method active in the system.

STEP NO	LOG LEVEL	CLASS	TEXT
1	INFO	TridtTaxCalculationSOAPService or TridtTaxCalculationRESTService	Starting ONESOURCE Determination process...
2	DEBUG	TridtSoapLoggingService for SOAP TridtClientHttpRequestInterceptor for REST	ONESOURCE Determination Tax Request (XML/JSON) :- {1} : {2} {1}: Only valid for REST messages, either CALCULATE or COMMIT depending on the type of the request {2}: Request message Note: The message is only activated if thomsonreuters.idt.consolelogging.request=true and DEBUG log level is activated for classes TridtSoapLoggingService or TridtClientHttpRequestInterceptor
3	DEBUG	TridtSoapLoggingService for SOAP TridtClientHttpRequestInterceptor for REST	ONESOURCE Determination Tax Response (XML/JSON)- {1} : {2} {1}: Only valid for REST messages, either CALCULATE or COMMIT depending on the type of the request {2}: Response message Note: The message is only activated if thomsonreuters.idt.consolelogging.request=true and DEBUG log level is activated for classes TridtSoapLoggingService or TridtClientHttpRequestInterceptor

4	INFO	TridtTaxCalculationSOAPService or TridtTaxCalculationRESTService	Request stats for ID: {}, Total Tax Processing Duration: #{}# ms {1}: Unique request ID that includes the cart id and the timestamp {2}: Total time passed for the tax processing

KIBANA DASHBOARDS

In SAP Public Cloud Deployments these logs can be displayed in Kibana user interface. As a part of the integration package, two custom dashboards with several metrics and saved searches are provided for monitoring OneSource Determination integration.

Installation Instructions:

- In your trtaxintegration extension, you will find a folder that contains necessary files to import custom dashboards located in resources\kibana path
- Import dashboards, visualizations and saved searches
    - Go to Kibana/ Management / Saved Objects and click import
    - Choose “Tridt-Kibana-Dashboards.ndjson” located in resources\kibana folder and execute
- Start using your dashboards and saved searches in your SAP Commerce Public Cloud environment.

The dashboards, visualizations and saved searches are based on the logs described in the table above. They will not work properly, if any of these log items are deactivated by setting a lower log level in log4j2.

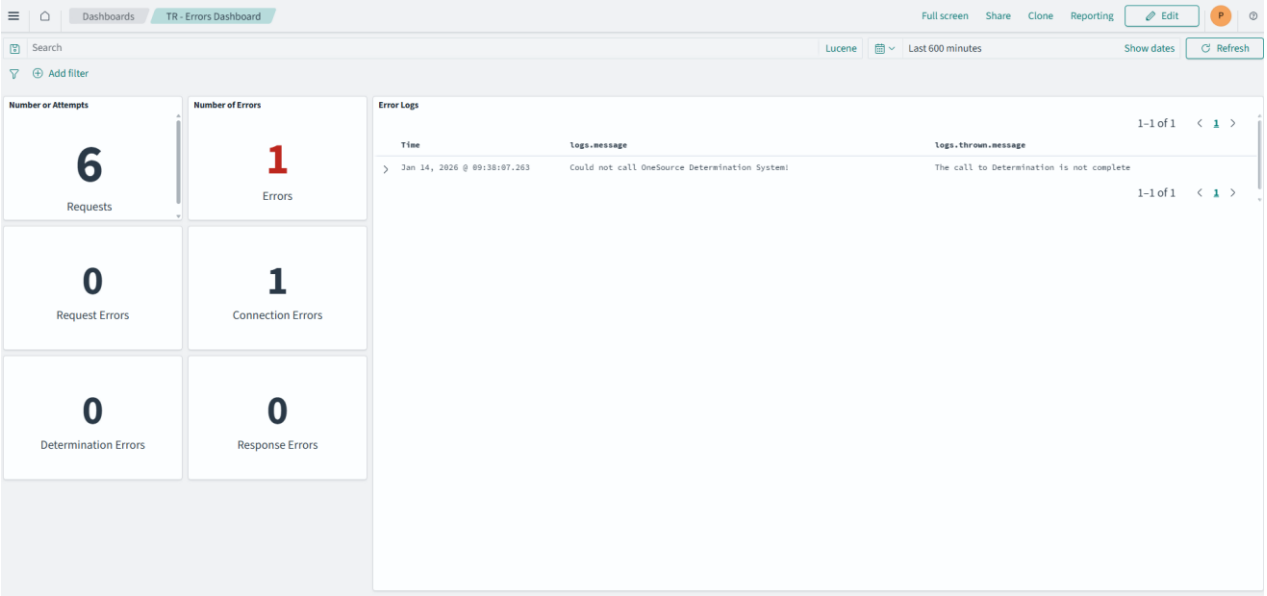
Errors Dashboard

The dashboard includes several visualizations that focus on tax integration-related errors and their breakdowns in a specified time frame.

The time frame can be set in the upper right section. The dashboard includes the following information:

- The number of tax call attempts
- The total number of errors
- The total number of request preparation related errors.
- The total number of connection errors.
- The total number of errors returned by ONESOURCE Determination.

