# ONESOURCE™ SAP COMMERCE INTEGRATION

## SETUP, CONFIGURATION AND USER GUIDE

VERSION 1.1.0.0

Document Version 1

THOMSON REUTERS®

# DOCUMENT HISTORY

| VERSION NUMBER | VERSION DATE | SUMMARY |
|---|---|---|
| V1 | March 2021 | Initial release |

# TABLE OF CONTENTS

# INTEGRATION OVERVIEW

This is an overview of the TAX calculation integration between SAP Commerce 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

As of version 1.1.0.0, the solution supports the following countries for tax integration:

- United States
- Canada
- European Union countries

## Prerequisites and Steps

For a 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 | Local and Cloud environment | • ONESOURCE IDT Integration extensions to be added to the system setup<br>• Building and updating the local system to activate the extensions.<br>• Maintenance of system configuration for connecting to the ONESOURCE IDT system. |
| 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.<br>• Username and Password required to send tax calculation requests |

## Supported SAP Commerce versions

SAP Commerce Cloud 2011

# Benefits

### Integration

ONESOURCE Indirect Tax Integration extensions seamlessly connect your SAP Commerce system to ONESOURCE Determination for tax calculations and appropriate return of tax results. The extensions are developed and maintained in-house by a team of Thomson Reuters Business Systems Analysts, Developers, and Quality Assurance employees providing the most advanced tax engine determination capability and compliance returns processing globally. Our solution can be fully assimilated into your existing e-commerce systems using our open integration architecture. Tax calculation calls can be easily inserted into existing system workflows and processes to deliver real-time solutions with accurate tax results. This integration is fully embedded in the SAP Commerce architecture. It can easily be deployed in SAP Public Cloud environments and can be configured and troubleshot on its native user interfaces like Administration Console, Backoffice and Kibana.

In case you have specific requirements, it is also possible to easily extend it just like out of the box SAP extensions by your own project resources or Thomson Reuters Professional Services team.

### Determination

ONESOURCE Indirect Tax Determination enables companies to consolidate their global tax policy in one central location. All enterprise-wide applications can use a single scalable instance of Determination and still deliver business-specific tax policy across multiple-business systems. Fully integrated to all your financial applications, Determination enables the passing of transaction data from the financial system to the tax engine and returns transaction taxes in real-time for fast, reliable, and accurate indirect tax determination. We offer fully supported standard Oracle and SAP integrations, as well as custom integrations via our tax calculation web service.

### Tax Certificate Manager

ONESOURCE Indirect Tax Certificate Manager is a solution for the precise tracking, validating, and governing of exemption certificates. As part of ONESOURCE, it provides integration to our ONESOURCE Indirect Tax Determination software that allows for the export of customers and exemption certificates. ONESOURCE Indirect Tax Certificate Manager improves efficiency in all aspects of the burdensome exemption certificate life cycle by reducing operating costs, mitigating risk, and increasing accuracy. ONESOURCE Indirect Tax Certificate Manager reduces audit exposure and assessments while empowering you with full control of the exemption certificate process to maintain Sarbanes-Oxley compliance.

### Reporting

ONESOURCE Indirect Tax Reporting software provides fast, accurate, and flexible reporting that's fully integrated with our ONESOURCE Indirect Tax global software suite to support your global compliance, reconciliation, and data analysis processes. An easy-to-use interface provides a library of over 40 production-ready reports that can deliver the most relevant data in a few simple clicks. Drill-down capabilities provide a way for you to quickly explore the underlying data details, all the way down to the lowest level individual authority taxes. Our summary-level or detail-level reports allow you to choose the type of report data that best meets your immediate tax data needs in the most efficient way possible.

**Compliance for US**

Regardless of location or industry, Sales & Use Tax Compliance has the forms required to meet your needs. It provides over 600 signature-ready state and local returns that are facsimiles of the official forms. Returns and schedules include sales, seller's use, consumer's use, and rental tax forms for all applicable states, as well as the District of Columbia. Industry-specific food and beverage returns are also included. In addition, more than 70 electronic returns are available and accepted in over 25 states. Sales & Use Tax Compliance is one of the market leaders in e-filing support. Thomson Reuters continues to work directly with state taxing authorities to ensure full compliance with each state's unique electronic filing requirements. The software also goes beyond borders to include the returns required for tax compliance in Canada.

# Architecture and Design Overview

In order 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. Includes the following functionality:
    - Implements SAP's External Tax Service interface.
    - Responsible to populate request details common for B2C and B2B
    - Responsible to make the SOAP call to the external service
    - Responsible to convert the SOAP response into SAP's data structures
    - Requires and auto loads following SAP extensions: commerceservices and springintegrationlibs
- **b2btrtaxintegration**: This extension is optional and should only be used if you have a B2B Storefront. Includes the 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 the 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 provides extra information regarding the reason for an invalid address
    - for address verification that returns more information about invalid addresses compared to the standard OCC end-point
    - Requires trtaxintegration as a prerequisite
    - Requires and auto loads following SAP extension: commercewebservices

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

- **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 SOAP call to the external service
    - Responsible to convert the SOAP 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.

## 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.
- The tax integration is only limited to the checkout pages of the storefront. 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.
- Address Verification Integration only supports verification of United States 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.

# INSTALLATION

## Local Environment Installation

- Decide which extensions are required for you 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 *'travsintegrationaddon'* is optional.
    - *<extension name='travsintegration ' />*
    - *<extension name= travsintegrationaddon />*
- If you added *travsintegrationaddon*, run the following ant command to install the addon for the required storefront extension.

Ex: ant addoninstall -Daddonnames="travsintegrationaddon" -DaddonStorefront.yacceleratorstorefront="{{myextension}}" (Please replace {{myextension}} with the name of the storefront extension in your setup )

- Build your system with “ant all"
- Update your system with “ant updatesystem”.

### *Activating External Tax Calls*

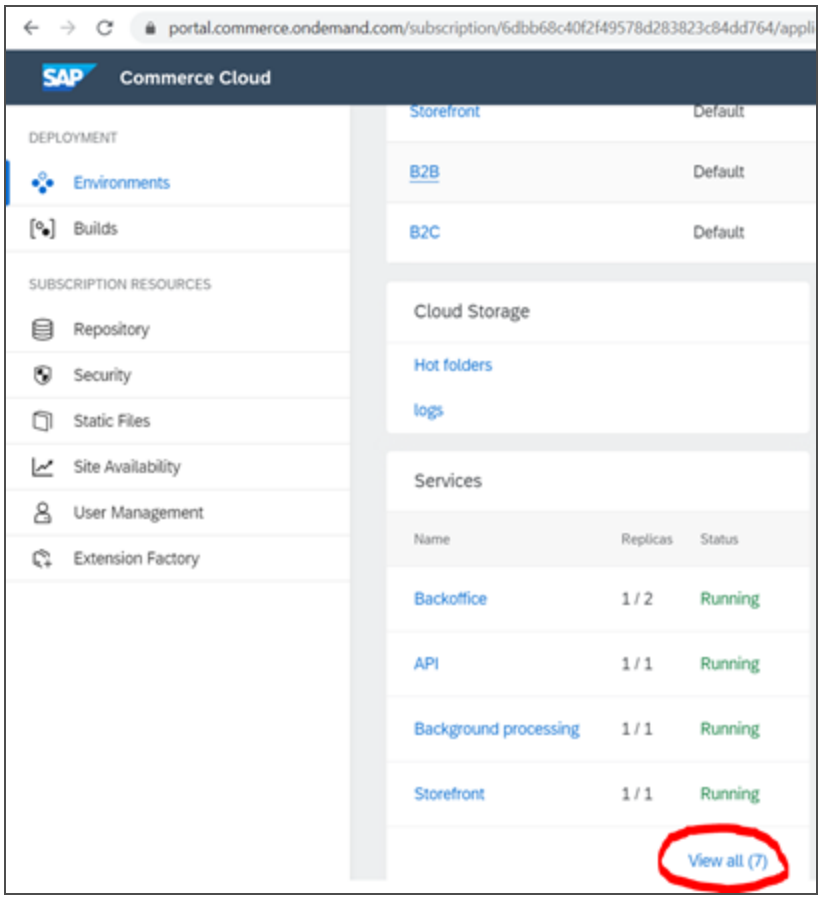
For ONESOURCE IDT external tax call to ONESOURCE IDT to be triggered in the system, the following prerequisites should be met:

- 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 a delivery address and a delivery mode assigned
- In case of a pickup in-store scenario, a point of service address should be maintained.