- The total number of response processing related errors.
- List of all errors

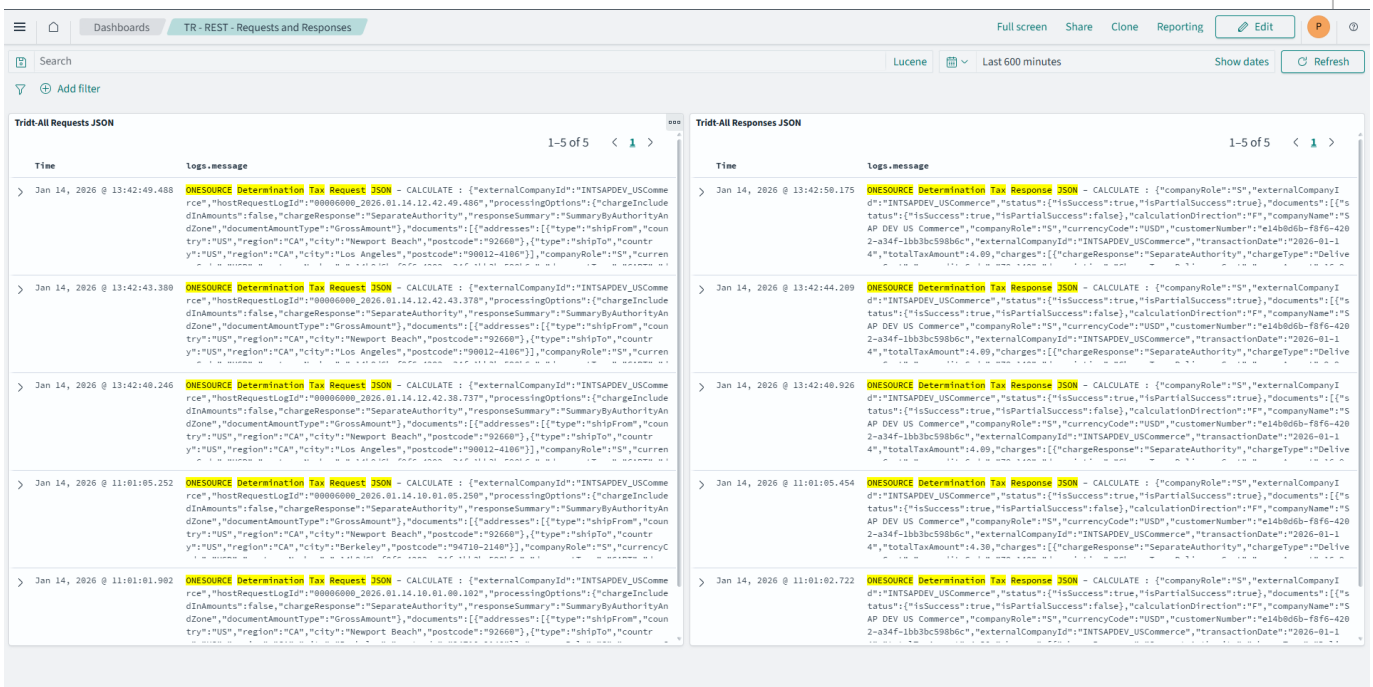


Requests and Responses Dashboards

There are two dashboards (one for XML's and the other for JSON's) that includes two saved searches for request and responses in a specified time frame. Please note that detailed logging should be activated for having messages to be written to the system log.

The time frame can be set in the upper right section.

Details of a single message can be displayed by expanding the details and clicking on the "View Single Document" button.



You can copy a log ID (Ex: 00007000_2020.04.29.15.24.39), go to the Discover page of Kibana and execute a search to display all tax integration relevant logs for a specific call. On the same page, you can also use wildcards to display all logs from multiple tax calls for a single cart (Ex: 00007000_*).

AVS Dashboard

This dashboard summarizes various information regarding the Address Verification Service integration, including the number of verification attempts and errors together with request and response logs in a given timeframe.

Dashboards | TR - AVS Dashboard | Full screen | Share | Clone | Reporting | Edit

Search | Lucene | Last 600 minutes | Show dates | Refresh

Add filter

Number of Attempts

7

Count

Number of Errors

1

Count

Errors

1–1 of 1

Time	logs.message
Jan 14, 2026 @ 13:36:05.594	Address Verification Service call failed and not completed

1–1 of 1

AVS Requests

1–6 of 6

Time	logs.message
Jan 14, 2026 @ 13:42:30.890	ONESOURCE Determination AVS Request (REST) - {"item":{"address1":"111 Main st ","address2":"","city":"Los Angeles","region":"CA","postalCode":"90001","country":"US"}}
Jan 14, 2026 @ 11:32:16.825	ONESOURCE Determination AVS Request (REST) - {"item":{"address1":"111 Main st","address2":"","city":"Los Angeles","region":"CA","postalCode":"90001","country":"US"}}
Jan 14, 2026 @ 11:09:55.893	ONESOURCE Determination AVS Request (REST) - {"item":{"address1":"111 Main st","address2":"","city":"Los Angeles","region":"CA","postalCode":"90001","country":"US"}}
Jan 14, 2026 @ 11:02:31.930	ONESOURCE Determination AVS Request (SOAP)- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP-ENV:Header> <wsse:Security SOAP-ENV:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.or

AVS Responses

1–6 of 6

Time	logs.message
Jan 14, 2026 @ 13:42:31.160	ONESOURCE Determination AVS Response (REST) - {"status":"CORRECTED","messages":[{"description":"Directional or Suffix change,ZIP code change","type":"INFO"}],"items":[{"address1":"111 N Main St","address2":"","city":"Los Angeles","region":"CA","isResidential":"false","postalCode":"90012-4106","country":"United States of America","county":"Los Angeles","countryISO2":"US","countryISO3":"USA"},{"address1":"111 S Main St","address2":"","city":"Los Angeles","region":"CA","isResidential":"false","postalCode":"90012-3701","country":"United States of Americ
Jan 14, 2026 @ 11:32:17.014	ONESOURCE Determination AVS Response (REST) - {"status":"CORRECTED","messages":[{"description":"Directional or Suffix change,ZIP code change","type":"INFO"}],"items":[{"address1":"111 N Main St","address2":"","city":"Los Angeles","region":"CA","isResidential":"false","postalCode":"90012-4106","country":"United States of America","county":"Los Angeles","countryISO2":"US","countryISO3":"USA"},{"address1":"111 S Main St","address2":"","city":"Los Angeles","region":"CA","isResidential":"false","postalCode":"90012-3701","country":"United States of Americ
Jan 14, 2026 @ 11:09:56.223	ONESOURCE Determination AVS Response (REST) - {"status":"CORRECTED","messages":[{"description":"Directional or Suffix change,ZIP code change","type":"INFO"}],"items":[{"address1":"111 N

TROUBLESHOOTING SCENARIOS

CLASSIFICATION OF HIGHER-LEVEL EXCEPTIONS

A typical tax call is composed of four different sub-processes and they run one after the other.

Request Preparation → Establishing a connection to Determination → Determination Engine's internal process → Response Processing

In case of an error, you can make a search logs with the following keywords to understand which part of the process is causing the error.

Once you determine the problematic area, you should look for child exceptions/errors to determine the root cause.

EXCEPTION	TEXT	REASON
TridtRequestException	Request Preparation	SAP Commerce
TridtConnectionException	Establishing a connection to Determination	SAP Commerce / Determination
TridtDeterminationException	Determination Engine's internal process	Determination
TridtResponseException	Response Processing	SAP Commerce
DefaultExternalTaxesService.saveOrder		
TaxValue.parseTaxValue		

REQUEST PREPARATION EXCEPTIONS

You'll find the following exception in your logs: **TridtRequestException** – "Could not prepare the request for ONESOURCE Determination! Order/Cart/Return Request {1} cannot be converted to a request" where {1} is the document number.

This is a generic exception which indicates that the real problem is in request preparation. You need to investigate logs for further details. Here are some examples for possible errors and their reasons:

EXCEPTION	TEXT	REASON
ConversionException	No External Company Id found for Thomson Reuters Tax integration	thomsonreuters.idt.externalcompanyid system property is missing
IllegalArgumentException	The default Point of Service	BaseStore.defaultDeliveryOrigin field is

	assigned to the Base Store cannot be null.	empty.
IllegalArgumentException	No Ship-from address is assigned to the base store	BaseStore.defaultDeliveryOrigin.address field is empty.
NoSuchMethodException	Dynamic address field fetch failed!	Check thomsonreuters.idt.address.* properties. The value does not correspond to a field in AddressModel
IllegalArgumentException	Field XXX is not a String field!	Check thomsonreuters.idt.address.* properties. The value should be a String typed field in AddressModel

CONNECTION EXCEPTIONS

You'll find the following exception in your logs: **TridtConnectionException** – Could not call ONESOURCE Determination System!

This is a generic exception which indicates that the real problem occurred while trying to connect to Determination. You need to investigate logs for further details. Here are some possible errors and their reasons:

This kind of errors mostly occur when the REST/SOAP Request is malformed or contains invalid authentication credentials.

Here are some examples:

EXCEPTION	TEXT	REASON
HttpClientErrorException	400 Missing the required parameter 'headerParams' with Correlation-Id, Authorization	Check your authentication details, (See section Setting up Authentication of this document)
UnknownHostException	{{url}}	Check your URL string in system configuration
SSLHandshakeException	Remote host terminated the handshake	Network problem. Tax server could not be reached
EOFException	SSL peer shut down incorrectly	Network problem. Tax server could not be reached
NullPointerException	Cannot invoke "org.bouncycastle.util.io.pem.PemObject.getContent()" because "pemObject" is null	Check your private key file in REST connections

IOException	Tax API - Token could not be refreshed	Assertion token could not be generated. Check your authentication details, (See section Setting up Authentication of this document)
OAuth2Exception	OAuth Error	OAuth authentication has failed, check your authentication details
OAuth2AccessDeniedException	Error requesting access token.	OAuth authentication has failed, check your authentication details
SocketException	Connection timed out (Read failed)	Determination failed to send any response. This indicates an internal problem in Determination System. Note: The connection process is backed up with a retry mechanism that resends the message up to a certain number of attempts. Please check thomsonreuters.idt.connection.retry.attempts and thomsonreuters.idt.connection.retry.delay configuration parameters to tweak this behaviour.
WebServiceTransportException	404	Determination system could not be found. Check thomsonreuters.idt.soap.url
SoapFaultClientException	A security error was encountered when verifying the message	Username and password can be wrong. Check thomsonreuters.idt.soap.username and thomsonreuters.idt.soap.password system properties
WebServiceIOException	I/O error: Permission denied: connect: {domain}; nested exception is “java.net.SocketException”: Permission denied: connect: {domain} where {domain} is the URL domain of Determination Service	Network communication error during integration, check network settings are correct
WebServiceIOException	I/O error: Connection reset; nested exception is “javax.net.ssl.SSLException”: Connection reset	Network communication error during integration, check network settings are correct

DETERMINATION ERRORS

You'll find one of the following exceptions in your logs:

TridtDeterminationException – "ONESOURCE Tax Determination returned an error! Determination error for id {1}: {2} " where {1} is the cart number and {2} is the error text returned by the Determination.
This is a generic exception which indicates that the Determination failed to process the request and returned a response with a status "Not Success". You need to investigate logs for further details.

Here are some examples:

TEXT	REASON
COMPANY_NOT_FOUND	External Company ID cannot be found
EUSG67	Seller must register for intra-community supplies Authority: {{EU Country}}

RESPONSE PROCESSING EXCEPTIONS

You'll find one the following exceptions in your logs:

- **TridtResponseException** – "ONESOURCE Tax Determination response could not be processed! Response processing failed for {1}" where {1} is the document number. This exception indicates that response XML/JSON could not be converted into SAP's internal TaxValue structure

- Any exception in **de.hybris.platform.commerceservices.externaltax.impl.DefaultExternalTaxesService.saveOrder** in dicates that converted TaxValue's could not be saved into the database.

- Any exception in **de.hybris.platform.util.TaxValue.parseTaxValue** indicates that converted TaxValue's could not be saved into the database.

Those are generic exceptions. You need to investigate logs for further details. Here are some possible errors and their reasons:

EXCEPTION	TEXT	REASON
ConversionException	Tax response is corrupted, conversion not possible	The response XML/JSON received, does not have a valid document inside.
NullPointerException	Cannot invoke "java.util.List.isEmpty()" because "lines" is null	The response XML/JSON received, does not have a valid line inside.
NullPointerException	Cannot invoke	There is no tax value info for a line in the

	"java.util.List.stream()" because "taxList" is null	response XML/JSON.
DataIntegrityViolationException	ModelSavingException query; SQL []; String or binary data would be truncated.; nested exception is com.microsoft.sqlserver .jdbc.SQLServerException: String or binary data would be truncated.	Database table field length for CartEntries.taxValueInternal field is 255 characters long by design. This field is too short to keep all tax values for certain cases. SAP suggests changing the field length by running an "Alter Table" SQL query. In SAP Public Cloud deployments, customers can not perform this operation. You should open an incident to SAP Support, with component CEC-HCS-CCAZ-DBO, and ask SAP to change the length of field p_taxvaluesinternal in table cartentries and orders to 1000 characters.
NumberFormatException	error parsing tax value string '<TV<XX-XXX #[ATT]sales[ATT]#4.07#true#4.07# USD>VT>' : java.lang.NumberFormatException: For input string: "[ATT]sales[ATT]"	One of the "tax code" for an item possible includes a special character that s misinterpreted by the SAP Commerce internal logic. Check the response XML/JSON for details.

OTHER PROBLEMS

Sometimes there are no exceptions visible in logs, but the tax values are supposed to have a different value. For those cases most of the time you need to focus on Request Preparation and you need to make sure that all relevant fields are included in the message and the values in the request XML/JSON are correct.

Here are some examples:

AREA	SYPTOM	SOLUTION
Address	Geocode field is missing in the request XML/JSON	thomsonreuters.idt.address.postalcodeIncludesGeoCode should be set to false and thomsonreuters.idt.address.geocode should point to a String typed field in AddressModel

Address	Format of postal code is wrong	If thomsonreuters.idt.address.postalcodeIncludesGeoCode is true, postalCode field in AddressModel is expected to be in XXXXX-YYYY format where X's are for postal code and Y's are for Geocode
Address	County, city or district fields are missing in request XML/JSON	Check properties starting with thomsonreuters.idt.address.county (or city or district). The value should point to a String typed field in AddressModel.
Address	No bill-to address in the request for a B2B customer	The source of Bill-To address depends on how cart is paid. If it is a credit card payment, the bill-to address is searched in the PaymentInfo associated with the cart. If it can't be found, bill-to address is assumed to be the same with the ship-to address and no separate address is added to the tax request. If it is an account payment, the B2BUnit associated with the B2BCustomer is found and B2BUnit.billingAddress field is used to populate address fields. Please check there is no problem in those fields.
Header	Customer number is wrong for a B2B customer	The B2BUnit associated with the B2BCustomer is found and name and uid field is used for customer name and id. Please check there is no problem in those fields.
Response	Wrong summarization level or no summarization carried out at all.	Check thomsonreuters.idt.response.summarization property. Value may be wrong.

SYSTEM BEHAVIOR ON INTEGRATION FAILURES

Common Behavior

If there is a connection error, the request will be resent after waiting 1000 ms up to 3 times if necessary.