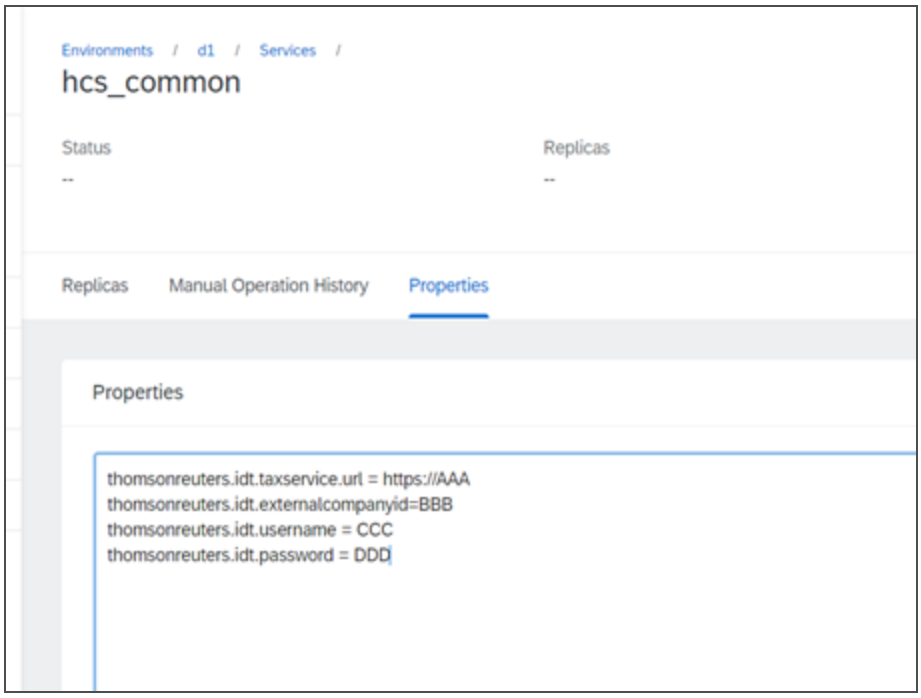
## SAP Public Cloud Environment

- 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.

- Since it is a security best practice to not to keep any sensitive data in the code repository, you can provide the username and password necessary to connect to ONESOURCE IDT on the SAP Commerce Cloud Portal. Please carry out the following steps:
    - Login to https://portal.commerce.ondemand.com/ with your SAP credentials
    - Under Environments choose your environment
    - Click “View All” link in “Services” tile of your environment




- Choose “hcs_common” and “Properties” tab
- Provide your username and password
  thomsonreuters.idt.taxservice.url = https://AAA
  thomsonreuters.idt.externalcompanyid=BBB
  thomsonreuters.idt.username = CCC
  thomsonreuters.idt.password = DDDin the following format:



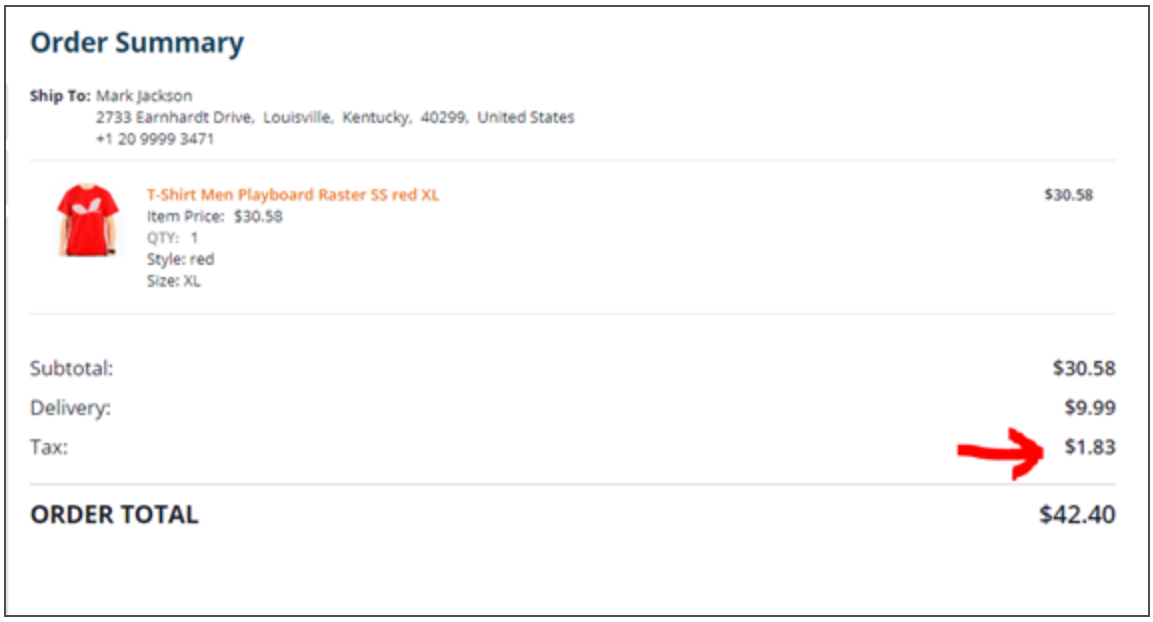

- Click "**Save**" button and then "**Apply Configurations**". Your application servers will be restarted with the provided values.

# 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 in order to calculate the exact sales tax. After each call Determination response is read, summarized if necessary and persisted on the SAP Commerce cart object. The tax call occurs when the user clicks “Next” to proceed to the following step. The result for the total tax value is displayed in the “Order Summary” section of the next checkout step.




In this integration, the tax call is only triggered if any tax critical information is updated between checkout steps. If nothing is changed on the document, no calls will be initiated while transitioning to the next step.

Here are the major fields that affect tax calculation:

- Add or remove cart items
- Changes in existing cart items
    - Quantity
    - Price
    - Product
    - Delivery address
    - Delivery mode
    - Pickup store
    - Delivery address
    - Delivery mode
    - Delivery cost
    - Billing address
    - Header promotion

**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 taken into account 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 the 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, provided that 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. AbstractOrderModel.net field value is taken into account while preparing the tax request so that in the request XML:

- For net pricing
  - <CALCULATION_DIRECTION> is set as "F – Forward"
  - <GROSS_AMOUNT> tag is used for the item amount
- For gross pricing
  - <CALCULATION_DIRECTION> is set as "R – Reverse"
  - <GROSS_PLUS_TAX > tag is used for the item amount

## *Taxes for the Delivery Cost*

The ONESOURCE Determination integration supports tax calculation for delivery costs. This delivery cost can be sent in two different ways:

- Single delivery cost as an additional item : For each tax request an additional item is added to the order for the delivery.
    - AbstractOrderModel.deliveryCost field is used as the amount for this item.
    - The product/commodity code for the delivery cost item is determined by the “thomsonreuters.idt.request.shipping.productcode” system property.
    - Quantity will always be 1.
    - The tax results are kept as TaxValue’s in cart/order header (AbstractOrderModel.totalTaxValues).
    - Distributed delivery cost per item: 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.
        - 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.
        - 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.
        - The tax results are kept as TaxValues in items rather than in the header (AbstractOrderEntry.taxValues). A prefix is added to the tax code (Delivery-State) to differentiate delivery cost tax values from product tax values. The summarization options valid throughout the system also apply for the delivery cost taxes.

Single 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 .

## *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 shipping 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.

**Example**: Suppose the following taxes are calculated and returned by the Determination System for a cart having no delivery cost. You will see result of each different summarization option in the relevant section below.

| CART ENTRY NO: | TAX AUTHORITY | TAX VALUE ($) | ERP TAX CODE | AUTHORITY TYPE | ZONE LEVEL |
|---|---|---|---|---|---|
| 1 | A1 | 2.00 | T1 | State Sales/Use | State |
| 1 | A2 | 3.00 | T1 | County Sales/Use | County |
| 2 | A1 | 4.00 | T1 | State Sales/Use | State |
| 2 | A2 | 5.00 | T1 | County Sales/Use | County |
| 2 | A3 | 6.00 | T2 | County Sales/Use | County |
| 3 | A1 | 7.00 | T1 | State Sales/Use | State |
| 3 | A2 | 8.00 | T1 | County Sales/Use | County |
| 3 | A3 | 9.00 | T1 | County Sales/Use | County |
| 3 | A4 | 10.00 | T3 | County Rental | County |

You can choose one of the following options for summarization and set to thomsonreuters.idt.response.summarization system property:

- **SummaryByErpCode**: The tax values are grouped by <TAX><ERP_TAX_CODE> tag in the response. In other words, the tax values having the same ERP code are consolidated into a single tax value.

Ex: Here is the tax value structure you'll find in the cart in SAP Commerce for the example above:

| CART ENTRY NO: | TAX CODE | TAX VALUE ($) |
|---|---|---|
| 1 | T1 | 5.00 |
| 2 | T1 | 9.00 |
| 2 | T2 | 6.00 |
| 3 | T1 | 24.00 |
| 3 | T3 | 10.00 |

- **SummaryByAuthority**: The tax values are grouped by <TAX><AUTHORITY_TYPE> tag in the response. In other words, the tax values having the same authority types are consolidated into a single tax value.