These parameters can be changed in system configuration (thomsonreuters.idt.connection.retry.delay and thomsonreuters.idt.connection.retry.attempts)

Specific Cases (After Unsuccessful Trials)

Cart

If any error occurs in the integration in checkout while the user is still interacting with the system, two options are possible:

- If thomsonreuters.idt.fallback.fallbackoncart is true, thomsonreuters.idt.fallback.fixedpercentage value is applied to all items and the delivery cost as a fixed tax rate
- If thomsonreuters.idt.fallback.fallbackoncart is false, an exception is thrown. In this case, it is client's responsibility to catch this exception and convert it to a meaningful and appropriate warning message on the storefront.

Order Process

An exception is thrown, and the process fails. After examining the logs and eliminating the error, the order process should be resumed to redo the tax posting.

Backoffice

An exception is thrown, and the operation fails. An error is displayed on the screen. After examining the logs and eliminating the error, the Refresh Tax action should be retriggered.

Return Request

An exception is thrown, and the process fails. After examining the logs and eliminating the error, the return process should be resumed to redo the tax posting.

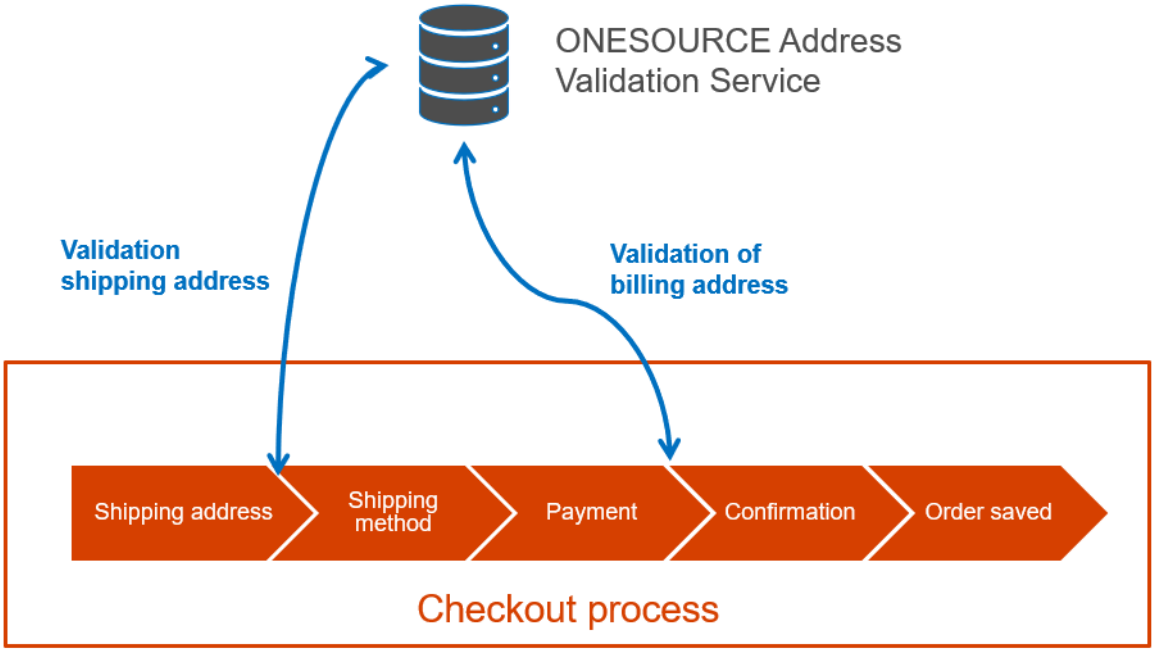
ADDRESS VALIDATION SERVICE (AVS)

For triggering a successful address validation call to ONESOURCE AVS, the following prerequisites should be met:

- AVS Calls should be activated in system configuration (thomsonreuters.avs.integration.active=true)
- External Company ID for AVS should be defined in system configuration.
- Authentication details must be defined for the chosen connection method (REST or SOAP)
- The user enters an address in the storefront either in checkout or in My Account / Address Book.

AVS BUSINESS PROCESS DESCRIPTION

Checkout



During a checkout process both in B2C and B2B storefronts, two address validation requests can be sent to the Address Validation Service, to validate the address details provided by the user. In this process, the user fills the address form to create a new address in the system and clicks “Next” to proceed to the next step. At this stage, AVS service is called with the form fields. The service can return three different results:

- **Address confirmed:** In this case, the address is saved on the cart (and optionally on the address book of the user) and the next checkout step is displayed afterward.

- **Address corrected:** AVS returns a similar address with some fields corrected. The new address is displayed as a popup to the user. The user can then accept the new address or continue with the old version.

- **Address rejected:** When the address can not be identified by the service, an error is returned by the

system together with a reason. The user stays on the address step to make the necessary corrections and the error and the reason are displayed as a notification.

In the checkout step, if the user selects an address from the address book (previously used and stored addresses) the integration will not be triggered.

My Account / Address Book

In the accelerators of SAP Commerce, users can edit their stored addresses in Address Book page under My Account section. When adding a new address or editing an existing one, the same AVS call mentioned above is triggered and the results are processed in a similar way.

AVS SERVICE CONNECTION METHODS – SOAP VS REST

Similar to the tax service, AVS supports two different connection methods, SOAP and REST. The major difference between these two methods is the countries supported. Please refer to the table below for supported countries for each connection.

CONNECTION TYPE	US	CANADA	OTHER COUNTRIES
SOAP	Supported	Not Supported	Not Supported
REST	Supported	Supported	Supported

Clients can choose either of the method depending on their requirements. Maintain "thomsonreuters.avs.rest.active" system property to set the active connection method for AVS integration. If this property is set to true, REST will be active, otherwise it will be a SOAP call. For the authentication details, refer to the Setting up the Authentication section for the tax integration. AVS integration has a different set of configuration items but the main logic is the same as tax integration.

AVS CONFIGURATION OPTIONS

It is possible to change the default integration behavior by providing certain system properties in local.properties file for local installations and manifest.json file SAP Public Cloud Systems. Each option is described below in detail:

CONFIGURATION KEY	DESCRIPTION	CAN BE BASE STORE SPECIFIC ?
Connection - Common		
thomsonreuters.avs.integration.active	If true, integration will be active. Default value: true	Yes
thomsonreuters.avs.externalcompanyid	External Company ID value used in your ONESOURCE Determination setup Default value: Empty	No
Connection - REST		
thomsonreuters.avs.rest.active	If true, REST integration will be active, SOAP otherwise. Default value: true	Yes
thomsonreuters.avs.rest.url	REST end-point for AVS	No
thomsonreuters.avs.rest.clientid	Client ID If this property is set together with clientsecret, Client ID and Secret method is used for authenticating REST calls. Check Setting up Authentication section for details Default value: Empty	No
thomsonreuters.avs.rest.clientsecret	If this property is set together with clientid, Client ID and Secret method is used for authenticating REST calls. Check Setting up Authentication section for details Default value: Empty	No
thomsonreuters.avs.rest.scope	Optional property that only needs to be set when connecting to EMEA or OCI Determination. Please contact Thomson Reuters for details Default value: urn:tr:onesource:auth:api:IndirectTaxDetermination which is for AMERS AWS. Use	No

	urn:tr:onesource:auth:api:IndirectTaxDeterminationII for AWS EMEA and urn:tr:onesource:auth:api:IndirectTaxDeterminationIV for OCI	
thomsonreuters.avs.connection.retry.delay	Waiting duration between retry attempts in milliseconds. Default value: 1000	No
thomsonreuters.avs.connection.retry.attempts	Number of retries in case of a connection error Default value: 3	No
Connection - SOAP		
thomsonreuters.avs.soap.username	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.avs.soap.password	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.avs.soap.url	The value will be provided by Thomson Reuters Default value: Empty	No
thomsonreuters.avs.connection.readtimeout	Timeout settings for SOAP connections Default value: 5000	No
thomsonreuters.avs.connection.connectiontimeout	Timeout settings for SOAP connections Default value: 5000	No
Logging		
thomsonreuters.avs.consolelogging.request	If true, the request message will be logged. Default value: false	No
thomsonreuters.avs.consolelogging.response	If true, the response message will be logged. Default value: false	No
Others		
thomsonreuters.avs.integration.connection.errorallowed	If true, in case of connection or system errors, users will be allowed to proceed to the next checkout step.	Yes

	Default value: false	
thomsonreuters.avs.integration.connection.errormessage	Default value: Address Verification Service is down.	Yes

AVS UI MESSAGES

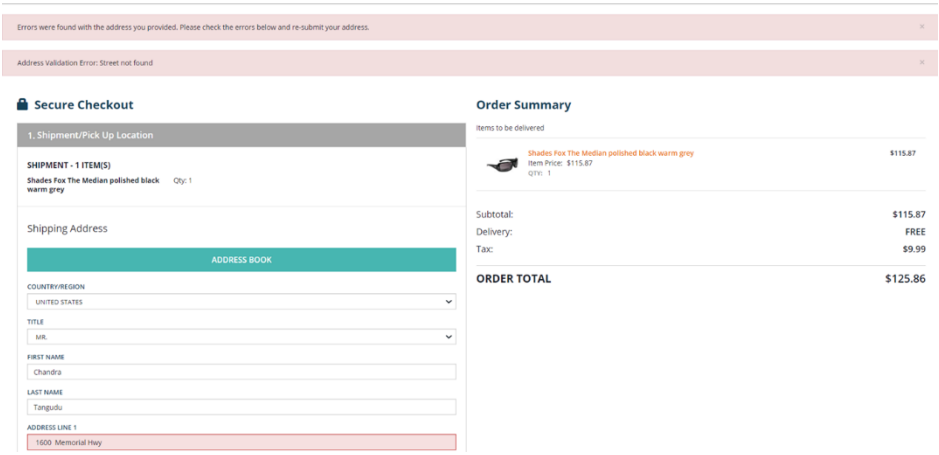
In case of an error, ONESOURCE Address Validation Service returns an explanation text that describes the root cause. Here are some possible error messages that can be returned by the service:

- Street number or box number out of range
- Please enter a valid address number.
- Street not found.
- Please input a street address.
- Address not found.
- Multiple addresses match.

To display this text in UI to enable the user to make the necessary corrections, certain steps have to be carried out depending on your storefront UI technology.

Accelerator Based UI

travsintegrationaddon extension must be added to your setup. Please refer to section “Architecture and Design Overview” for further details. After the addon is active you should see an extra message when AVS returns an error.



Spartacus UI

trtaxintegrationocc extension provides an enhanced version of SAP's address verification OCC endpoint (POST - /{baseSiteId}/users/{userId}/addresses/verification) which can be access on (POST - /{baseSiteId}/users/{userId}/addresses/travs). This endpoint's request and response data structure is the same with the original endpoint but it returns the AVS error in the reason field of the response. Here is a sample response from this endpoint:

```
{
    "decision": "REJECT",
    "errors": {
        "errors": [
            {
                "message": "Invalid field: addressline1",
                "reason": "Street not found",
                "subject": "line1",
                "subjectType": "parameter",
                "type": "ValidationError"
            }
        ]
    }
}
```

In your Spartacus project you can replace OCC endpoint for address verification with this enhanced version and customize your page to display the reason field as an error message.

The customizations on Spartacus UI are beyond the scope of this integration package.

AVS LOGGING AND MONITORING

In the lifetime of a single address validation call, several log messages are generated by the system and those can be displayed in the standard application log in SAP Commerce. The table below summarizes the logs that should be observed in the logs for an error-free address validation call process.

STEP NO	LOG LEVEL	CLASS	TEXT
1	INFO	TravsAddressVerificationService	Calling Addres Verification Service…
2	INFO	TravsSoapLoggingService or TravsClientHttpRequestInterceptor	ONESOURCE Determination AVS Request (REST/SOAP) - {1} {1}: Request XML The message is only activated if thomsonreuters.avs.detailedlogging.request =true and DEBUG log level is activated for class

			TravsSoapLoggingService or TravsClientHttpRequestInterceptor
3	INFO	TravsSoapLoggingService or TravsClientHttpRequestInterceptor	ONESOURCE Determination AVS Response (REST/SOAP) - {1} {1}: Response XML The message is only activated if thomsonreuters.avs.detailedlogging.response =true and DEBUG log level is activated for class TravsSoapLoggingService or TravsClientHttpRequestInterceptor
4	INFO	TravsAddressVerificationService	Request stats for AddressLine1: {1}, Response Waiting Duration: ##{2}## {1}: Address string {2}: Time passed for the whole AVS process

In SAP Public Cloud Deployments these logs can be displayed in Kibana user interface.

If an exception occurs during the service call, the root cause will be logged together with this error message: **“Address Verification Service call failed and not completed”.**

For troubleshooting in Kibana, you can use this text to find the error logs and investigate the accompanying exception.

APPENDIX 1: TAX DATA MAPPING – REST

The table below describes the logic behind how SAP Commerce internal data structures are used to generate the Determination Request JSON and how Determination Response JSON is mapped back to SAP Commerce internal structures.