Ex: Here is the tax value structure you'll find in the cart in SAP Commerce for the example above:

| CART ENTRY NO: | TAX CODE | TAX VALUE ($) |
|---|---|---|
| 1 | State Sales/Use | 2.00 |
| 1 | County Sales/Use | 3.00 |
| 2 | State Sales/Use | 4.00 |
| 2 | County Sales/Use | 11.00 |
| 3 | State Sales/Use | 7.00 |
| 3 | County Sales/Use | 17.00 |
| 3 | County Rental | 10.00 |

- **SummaryByZone**: The tax values are grouped by <TAX><EFFECTIVE_ZONE_LEVEL> tag in the response. In other words, the tax values having the same zone level are consolidated into a single tax value.

Ex: Here is the tax value structure you'll find in the cart in SAP Commerce for the example above:

| CART ENTRY NO: | TAX CODE | TAX VALUE ($) |
|---|---|---|
| 1 | State | 2.00 |
| 1 | County | 3.00 |
| 2 | State | 4.00 |
| 2 | County | 11.00 |
| 3 | State | 7.00 |
| 3 | County | 27.00 |

- **FullDetails**: The tax values are not grouped at all and converted to TaxValue's as is.

Ex: Here is the tax value structure you'll find in the cart in SAP Commerce for the example above:

| CART ENTRY NO: | TAX CODE | TAX VALUE ($) |
|---|---|---|

| | | |
|---|---|---|
| 1 | A1 | 2.00 |
| 1 | A2 | 3.00 |
| 2 | A1 | 4.00 |
| 2 | A2 | 5.00 |
| 2 | A3 | 6.00 |
| 3 | A1 | 7.00 |
| 3 | A2 | 8.00 |
| 3 | A3 | 9.00 |
| 3 | A4 | 10.00 |

## *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).

## *Canada Specific Functionality*

- In a tax request, for all Canadian addresses, <PROVINCE> tag is used instead of <STATE>.

## *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:
    - <IS_AUDITED> tag value is set 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.
    - Please note that if all items are fully cancelled and nothing remains to be shipped, no audit tax call happens, since "Tax Posting" business process step is never reached in the order business process. In this case, no corrections need to be posted to the Determination engine, since no audit call were made previously for this order.
    - In order to be consistent with the estimation calculations in checkout, the <INVOICE_DATE> tag which is the basis for the tax calculation date, is set as the order date (AbstractOrderModel.date field), not the actual posting date.
    - In order 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.
    - The business process definition running behind the orders should also be updated. A sample process definition 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-order-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 order business process keeping in mind the following points:
        - The example process introduces two steps: tridtTaxAuditUpdateAfterPartialCancellation and tridtTaxAuditVoidAfterFullCancellation which handles audit updates in case or a partial or full cancellation.
        - Look for actions that have "transition name='CANCELLED'" or wait nodes where "choice id='cancelled'" in your original process definition and make sure the flow is directed to the new steps above after those transitions depending on the nature of the cancellation.
        - 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.
    - 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. submitOrderProcessCode field)

## *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.

In order 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.

**Note**: In your custom customizations, if you have to call CalculationService interface, please consider calling ExternalTaxesService right after in order 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*

In order to efficiently manage taxes in your return processes, return business process definition has to be customized.

- Similar to orders, 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 tax values can be taken into account 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 : Similar to 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: Similar to 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 : Similar to 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: Similar to 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.

Note: 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:

- Single delivery cost as an additional item method is used for the delivery cost. Please check "Taxes for the Delivery Cost" section for details.
- For technical reasons, the system performs two calls to ONESOURCE Determination, one after the other. First, an estimation tax call is performed for the entire sales order. The results are than used for preparation of the request XML of the second tax call that will trigger the calculation of taxes for the return request. This logic is valid both for estimation and audit calls.

## *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 activated in the base store assigned to the order.
- Two new buttons are introduced instead of the two actions above as "Recalculate order totals with external tax" and "Calculate with promotions with external tax" that will call Determination and update tax values in the order and respective order entries after executing out-of-the-box calculation 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*

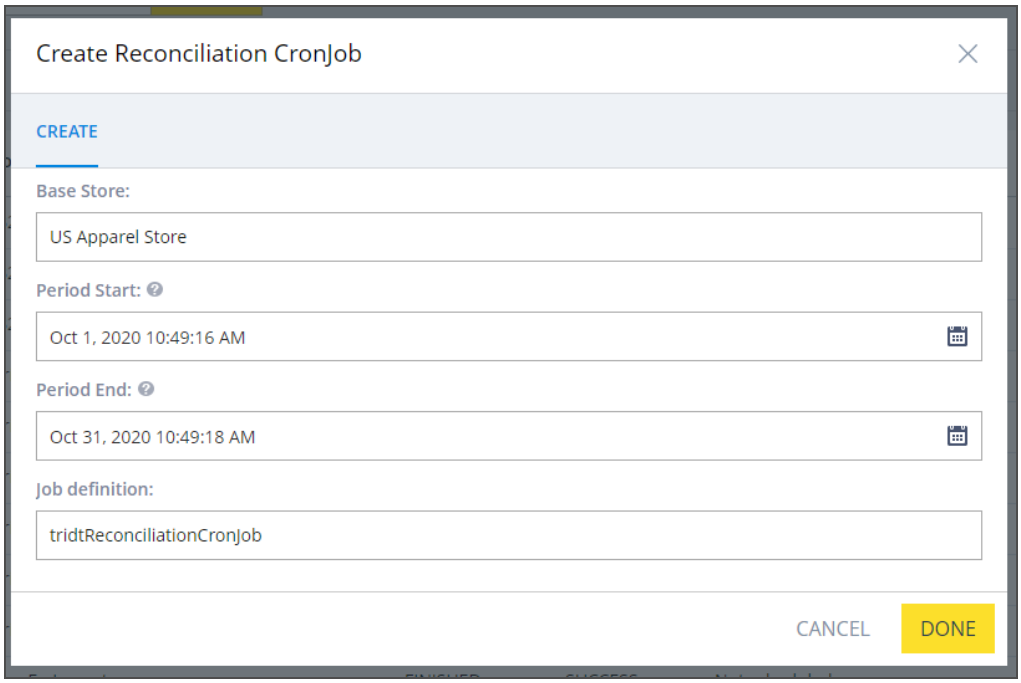
The integration package provides a reconciliation report that can be used in month-end closing. By running a cronjob, users can download a list of audited orders and returns in a specified period for a specified base store and compare the results with ONESOURCE Determination reports in order to identify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

In order to initiate the cronjob:

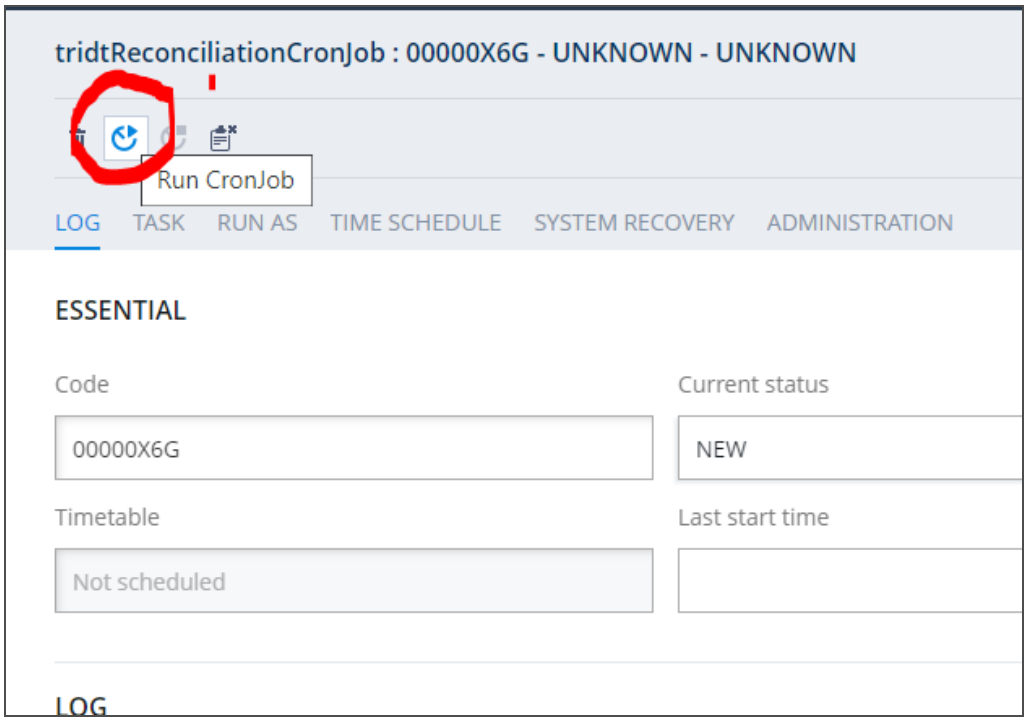
- In Backoffice UI, go to System/ Background Processes / Cronjobs
- Create a new cronjob by clicking on the little triangle near the "Create new Cronjob" button and select TR Tax Reconciliation Job.