TAX REQUEST MESSAGE		
LEVEL	**FIELD NAME**	**SAP COMMERCE POPULATION LOGIC**
Header	externalCompanyId	Configuration: thomsonreuters.idt.externalcompanyid
Header	hostRequestLogId	For carts: AbstractOrder.code+Timestamp (Timestamp is in "yyyy.MM.dd.HH.mm.ss" format) For orders: AbstractOrder.code For returns: ReturnRequest.code
Header	processingOptions	Contains multiple fields, check Processing Option level
Header	callingSystemNumber	Only exist in a commit request Configuration: thomsonreuters.idt. callingsystemnumber
Header	hostSystem	Only exist in a commit request Configuration: thomsonreuters.idt. hostsystem
Header	documents	Collection of Document objects
Document	addresses	Collection of Address objects
Document	charges	Collection of Charge objects, only exists if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=false
Document	companyRole	Fixed value 'S'
Document	currencyCode	AbstractOrder.currency.isocode
Document	customerName	Only valid in B2B scenarios B2BUnit is found based on AbstractOrder.user and "B2BUnit.name" value is sent
Document	customerNumber	In B2C: AbstractOrder.user.customerID In B2B: B2BUnit is found based on AbstractOrder.user and B2BUnit.uid is sent
Document	deliveryTerm	Only valid in B2B scenarios.

		Will be used only if thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms =true
Document	documentType	One of these: CART, ORDER or RETURN REQUEST
Document	documentDate	For carts: Get the current date For Orders: Order.date For returns: Depends on "thomsonreuters.idt.returns.usegoodsacceptancedate". If config value is true ReturnRequest.tridtTaxPostingDate is used. In other cases ReturnRequest.order.date is used as date.
Document	isCredit	Only true for return requests
Document	registrations	Collection of Registration objects, only exists if thomsonreuters.idt.registration.seller or thomsonreuters.idt.registration.buyer is true.
Document	documentNumber	Only exist in a commit request For orders: AbstractOrder.code For returns: ReturnRequest.code Not used for carts
Document	originalDocumentNumber	Only exist in a commit request Associated order for the return request Only for return requests
Document	originalDocumentDate	The date for the associated order Only for return requests
Document	taxDeterminationDate	The date for the associated order Only for return requests
Document	lines	Collection of Line objects
Processing Option	chargeIncludedInAmounts	Fixed value 'false'
Processing Option	chargeResponse	Fixed value 'SeparateAuthority'
Processing Option	responseSummary	One of these values: SummaryByErpCode, SummaryByAuthorityAndZone, FullDetails
Processing Option	documentAmountType	Depends on “AbstractOrder.net” value. If true, it is 'GrossAmount', else it is 'GrossPlusTaxAmount'
Address	type	One of these values: 'shipTo', 'billTo', 'shipFrom', 'sellerPrimary', 'buyerPrimary', 'orderOrigin', 'orderAcceptance'

		Also check following system properties: thomsonreuters.idt.request.address.sendorderoriginaddress thomsonreuters.idt.request.address.sendorderacceptanceaddr ess thomsonreuters.idt.request.address.sendsellerprimary thomsonreuters.idt.request.address.sendbuyerprimary
Address	country	Address.country.isoCode
Address	region	Address.region.isoCodeShort
Address	county	Address.(field) field is determined based on thomsonreuters.idt.address.county
Address	city	Address.(field) field is determined based on thomsonreuters.idt.address.city
Address	district	Address.(field) field is determined based on thomsonreuters.idt.address.district
Address	postcode	Address.postalcode field value may be stripped based on thomsonreuters.idt.address.postalcodeIncludesGeoCode
Address	geocode	Address.postalcode or Address.(field) Please check configuration options for the following system properties: thomsonreuters.idt.address.postalcodeIncludesGeoCode thomsonreuters.idt.address.geocode
Charges	type	fixed value 'Delivery Cost'
Charges	amount	AbstractOrder.deliveryCost or a calculated value if thomsonreuters.idt.request.shipping.sendmultipledeliverycostit ems=true
Charges	productCode	Only used if thomsonreuters.idt.request.item.sendCommodityCode=false The value is fetch from thomsonreuters.idt.request.shipping.productcode
Charges	commodityCode	Only used if thomsonreuters.idt.request.item.sendCommodityCode=true The value is fetch from thomsonreuters.idt.request.shipping.productcode
Delivery Term	type	fixed value 'Incoterms'

Delivery Term	term	Either AbstractOrder.deliveryMode.tridtIncoterms or first 10 characters of AbstractOrder.deliveryMode.code
Registration	type	Either 'SellerRole' or 'BuyerRole'
Registration	registrationNumber	For seller registrations, the values are populated from BaseStore. tridtTaxRegistrationNumbers For buyer registrations, the values are populated from B2BUnit. tridtTaxRegistrationNumbers
Line	addresses	AbstractOrderEntry.deliveryPointOfService.address or AbstractOrderEntry.deliveryAddress Only if it is a pickup item or if the item delivery address is not empty
Line	amount	AbstractOrderEntry.TotalPrice + AbstractOrderEntry.DiscountValues
Line	charges	Collection of Charge objects, only exists if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=true
Line	deliveryTerm	AbstractOrderEntry.deliveryMode.code Only valid in B2B scenarios Will be used only if item delivery mode is not empty and thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms=true
Line	discountAmount	(Proportionally distributed AbstractOrder.DiscountValues) + (Sum of all AbstractOrderEntry.DiscountValues)
Line	lineNumber	AbstractOrderEntry.entryNumber
Line	partNumber	AbstractOrderEntry.product.code
Line	productCode	Only used if thomsonreuters.idt.request.item.sendCommodityCode=false For the value check the following configuration properties: thomsonreuters.idt.request.shipping.productcode thomsonreuters.idt.request.item.taxcode.enhancedsearch.productthierarchy.enabled thomsonreuters.idt.request.item.taxcode.enhancedsearch.categoryhierarchy.enabled thomsonreuters.idt.request.item.taxcode.fallbackCode
Line	commodityCode	Only used if thomsonreuters.idt.request.item.sendCommodityCode=true For the value check the following configuration properties: thomsonreuters.idt.request.shipping.productcode

		thomsonreuters.idt.request.item.taxcode.enhancedsearch.producthierarchy.enabled thomsonreuters.idt.request.item.taxcode.enhancedsearch.categoryhierarchy.enabled thomsonreuters.idt.request.item.taxcode.fallbackCode
Line	unitOfMeasure	AbstractOrderEntry.unit
Line	quantities	Wrapper for the Line Quantity level
Line Quantity	amount	AbstractOrderEntry.quantity

TAX RESPONSE MESSAGE (Only fields used in response processing are listed for simplification)		
LEVEL	**FIELD NAME**	**SAP COMMERCE MAPPING LOGIC**
Response Header	status	Hold status information
Status	isSuccess	Used to check if the calculation is successful
Response Document	documents	Collection of Document objects
Response Document	charges	Collection of Response Charge objects, only exists if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=false
Response Document	currencyCode	Used to populate currency in TaxValue
Response Document	lines	Collection of Response Line objects
Response Line	lineNumber	Line number
Response Line	charges	Collection of Response Charge objects, only exists if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=true
Response Line	taxes	Collection of Tax objects
Response Charge	taxes	Collection of Tax objects
Tax	erpTaxCode	Depending on the thomsonreuters.idt.response.summarization value, this field is used to populate TaxValue.code
Tax	authorityType	Depending on the thomsonreuters.idt.response.summarization value, this field is used to populate TaxValue.code

Tax	authorityName	Depending on the thomsonreuters.idt.response.summarization value, this field is used to populate TaxValue.code
Tax	taxAmount	Contains multiple fields, check Tax Amount level
Tax Amount	authorityAmount	Used to populate TaxValue.value and TaxValue.appliedValue

APPENDIX 2: TAX DATA MAPPING – SOAP

The table below describes the logic behind how SAP Commerce internal data structures are used to generate the Determination Request XML and how Determination Response XML is mapped back to SAP Commerce internal structures.

TAX REQUEST MESSAGE		
LEVEL	**DATA ELEMENT**	**SAP COMMERCE POPULATION LOGIC**
Header - UsernameToken	Username	Configuration: thomsonreuters.idt.soap.username
Header - UsernameToken	Password	Configuration: thomsonreuters.idt.soap.password Will be masked in logs
INDATA	version (attribute of INDATA)	Fixed value 'G'
INVOICE	EXTERNAL_COMPANY_ID	Configuration: thomsonreuters.idt.externalcompanyid
INVOICE	HOST_SYSTEM	Configuration: thomsonreuters.idt.hostsystem
INVOICE	CALLING_SYSTEM_NUMBER	Configuration: thomsonreuters.idt.callingsystemnumber
INVOICE	HOST_REQUEST_INFO / HOST_REQUEST_LOG_ENTRY_ID	This tag is only added to the request if the request is being resent by the retry mechanism in case of a connection error. The value is a string representation for the number of retries.
INVOICE	CALCULATION_DIRECTION	Based on “AbstractOrder.net” field. True → ‘F’, False → ‘T’
INVOICE	COMPANY_ROLE	Fixed value 'S'
INVOICE	CURRENCY_CODE	AbstractOrder.currency.isocode
INVOICE	DOCUMENT_TYPE	One of these: CART, ORDER or RETURN REQUEST
INVOICE	INVOICE_DATE	For carts: Get the current date For Orders: Order.date For returns: ReturnRequest.order.date "thomsonreuters.idt.returns.usegoodsacceptancedate". If config value is true ReturnRequest.tridtTaxPostingDate is used. In other cases ReturnRequest.order.date is used as

		date.
INVOICE	INVOICE_NUMBER	For carts: AbstractOrder.code+Timestamp (Timestamp is in “yyyy.MM.dd.HH.mm.ss” format) For orders: AbstractOrder.code For returns: ReturnRequest.code
INVOICE	IS_AUDITED	true/false For carts; always false For orders; if thomsonreuters.idt.request.audit.auditafterdelivery = false or delivery status is SHIPPED → true, otherwise false For returns; only true if return request status is one of COMPLETED, REVERSING_TAX, REVERSING_PAYMENT, TAX_REVERSED, PAYMENT_REVERSED, TAX_REVERSAL_FAILED, RECEIVED, PAYMENT_REVERSAL_FAILED
INVOICE	IS_CREDIT	Fixed value : true Only for return requests
INVOICE	ORIGINAL_DOCUMENT_ID	Associated order for the return request Only for return requests
INVOICE	ORIGINAL_INVOICE_DATE	The date for the associated order Only for return requests
INVOICE	TAX_DETERMINATION_DATE	The date for the associated order Only for return requests
INVOICE	TRANSACTION_TYPE	Fixed value 'GS'
INVOICE	CUSTOMER_NUMBER	In B2C: AbstractOrder.user.customerID In B2B: B2BUnit is found based on AbstractOrder.user and B2BUnit.uid is sent
INVOICE	CUSTOMER_NAME	Only valid in B2B scenarios B2BUnit is found based on AbstractOrder.user and “B2BUnit.name” is sent

INVOICE	DELIVERY_TERMS	Only valid in B2B scenarios. Either AbstractOrder.deliveryMode.tridtIncoterms or first 10 characters of AbstractOrder.deliveryMode.code Will be used only if thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms =true
INVOICE	REGISTRATIONS / SELLER_ROLE	If thomsonreuters.idt.registration.seller=true, BaseStore. tridtTaxRegistrationNumbers are added to the list
INVOICE	REGISTRATIONS / BUYER_ROLE	If thomsonreuters.idt.registration.buyer=true, B2BUnit. tridtTaxRegistrationNumbers are added to the list
INVOICE	SHIP_FROM	BaseStore.defaultDeliveryOrigin.address The AddressModel found will be the basis to fill the address fields described below
INVOICE	SHIP_TO	AbstractOrder.deliveryAddress The AddressModel found will be the basis to fill the address fields described below
INVOICE	BILL_TO	In B2C scenarios: Check AbstractOrder.paymentInfo.billingAddress, if not null use this address. If null, no Bill-To element is sent In B2B scenarios: Check AbstractOrder.paymentInfo, if it is a Credit Card, then the same as B2C, otherwise find B2BUnit associated with the user and send B2BUnit.billingAddress The AddressModel found will be the basis to fill the address fields described below
INVOICE	ORDER_ACCEPTANCE	Check system property thomsonreuters.idt.request.address.sendorderacceptanceaddr ess If the property is true, BaseStore.tridtOrderAcceptanceAddress will be sent. The AddressModel found will be the basis to fill the address fields described below
INVOICE	ORDER_ORIGIN	If activated by thomsonreuters.idt.request.address.sendorderoriginaddress: In B2C scenarios, will be the same as BILL-TO or SHIP_TO if the others are not available.

		In B2B scenarios, B2BUnit.contactAddress will be sent if found. If not, it will be the same as BILL-TO or SHIP_TO if the others are not available. The AddressModel found will be the basis to fill the address fields described below
INVOICE	SELLER_PRIMARY	Check system property thomsonreuters.idt.request.address.sendsellerprimary If the property is true, BaseStore.tridtOrderAcceptanceAddress will be sent. The AddressModel found will be the basis to fill the address fields described below
INVOICE	BUYER_PRIMARY	If activated by thomsonreuters.idt.request.address.sendorderoriginaddress: In B2C scenarios, will be the same as BILL_TO or SHIP_TO if the others are not available. In B2B scenarios, B2BUnit.contactAddress will be sent if found. If not, it will be the same as BILL_TO or SHIP_TO if the others are not available. The AddressModel found will be the basis to fill the address fields described below
ADDRESS	COUNTRY	Address.country.isoCode
ADDRESS	STATE	Address.region.isoCodeShort
ADDRESS	PROVINCE	Address.region.isoCodeShort Only for Canadian addresses
ADDRESS	COUNTY	Address.(field) field is determined based on thomsonreuters.idt.address.county
ADDRESS	CITY	Address.(field) field is determined based on thomsonreuters.idt.address.city
ADDRESS	DISTRICT	Address.(field) field is determined based on