- 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.

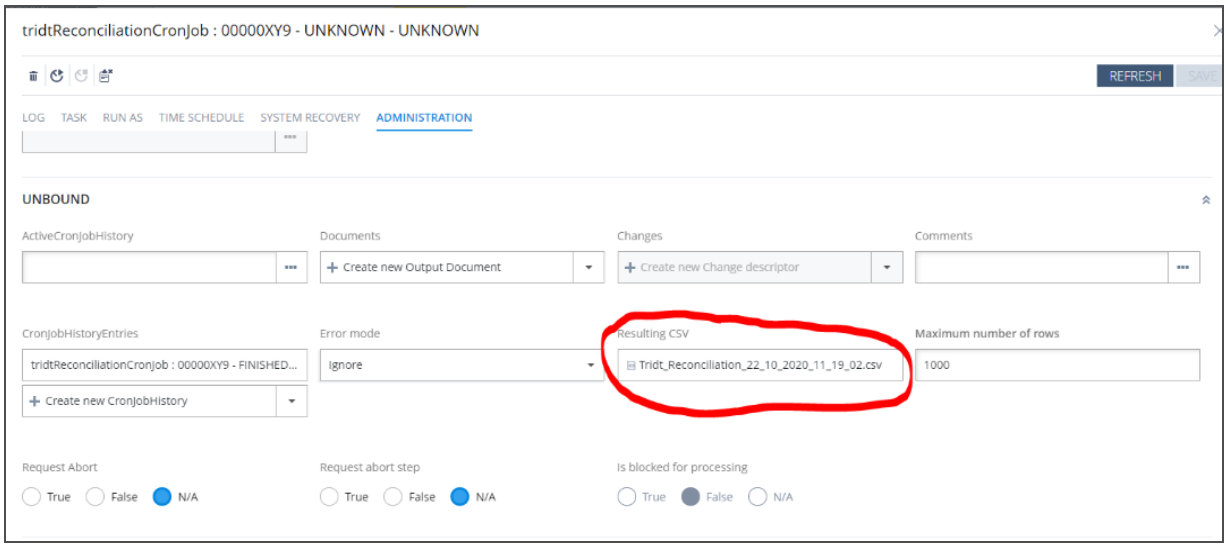
Ex: For a user in EST (browser timezone) on a system in CST, Oct 1, 2020 10:49:16 AM EST is adjusted as Oct 1, 2020 12:00:00 AM CST for a period start date.




- 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.

- 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 |
|---|---|

| Connection | |
|---|---|
| thomsonreuters.idt.taxservice.url | The value will be provided by Thomson Reuters<br><br>Default value: Empty |
| thomsonreuters.idt.externalcompanyid | The value will be provided by Thomson Reuters. The value in this parameter may be overridden if an External Company ID is defined for the base store of the cart/order.<br><br>Default value: Empty |
| thomsonreuters.idt.username | The value will be provided by Thomson Reuters<br><br>Default value: Empty |
| thomsonreuters.idt.password | The value will be provided by Thomson Reuters<br><br>Default value: Empty |
| thomsonreuters.idt.hostsystem | String that represents the role of the system. Ex: DEV, QA, PROD etc.<br><br>Default value: DEV |
| thomsonreuters.idt.callingsystemnumber | String that represents the calling system sending the transaction.<br><br>Default value: master |
| thomsonreuters.idt.connection.connectiontimeout | Connection timeout duration in milliseconds<br><br>Default value: 5000<br><br>This setting can only be defined system-wide and cannot be defined for a Base Store |

| | |
|---|---|
| thomsonreuters.idt.connection.readtimeout | Read timeout duration in milliseconds<br>Default value: 5000<br>This setting can only be defined system-wide and cannot be defined for a Base Store |
| thomsonreuters.idt.connection.retry.delay | Waiting duration between retry attempts in milliseconds<br>Default value: 1000<br>This setting can only be defined system-wide and cannot be defined for a Base Store |
| thomsonreuters.idt.connection.retry.attempts | Number of retries in case of a connection error<br>Default value: 3<br>This setting can only be defined system-wide and cannot be defined for a Base Store |
| Address | |

| | |
|---|---|
| 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 <POSTAL_CODE> tag and the second part will be used for the <GEOCODE> tag in the request XML. If no hyphen is detected, the value will be used just for the <POSTAL_CODE> tag and <GEOCODE> will be left empty.<br><br>If it is false Address.postalCode field is just used for the <POSTAL_CODE> tag and thomsonreuters.idt.address.geocode will be used to generate <GEOCODE> content in the request XML.<br><br>Default value: true |
| thomsonreuters.idt.address.sendGeoCode | If it is true, geocode will be included in the request. The value will depend on thomsonreuters.idt.address.postalcodeIncludesGeoCode and thomsonreuters.idt.address.geocode parameters.<br><br>Default value: false |
| thomsonreuters.idt.address.geocode | Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property<br><br>Default value: Empty |
| thomsonreuters.idt.address.county | Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property<br><br>Default value: district |

| | |
|---|---|
| thomsonreuters.idt.address.city | Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property<br><br>Default value: town |
| thomsonreuters.idt.address.district | Any String typed fieldname (SAP standard or custom) on “Address” type can be provided as a parameter for this property<br><br>Default value: district |
| 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 is also used as the order acceptance address.<br><br>Default value: false |
| thomsonreuters.idt.request.address.sendorderoriginaddress | In B2C, if set to true, ship-to address will be copied to the order-origin address.<br><br>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, ship-from address will be copied to order origin address.<br><br>Default value: false |
| 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 bill-to address is also used as the seller primary address.<br><br>Default value: false |

| | |
|---|---|
| thomsonreuters.idt.request.address.sendbuyerprimary | In B2C, if set to true, bill-to address will be copied to the buyer primary address.<br><br>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 address will be copied to buyer primary address.<br><br>Default value: false |
| Product Code / Commodity Code Determination | |
| thomsonreuters.idt.request.item.sendCommodityCode | If set to true, <COMMODITY_CODE> tag will be sent for each line item. <PRODUCT_CODE> tag will be sent otherwise.<br><br>Default value: false |
| thomsonreuters.idt.request.item.taxcode.enhancedsearch.productthierarchy.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.<br><br>Default value: false |
| 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.<br><br>Default value: false |

| | |
|---|---|
| thomsonreuters.idt.request.item.taxcode.fallbackCode | If no value is found for product/commodity code, this text will be used as a fallback code.<br><br>Default value: Empty |
| Delivery Cost | |
| thomsonreuters.idt.request.shipping.productcode | This value will be used as the default product/commodity code for the delivery cost item.<br><br>Default value: FREIGHT |
| thomsonreuters.idt.request.b2b.senddeliverymodeasincoterms | TOnly in B2B scenarios, the code of the DeliveryMode associated with the cart, can be sent in the <DELIVERY_TERMS>. This option can be used if Incoterm values are kept as DeliveryModes in SAP Commerce and the value chosen affects the tax calculation.<br><br>Default value: false<br><br>Only valid for B2B scenario |
| thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems | If activated, a separate delivery cost item will be included in the request for each delivery related item in cart/order. Delivery cost on the header will be distributed to each delivery item based on the respective item value.<br><br>Default value: false |
| 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.<br><br>Default value: false |

| | |
|---|---|
| thomsonreuters.idt.registration.buyer | If activated, any registration number maintained in the B2BUnit will be included in the request as buyer registration numbers.<br><br>Default value: false<br><br>Only valid for B2B scenario |
| Response Summarization | |
| thomsonreuters.idt.response.summarization | The value should be either of these four values:<br><br>- SummaryByErpCode<br><br>- SummaryByAuthority<br><br>- SummaryByZone<br><br>- FullDetails<br><br>Details are described in section Summarization Options for Tax Authority Details<br><br>Default value: SummaryByZone |
| thomsonreuters.idt.response.shipping.prefix | This prefix will be used for naming every delivery cost related tax value.<br><br>Default value: Delivery- |
| Returns | |
| thomsonreuters.idt.returns.usegoodsacceptancedate | If set to true, goods acceptance date is sent as the base date for tax calculation.<br><br>If set to false, the date field for the associated order is used instead.<br><br>Default value: false |
| 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 in order not to stop the user from completing the checkout.<br><br>Default value: false |
| thomsonreuters.idt.fallback.fixedpercentage | The fixed-rate for the fallback process.<br><br>Default value: Empty |
| Logging | |
| thomsonreuters.idt.detailedlogging.request | If set to true, the details of the request XML will be saved in the logs.<br><br>Default value: true |
| thomsonreuters.idt.detailedlogging.response | If set to true, the details of the response XML will be saved in the logs.<br><br>Default value: true |

## *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.

**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.

You can achieve this in five steps:

1. Decide on the hierarchy. Customer attributes can be sent either in:
    a. Header
    b. Product item
    c. Delivery cost items
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:
        i. source: de.hybris.platform.core.model.order.AbstractOrderModel
        ii. target: taxcalculationservice.wsdl.IndataType
    b. For product item attributes:
        i. source: com.thomsonreuters.idt.integrations.sapcommerce.models.TridtDiscountedOrderEntry
        ii. target: taxcalculationservice.wsdl.IndataLineType
    c. For delivery cost item attributes:
        i. source: de.hybris.platform.core.model.order.AbstractOrderModel
        ii. target: taxcalculationservice.wsdl.IndataLineType
4. Implement the business logic for each class. Add name/value pairs in taxcalculationservice.wsdl.UserElementType type to the respective structure. Currently restricted to the values ATTRIBUTE2 through ATTRIBUTE50. Note that ATTRIBUTE1 is already used in the standard logic in trtaxintegration extension.
5. Manage dependency injection in {{extension}}-spring.xml file
    a. Add a new bean for the populator. Ex:
        i. <alias name=**"{{populator bean ID}}"** alias=**"{{populator bean ID}}"** />
        ii. <bean id=**"{{populator bean ID}}"** class=**"{{Populator class path}}"**/>
    b. Add the populator bean to the converter bean as a dependency

**For header attributes:**

<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" />

<ref bean=**"{{populator bean ID}}"** />

</list>

</property>

</bean>

**For item attributes:**

<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:**

<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 prerequisites described in section Business Process Description - Checkout are evaluated to decide whether a new tax call is necessary.

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 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 utilized in case you decide to customize your checkout, order details and return details pages.

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

# LOGGING AND MONITORING

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.

| STEP NO | LOG LEVEL | CLASS | TEXT |
|---|---|---|---|
| 1 | INFO | DefaultTridtTaxCalculationService | Starting ONESOURCE Determination process... |
| 2 | DEBUG | TridtSoapLoggingService | Log ID: {1}, ONESOURCE Determination Request XML: {2}: Request XML<br><br>Note: The message is only activated if thomsonreuters.idt.detailedlogging.request=true and DEBUG log level is activated for class TridtSoapLoggingService |
| 3 | DEBUG | TridtSoapLoggingService | Log ID: {1}, ONESOURCE Determination Response XML: {2}<br><br>{1}: Unique request ID that includes the cart id and the timestamp<br><br>{2}: Request XML<br><br>Note: The message is only activated if thomsonreuters.idt.detailedlogging.response=true and DEBUG log level is activated for class TridtSoapLoggingService |
| 4 | INFO | DefaultTridtTaxCalculationService | Request stats for ID: {1} , Request Preparation Duration: #{2}#, Response Waiting Duration: ##{3}##, Response Processing Duration: ###{4}###<br><br>{1}: Unique request ID that includes the cart id and the timestamp<br><br>{2}: Time passed while preparing the request<br><br>{3}: Time passed in Determination System processing<br><br>{4}: Time passed while processing the response |

In SAP Public Cloud Deployments these logs can be displayed in Kibana user interface. As a part of the integration package, three 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
- The dashboards need three scripted fields to be available in your Index Pattern. Those fields should be created manually
    - Go to Kibana/ Management / Index Patterns / Scripted Fields tab
    - Follow instructions in Tridt_Kibana_ScriptedFields.txt to create three scripted fields
- Import dashboards, visualizations and saved searches
    - Go to Kibana/ Management / Saved Objects and click import
    - Choose “Tridt_Kibana_Customizations.json” located in resources\kibana folder and execute
- Start using your dashboards and saved searches in your SAP Commerce Public Cloud environment.

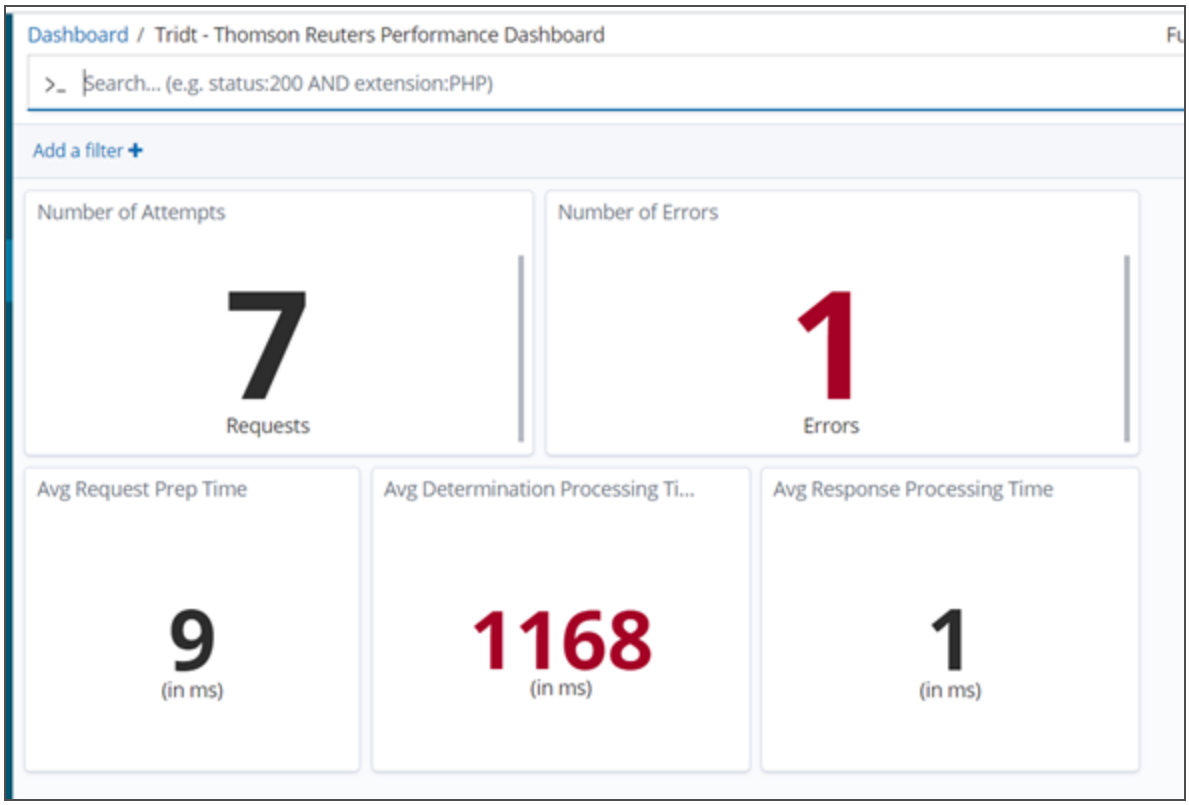
**Note**: The dashboards, visualizations and saved searches are based on the logs described in the table above. They will not work properly, if any of these log items are deactivated by setting a lower log level in log4j2.

## Performance Dashboard:

The dashboard includes several visualizations that show critical metrics about the overall status and performance of the integration 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
- Average Request Preparation Time (SAP Commerce)
- Average Determination Processing Time (Connection + Determination Engine)
- Average Response Processing Time (SAP Commerce)