		thomsonreuters.idt.address.district
ADDRESS	POSTCODE	Address.postalcode field value may be stripped based on thomsonreuters.idt.address.postalcodeIncludesGeoCode
ADDRESS	GEOCODE	Address.postalcode or Address.(field) Please check Configuration Options for the following system properties: thomsonreuters.idt.address.postalcodeIncludesGeoCode thomsonreuters.idt.address.geocode
LINE	ID	AbstractOrderEntry.entryNumber
LINE	LINE_NUMBER	AbstractOrderEntry.entryNumber
LINE	PRODUCT_CODE / COMMODITY_CODE	Please check Configuration Options for the following system properties: thomsonreuters.idt.request.item.sendCommodityCode thomsonreuters.idt.request.item.taxcode.enhancedsearch.productthierarchy.enabled thomsonreuters.idt.request.item.taxcode.enhancedsearch.categoryhierarchy.enabled thomsonreuters.idt.request.item.taxcode.fallbackCode
LINE	PART_NUMBER	AbstractOrderEntry.product.code
LINE	GROSS_AMOUNT	Used if “AbstractOrder.net” = true AbstractOrderEntry.TotalPrice + AbstractOrderEntry.DiscountValues
LINE	GROSS_PLUS_TAX	Used if “AbstractOrder.net” = false AbstractOrderEntry.TotalPrice + AbstractOrderEntry.DiscountValues
LINE	DISCOUNT_AMOUNT	Proportionally distributed AbstractOrder.DiscountValues + AbstractOrderEntry.DiscountValues
LINE	AMOUNT	AbstractOrderEntry.quantity

LINE	UOM	AbstractOrderEntry.unit
LINE	SHIP_TO	AbstractOrderEntry.deliveryPointOfService.address or AbstractOrderEntry.deliveryAddress Only if it is a pickup item or if the item delivery address is not empty
LINE	SHIP_FROM	AbstractOrderEntry.deliveryPointOfService.address Only if it is a pickup item
LINE	DELIVERY_TERMS	AbstractOrderEntry.deliveryMode.code Only valid in B2B scenarios Will be used only if item delivery mode is not empty and thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms =true
LINE (DELIVERY)	ID	Last entry number + 1
LINE (DELIVERY)	LINE_NUMBER	Last entry number + 1
LINE (DELIVERY)	PRODUCT_CODE / COMMODITY_CODE	thomsonreuters.idt.deliveryitem Please check Configuration Options for the following system properties: thomsonreuters.idt.request.item.sendCommodityCode thomsonreuters.idt.request.shipping.productcode
LINE (DELIVERY)	GROSS_AMOUNT	Used if “AbstractOrder.net” = true AbstractOrder.deliveryCost or a calculated value if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=true
LINE (DELIVERY)	GROSS_PLUS_TAX	Used if “AbstractOrder.net” = false AbstractOrder.deliveryCost or a calculated value if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=true
LINE (DELIVERY)	AMOUNT	Fixed value '1'
LINE (DELIVERY)	RELATED_LINE_NUMBER	Associated entry number (Only if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems)

		Not used for Returns
LINE (DELIVERY)	USER_ELEMENT/ATTRIBUTE1	Fixed value 'X' Flag to differentiate shipment lines from product lines, used in response processing

TAX RESPONSE MESSAGE (Only fields used in response processing are listed for simplification)		
LEVEL	**DATA ELEMENT**	**SAP COMMERCE MAPPING LOGIC** ExternalTaxDocument/target/lineItemTaxes/TaxValue for product lines and ExternalTaxDocument/target/shippingCostTaxes/TaxValue for shipment lines
INVOICE/LINE/TAX	AUTHORITY_NAME / ERP_TAX_CODE / AUTHORITYTYPE / EFFECTIVEZONELEVEL	Code ONESOURCE field is selected based on thomsonreuters.idt.response.summarization
INVOICE/LINE/TAX	TAX_AMOUNT/AUTHORITY_AMOUNT	Value Value is calculated based on thomsonreuters.idt.response.summarization. If summarization is requested, several TAX_AMOUNT/AUTHORITY_AMOUNT's are summed up based on the summarization level
INVOICE/LINE/TAX	-	Absolute Always false. SAP Commerce does not accept rates as external tax
INVOICE/LINE/TAX	TAX_AMOUNT/AUTHORITY_AMOUNT	appliedValue Same as value
INVOICE	CURRENCY_CODE	currency

APPENDIX 3: AVS DATA MAPPING – REST

The table below describes the logic behind how SAP Commerce internal data structures are used to generate the AVS Request JSON and how AVS Response JSON is mapped back to SAP Commerce internal structures.

AVS REQUEST MESSAGE		
LEVEL	**FIELD NAME**	**SAP COMMERCE POPULATION LOGIC**
Address	Address1	Address field form filled by the user
Address	Address2	Address field form filled by the user
Address	City	Address field form filled by the user
Address	Region	Address field form filled by the user
Address	Postal Code	Address field form filled by the user
Address	Country	Address field form filled by the user

AVS RESPONSE MESSAGE		
LEVEL	**FIELD NAME**	**SAP COMMERCE MAPPING LOGIC**
RequestStatus	Status	Status of the validation CONFIRMED, CORRECTED or ERROR
Messages	Status	List of messages returned by the service
ResponseAddress	Address1	Corrected value returned by the service
ResponseAddress	Address2	Corrected value returned by the service
ResponseAddress	City	Corrected value returned by the service
ResponseAddress	Region	Corrected value returned by the service
ResponseAddress	Postal Code	Corrected value returned by the service
ResponseAddress	Country	Corrected value returned by the service

APPENDIX 4: AVS DATA MAPPING – SOAP

The table below describes the logic behind how SAP Commerce internal data structures are used to generate the AVS Request XML and how AVS Response XML is mapped back to SAP Commerce internal structures.

AVS REQUEST MESSAGE		
LEVEL	**DATA ELEMENT**	**SAP COMMERCE POPULATION LOGIC**
Header - UsernameToken	Username	Configuration: thomsonreuters.avs.soap.username
Header - UsernameToken	Password	Configuration: thomsonreuters.avs.soap.password Will be masked in logs
ValidateAddressRequest	ExternalCompanyId	Configuration: thomsonreuters.avs.externalcompanyid
Address	Address1	Address field form filled by the user
Address	Address2	Address field form filled by the user
Address	City	Address field form filled by the user
Address	Region	Address field form filled by the user
Address	Postal Code	Address field form filled by the user
Address	Country	Address field form filled by the user

AVS RESPONSE MESSAGE		
LEVEL	**DATA ELEMENT**	**SAP COMMERCE MAPPING LOGIC**
RequestStatus	Status	Status of the call SUCCESS or ERROR
AddressResponse	Status	Status of the validation CONFIRMED, CORRECTED or ERROR

ResponseAddress	Address1	Corrected value returned by the service
ResponseAddress	Address2	Corrected value returned by the service
ResponseAddress	City	Corrected value returned by the service
ResponseAddress	Region	Corrected value returned by the service
ResponseAddress	Postal Code	Corrected value returned by the service
ResponseAddress	Country	Corrected value returned by the service