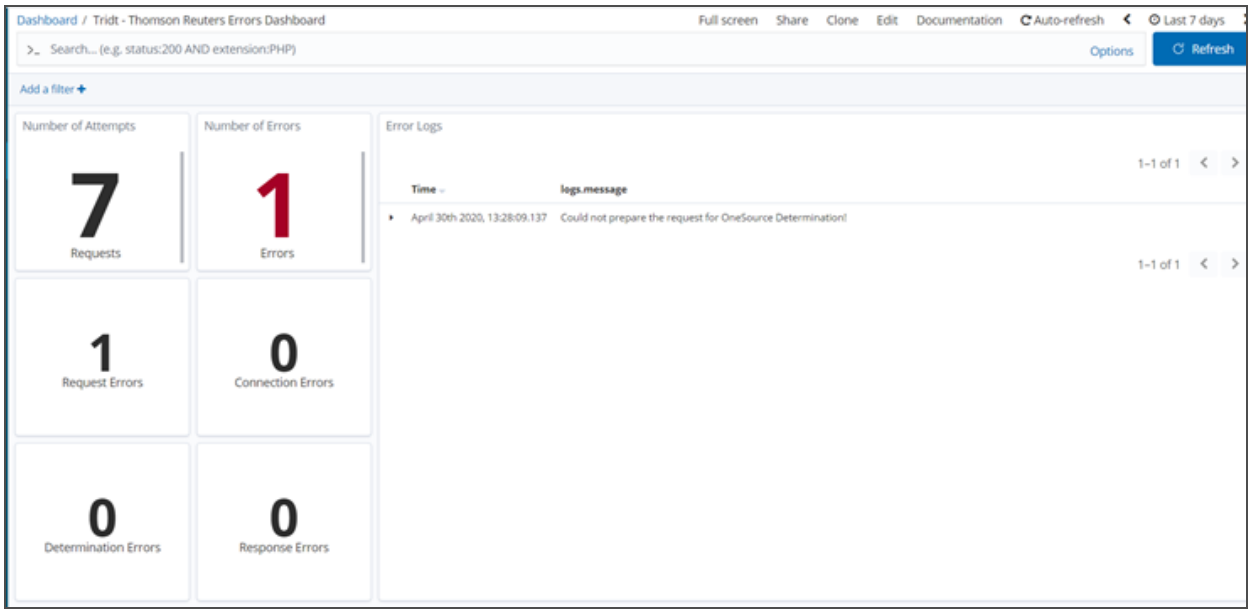



## Errors Dashboard:

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

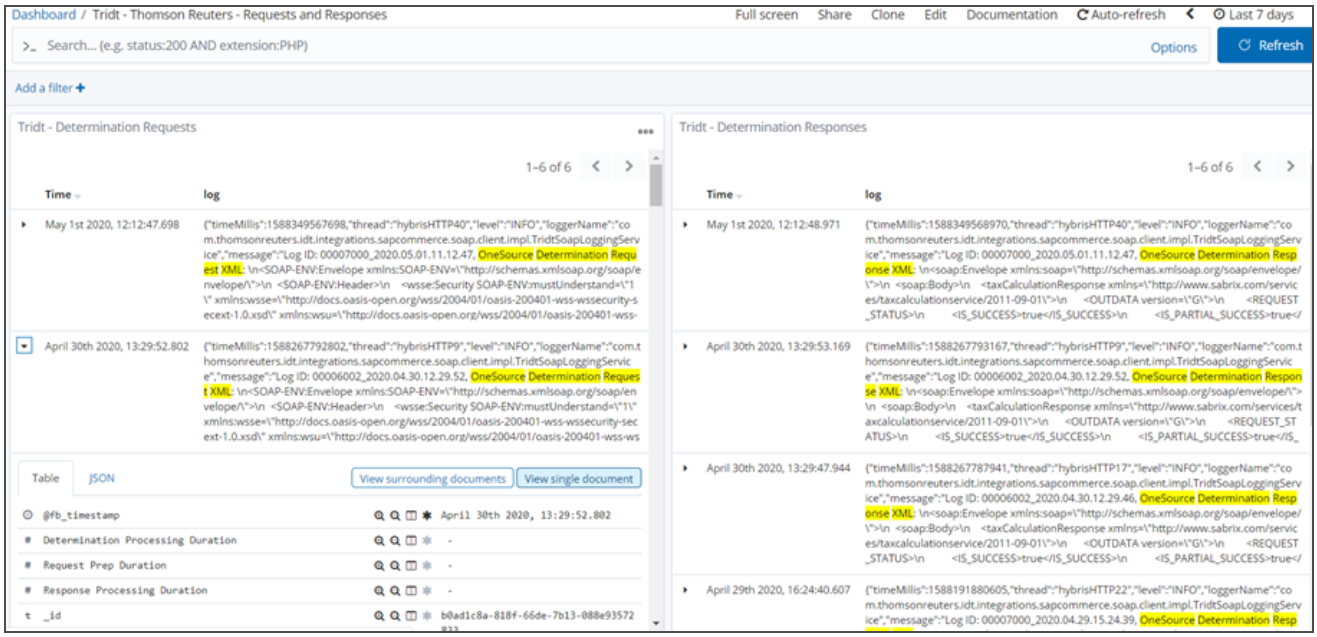



## Requests and Responses Dashboard:

The dashboard includes two saved searches for request and response XMLs in a specified time frame. Please note that detailed logging should be activated for having XML 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.




**Tip**: 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_*).

# TROUBLESHOOTING SCENARIOS

## Classification of Higher-Level Exceptions:

A typical tax process is composed of 4 different sub-processes and they run one after the other.

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

You can make a search with the following keywords in the logs in order to understand which part of the process is causing the error.

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

| KEYWORD TO SEARCH | SUBPROCESS THAT IS THROWING THE EXCEPTION | RESPONSIBLE SYSTEM | DETAILS |
|---|---|---|---|
| TridtRequestException | Request Preparation | SAP Commerce | Request Preparation Exceptions |
| TridtConnectionException | Establishing a connection to Determination | SAP Commerce | Connection Exceptions |
| TridtDeterminationException | Determination Engine's internal process | Determination | Determination Errors |
| TridtResponseException | Response Processing | SAP Commerce | Response Processing Exceptions |
| 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 of 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 assigned to the Base Store cannot be null. | BaseStore.defaultDeliveryOrigin field is 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 |

## 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 could not be converted into SAP's internal TaxValue structure
- Any exception in **de.hybris.platform.commerceservices.externaltax.impl.DefaultExternalTaxesService.saveOrder** indicates 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 received, does not have OutdataInvoiceType inside. |
| NullPointerException | Cannot invoke "java.util.List.isEmpty ()" because "lines" is null | The response XML received, does not have OutdataLineType inside. |
| NullPointerException | Cannot invoke "java.util.List.stream ()" because "taxList" is null | There is no tax value info for a line in the response XML. |

| | | |
|---|---|---|
| NullPointerException | element cannot be mapped to a null key | One of the tax values in the response message doesn't have the necessary key for summarization.<br><br>Check thomsonreuters.idt.response.summarization system property and check why that field is not coming up from Determination. |
| DataIntegrityViolationException | ModelSavingException query; SQL []; String or binary data would be truncated.;<br><br>nested exception is com.microsoft.sqlserver<br><br>.jdbc.SQLServerException:<br><br>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.<br><br>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 to 1000 characters. |
| NumberFormatException | error parsing tax value string '<TV<XX-XXX #[ATT]sales [ATT]#4.07#true#4.07#USD>VT>' :<br><br>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.<br><br>Check the response XML for details. |

## 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 SOAP Request is malformed or contains invalid authentication credentials. The SOAP Response would contain a <soap: Fault> element with two (2) child elements <faultcode> and <faultstring> as shown below:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
        <soap:Fault>
        <faultcode
xmlns:ns1="http://ws.apache.org/wss4j">ns1:SecurityError</faultcode>
        <faultstring>A security error was encountered when verifying the
message</faultstring>
        </soap:Fault>
        </soap:Body>
        </soap:Envelope>
```

Here are some examples:

| EXCEPTION | TEXT | REASON |
|---|---|---|
| SocketException | Connection timed out (Read failed) | Determination failed to send any response. This indicates an internal problem in Determination System.<br><br>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.url |
| SoapFaultClientException | A security error was encountered when verifying the message | Username and password can be wrong.<br><br>Check thomsonreuters.idt.username and thomsonreuters.idt.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.SSLExceptio n: 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 real problem is in request preparation. You need to investigate logs for further details.

Determination Request Errors indicates that the Request was unsuccessful and is indicated on the <IS_ SUCCESS> child element of the <REQUEST_STATUS> element found in the <OUTDATA> structure. The <IS_ SUCCESS> contains a Boolean value (either true or false) indicating the status of the request. When the <IS_ SUCCESS> value is false, the Integration should indicate to the calling application the request was unsuccessful. Below is a sample Determination Response where the Request is not successful:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
        <taxCalculationResponse
xmlns="http://www.sabrix.com/services/taxcalculationservice/2011-09-01">
        <OUTDATA version="G">
        <REQUEST_STATUS>
        <IS_SUCCESS>false</IS_SUCCESS>
        <IS_PARTIAL_SUCCESS>false</IS_PARTIAL_SUCCESS>
        <ERROR>
        <CODE>COMPANY_NOT_FOUND</CODE>
        <DESCRIPTION>Unknown Company.</DESCRIPTION>
        <ERROR_LOCATION_PATH>/OUTDATA/INVOICE[INVOICE_NUMBER='155']</ERROR_
LOCATION_PATH>
        </ERROR>
        </REQUEST_STATUS
```

The values of <CODE>, <DESCRIPTION> elements within the <ERROR> structure provides further details about the root cause of the error.

## 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 tags are included in the message and the values in the request XML are correct. Here are some examples:

| AREA | SYMPTOM | SOLUTION |
|---|---|---|
| Address | Geocode tag is missing in the request XML | thomsonreuters.idt.address.postalcodeIncludesGeoCode should be set to false and<br><br>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<br><br>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 | Check properties starting with thomsonreuters.idt.address.county (or city or district).<br><br>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.<br><br>If it is a credit card payment, the bill-to address is searched in the PaymentInfo associated with the cart.<br><br>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.<br><br>If it is an account payment, the B2BUnit associated with the B2BCustomer is found and B2BUnit.billingAddress field is used to populate address fields.<br><br>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 B2BUnit.billingAddress field is used for customer name and id.<br><br>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. |

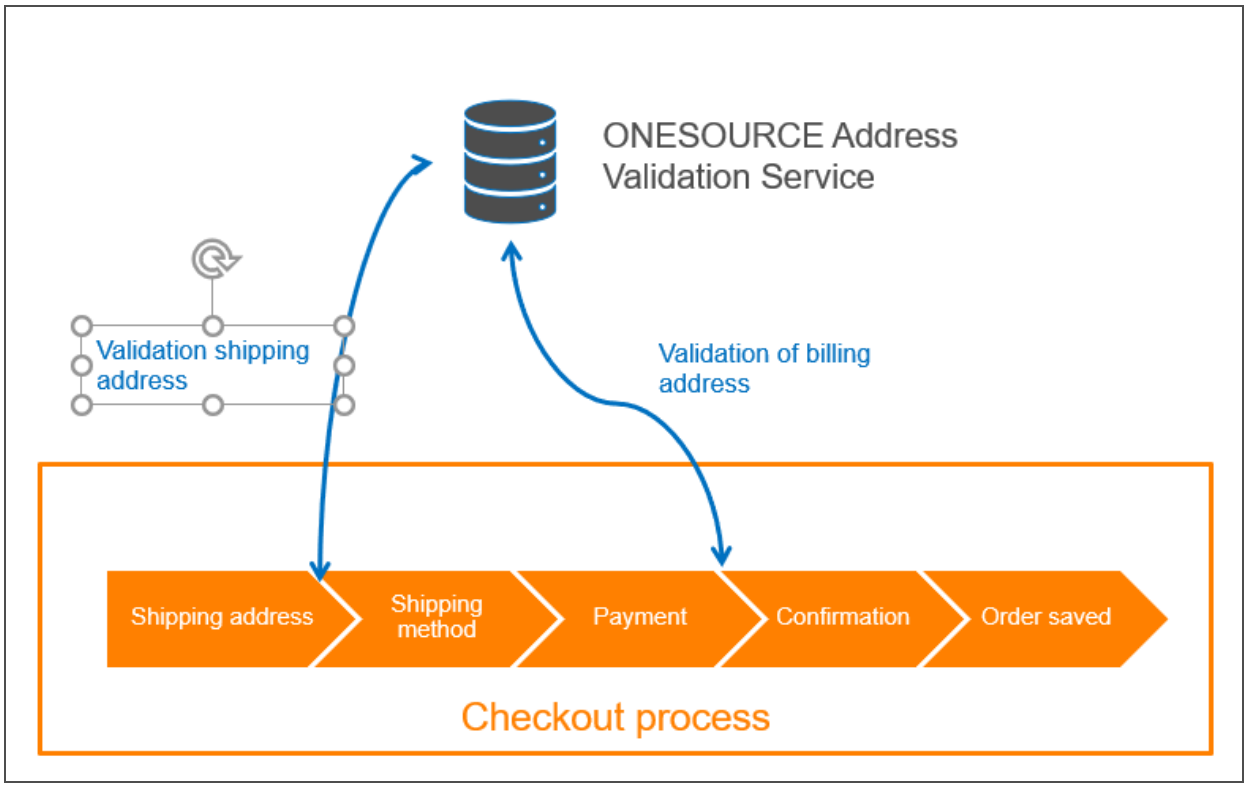
# ADDRESS VALIDATION SERVICE (AVS)

For external address validation call to ONESOURCE AVS to be triggered in the system, 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
- The user enters a US address in the storefront

# 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 in order 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 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:

| KEY | DESCRIPTION |
|---|---|
| thomsonreuters.avs.integration.active | If true, integration will be active<br>Default value: true |
| thomsonreuters.avs.addressvalidationservice.url | The value will be provided by Thomson Reuters.<br>Default value: Empty |
| thomsonreuters.avs.externalcompanyid | The value will be provided by Thomson Reuters<br>Default value: Empty |
| thomsonreuters.avs.password | The value will be provided by Thomson Reuters<br>Default value: Empty |
| thomsonreuters.avs.username | The value will be provided by Thomson Reuters<br>Default value: Empty |
| thomsonreuters.avs.detailedlogging.request | If true, the XML message for the request will be logged<br>Default value: false |

| | |
|---|---|
| thomsonreuters.avs.detailedlogging.response | If true, the XML message for the response will be logged<br><br>Default value: false |
| 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<br><br>Default value: false |
| thomsonreuters.avs.integration.connection.errormessage | Default value: Address Verification Service is down. |

## AVS UI Messages

In case of an error, ONESOURCE Address Validation Service returns an explanation text that describes the root cause.

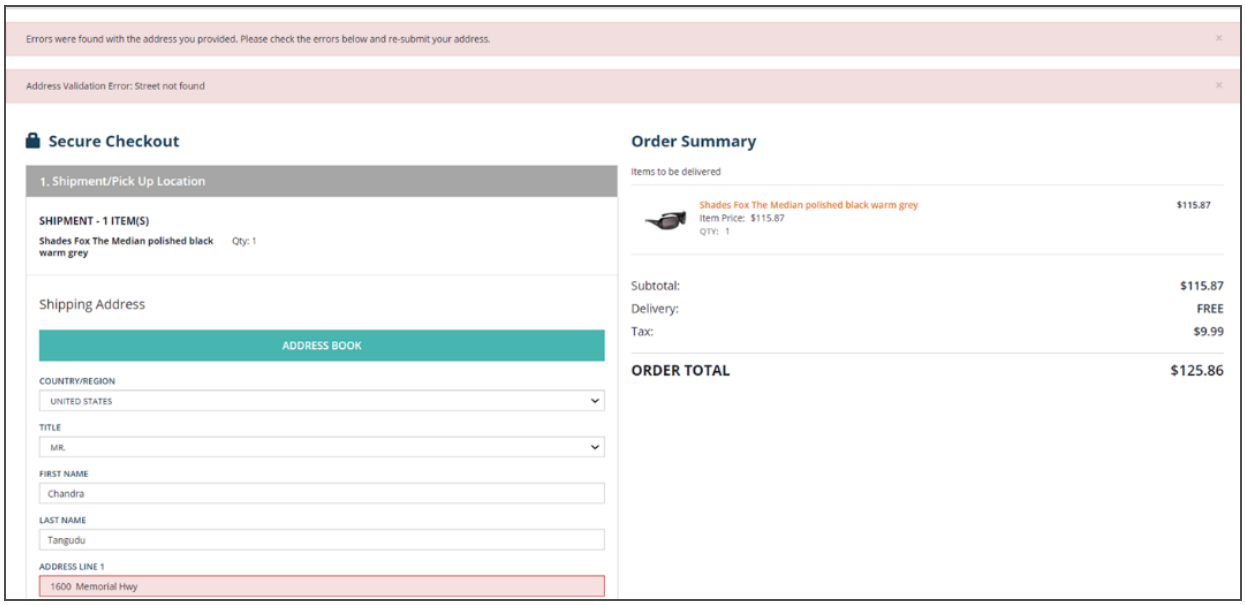
Here are some possible error messages that can be returned by the service:

- 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.

In order 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 is 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 | Log ID: {1}, Address Verification Service Request XML: {2}<br><br>{1}: Unique request ID that includes the cart id and the timestamp<br><br>{2}: Request XML<br><br>The message is only activated if thomsonreuters.avs.detailedlogging.request =true and DEBUG log level is activated for class TravsSoapLoggingService |
| 3 | INFO | TravsSoapLoggingService | Log ID: {1}, Address Verification Service Response XML: {2}<br><br>{1}: Unique request ID that includes the cart id and the timestamp<br><br>{2}: Response XML<br><br>The message is only activated if thomsonreuters.avs.detailedlogging.response =true and DEBUG log level is activated for class TravsSoapLoggingService |

| | | | |
|---|---|---|---|
| 4 | INFO | TravsAddressVerificationService | Request stats for AddressLine1: {1}, Response Waiting Duration: ##{2}##<br><br>{1}: ID for the AVS call<br><br>{2}: Time passed while waiting for the response from the external service |

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

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: DATA MAPPING

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.

| **ONESOURCE DETERMINATION** | **SAP COMMERCE** | |
|---|---|---|
| **Level** | **Data Element** | **Population Logic** |
| Request | | |
| Header - UsernameToken | Username | Configuration: thomsonreuters.idt.username |
| Header - UsernameToken | Password | Configuration: thomsonreuters.idt.password<br><br>Will be masked in logs |
| INDATA | version (attribute of INDATA) | Fixed value Default as 'G' |

| | | |
|---|---|---|
| INVOICE | EXTERNAL_ COMPANY_ID | Configuration: BaseStore.tridtExternalCompanyId or 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'Default is 'F' |
| INVOICE | COMPANY_ROLE | Fixed value Default is 'S' |
| INVOICE | CURRENCY_ CODE | AbstractOrder.currency.isocode |
| INVOICE | INVOICE_DATE | For carts: Get the current date<br>For Orders: Order.date<br>For returns: ReturnRequest.order.date |
| INVOICE | INVOICE_NUMBER | For carts: AbstractOrder.code+Timestamp (Timestamp is in yyyy.MM.dd.HH.mm.ss format)<br>For orders: AbstractOrder.code<br>For returns: ReturnRequest.code |

| | | |
|---|---|---|
| INVOICE | IS_AUDITED | true/false<br><br>For carts; always false<br><br>For orders; if thomsonreuters.idt.request.audit.auditafterdelivery = false or delivery status is SHIPPED →true, otherwise false<br><br>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<br><br>Only for return requests |
| INVOICE | ORIGINAL_ DOCUMENT_ID | Associated order for the return request<br><br>Only for return requests |
| INVOICE | ORIGINAL_ INVOICE_DATE | The date for the associated order<br><br>Only for return requests |
| INVOICE | TAX_ DETERMINATION_ DATE | The date for the associated order<br><br>Only for return requests |
| INVOICE | TRANSACTION_ TYPE | Fixed value Default as 'GS' |
| INVOICE | CUSTOMER_ NUMBER | In B2C: AbstractOrder.user.uid<br><br>In B2B: B2BUnit is found based on AbstractOrder.user and B2BUnit.uid is sent |
| INVOICE | CUSTOMER_NAME | Only valid in B2B scenarios<br><br>B2BUnit is found based on AbstractOrder.user and B2BUnit.name is sent |

| | | |
|---|---|---|
| INVOICE | DELIVERY_TERMS | From AbstractOrder.deliveryMode.code<br><br>Only valid in B2B scenarios<br><br>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<br><br>The AddressModel found will be the basis to fill the address fields described below |
| INVOICE | SHIP_TO | AbstractOrder.deliveryAddress<br><br>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<br><br>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<br><br>The AddressModel found will be the basis to fill the address fields described below |
| INVOICE | ORDER_ ACCEPTANCE | Check system property thomsonreuters.idt.request.address.sendorderacceptanceaddress<br><br>If the property is true, BaseStore.tridtOrderAcceptanceAddress will be sent.<br><br>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:<br>In B2C scenarios, will be the same as SHIP_TO.<br>In B2B scenarios, B2BUnit.contactAddress will be sent if found. If not, it will be the same as SHIP_TO.<br>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<br>If the property is true, BaseStore.tridtOrderAcceptanceAddress will be sent.<br>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:<br>In B2C scenarios, will be the same as BILL_TO.<br>In B2B scenarios, B2BUnit.contactAddress will be sent if found. If not, it will be the same as BILL_TO.<br>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 | STATEPROVINCE | Address.region.isoCodeShort<br>Only for Canadian addresses |
| ADDRESS | COUNTY | Address.(field)<br>field is determined based on thomsonreuters.idt.address.county |

| | | |
|---|---|---|
| ADDRESS | CITY | Address.(field)<br><br>field is determined based on thomsonreuters.idt.address.city |
| ADDRESS | DISTRICT | Address.(field)<br><br>field is determined based on thomsonreuters.idt.address.district |
| ADDRESS | POSTCODE | Address.postalcode<br><br>field value may be stripped based on thomsonreuters.idt.address.postalcodeIncludesGeoCode |
| ADDRESS | GEOCODE | Address.postalcode or Address.(field)<br><br>Please check Configuration Options for the following system properties:<br><br>thomsonreuters.idt.address.postalcodeIncludesGeoCode<br><br>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:<br><br>thomsonreuters.idt.request.item.sendCommodityCode<br><br>thomsonreuters.idt.request.item.taxcode.enhancedsearch.product hierarchy.enabled<br><br>thomsonreuters.idt.request.item.taxcode.enhancedsearch.categor yhierarchy.enabled<br><br>thomsonreuters.idt.request.item.taxcode.fallbackCode |
| LINE | PART_NUMBER | AbstractOrderEntry.product.code |
| LINE | DESCRIPTION | AbstractOrderEntry.product.name |

| | | |
|---|---|---|
| LINE | GROSS_AMOUNT | Used if AbstractOrder.net = true<br><br>AbstractOrderEntry.TotalPrice + AbstractOrderEntry.DiscountValues |
| LINE | GROSS_PLUS_ TAX | Used if AbstractOrder.net = false<br><br>AbstractOrderEntry.TotalPrice + AbstractOrderEntry.DiscountValues |
| LINE | DISCOUNT_ AMOUNT | Propoertionally distributed AbstractOrder.DiscountValues + AbstractOrderEntry.DiscountValues |
| LINE | AMOUNT | AbstractOrderEntry.quantity |
| LINE | UOM | AbstractOrderEntry.unit |
| LINE | SHIP_TO | AbstractOrderEntry.deliveryPointOfService.address or AbstractOrderEntry.deliveryAddress<br><br>Only if it is a pickup item or if the item delivery address is not empty |
| LINE | SHIP_FROM | AbstractOrderEntry.deliveryPointOfService.address<br><br>Only if it is a pickup item |
| LINE | DELIVERY_TERMS | AbstractOrderEntry.deliveryMode.code<br><br>Only valid in B2B scenarios<br><br>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<br><br>Please check Configuration Options for the following system properties:<br><br>thomsonreuters.idt.request.item.sendCommodityCode<br><br>thomsonreuters.idt.request.shipping.productcode |
| LINE (DELIVERY) | GROSS_AMOUNT | Used if AbstractOrder.net = true<br><br>AbstractOrder.deliveryCost or a calculated value if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=true |
| LINE (DELIVERY) | GROSS_PLUS_ TAX | Used if AbstractOrder.net = false<br><br>AbstractOrder.deliveryCost or a calculated value if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems=true |
| LINE (DELIVERY) | OVERRIDE_ AMOUNT | With Country, State, County, City, District, Postal code breakdown<br><br>Used only if the delivery cost is being refunded in return request |
| LINE (DELIVERY) | AMOUNT | Fixed value Default as '1' |
| LINE (DELIVERY) | RELATED_LINE_ NUMBER | Associated entry number (Only if thomsonreuters.idt.request.shipping.sendmultipledeliverycostitems)<br><br>Not used for Returns |
| LINE (DELIVERY) | USER_ ELEMENT/ATTRIB UTE1 | Fixed value Default is 'X'<br><br>Flag to differentiate shipment lines from product lines, used in response processing |
| Response | | 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<br><br>ONESOURCE field is selected based on thomsonreuters.idt.response.summarization |
| INVOICE/LINE/TAX | TAX_AMOUNT/AUTHORITY_AMOUNT | Value<br><br>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<br><br>Always false. SAP Commerce does not accept rates as external tax |
| INVOICE/LINE/TAX | TAX_AMOUNT/AUTHORITY_AMOUNT | appliedValue<br><br>Same as value |
| INVOICE | CURRENCY_CODE | currency |
| INVOICE/LINE/TAX | AUTHORITY_NAME / ERP_TAX_CODE / AUTHORITYTYPE / EFFECTIVEZONELEVEL | Code<br><br>ONESOURCE field is selected based on thomsonreuters.idt.response.summarization, thomsonreuters.idt.response.shipping.prefix is added as a prefix to the determined value. |
| INVOICE/LINE/TAX | TAX_AMOUNT/AUTHORITY_AMOUNT | Value<br><br>Value is calculated based on thomsonreuters.idt.response.summarization. If summarization is requested several TAX_AMOUNT/AUTHORITY_AMOUNTs are summed up based on the summarization level |

| | | |
|---|---|---|
| INVOICE/LINE/TAX | - | Absolute<br><br>Always false. SAP Commerce does not accept rates as external tax |
| INVOICE/LINE/TAX | TAX_AMOUNT/AUTHORITY_AMOUNT | appliedValue<br><br>Same as value |
| INVOICE | CURRENCY_CODE | currency |

# APPENDIX 2: AVS DATA MAPPING

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.

| ONESOURCE DETERMINATION | | SAPCOMMERCE |
|---|---|---|
| **Level** | **Data Element** | **Attribute** |
| Request | | |
| Header - UsernameToken | Username | Configuration: thomsonreuters.avs.username |
| Header - UsernameToken | Password | Configuration: thomsonreuters.avs.password<br><br>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 |

| Response | | |
|---|---|---|
| RequestStatus | Status | Status of the call<br>SUCCESS or ERROR |
| AddressResponse | Status | Status of the validation<br>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 |