# vintrace
# Point of Sales
# Integration Guide

vintrace

# Introduction

This guide describes how to use the vintrace REST API to provide your point of sales (POS) or eCommerce system's sales and refund data to vintrace.

Detailed REST API documentation can be found at https://vintrace.stoplight.io/.

NOTE: A REST API that allows your POS/eCommerce system to query available stock levels is also available.

# Authentication

The vintrace API uses either basic or token HTTP authorization for REST requests.

## Basic Authorization

Basic authorization uses a Base64-encoded username and password separated by a colon (:) and added to an authorization header. For example, if your username is *myusername* and your password is *mypassword*, you would need to send the following header.

```
Authorization: Basic bXl1c2VybmFtZTpteXBhc3N3b3Jk
```

Where the text after Basic is the Base64 representation of *myusername:mypassword*

## Token HTTP Authorization

This still uses the authorization header, but with a different scheme. Refer to Managing API Tokens for details on generating your API token.

```
Authorization: Bearer a0249fb3-3b46-47a8-99f2-619a61ac0eea
```

The value after Bearer is the API token generated by vintrace.

## Date Formats

The date is specified as the number of milliseconds that have elapsed since epoch (i.e., January 1, 1970). For example, UTC Sunday, August 27, 2017 14:00:00 is represented as 1503842400000.

## External References

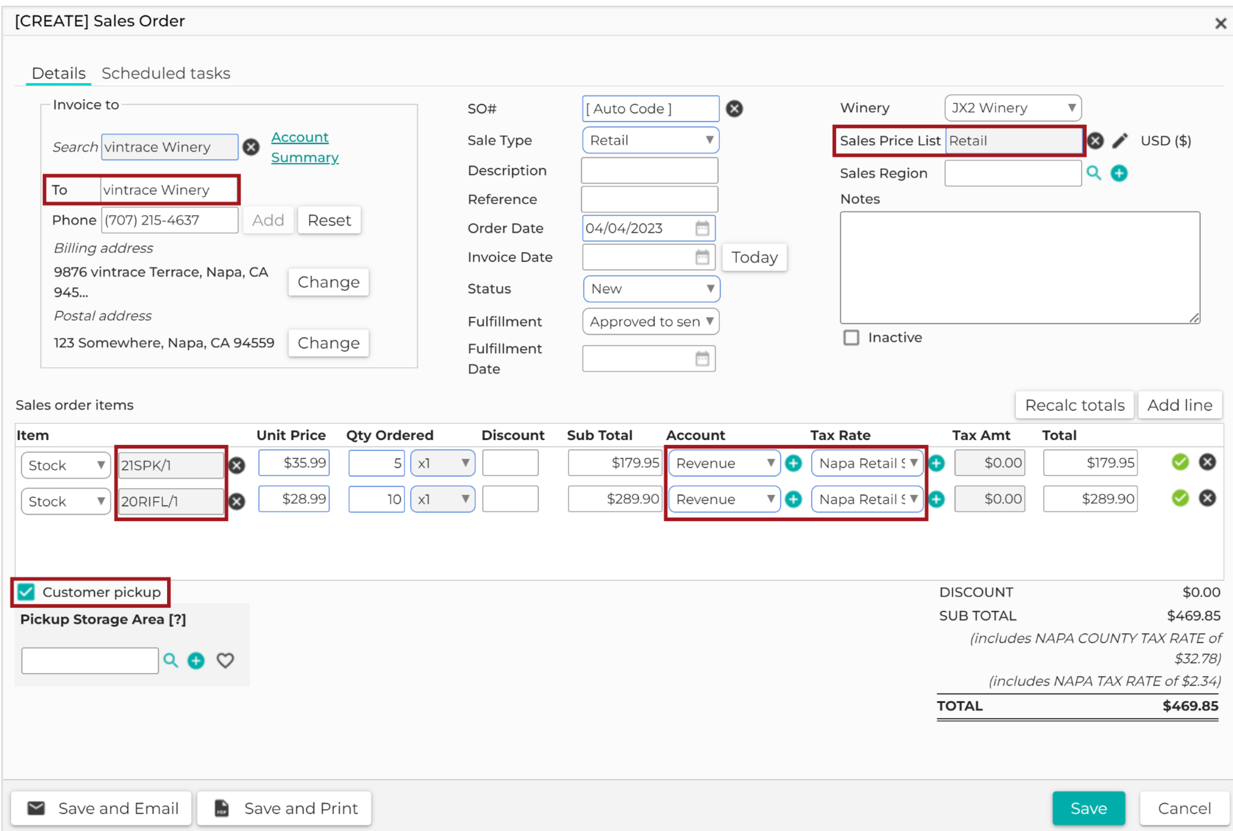
There are fields in the vintrace API that allow you to supply either the vintrace internal ID (e.g., customerId) for a record, or the human readable name (e.g., customerName). For these fields, we recommend using the vintrace internal ID if your software supports it as this is less likely to change than the corresponding customer name.

# Sales and Refunds

vintrace uses sales and refund information to track industry-specific taxes such as WET and NZ excise, as well as depleted stock levels from dedicated storage areas based on POS and eCommerce sales transactions.

## Sales

A sales order lists the items that have been sold to a customer.




Each sales order is associated with a price list. The price list determines the prices used for items. The price list in the sales order provides flexibility to charge different amounts for your goods based on an individual customer, vintage, or sales type (e.g., retail or wholesale).

Each sales order line item represents each individual sales item. You can configure the revenue account and tax rate per item. The revenue account tracks the revenue generated out of the sales. It can be configured from the price list, or it can be provided for the individual items in the sales order. Tax rates can be configured from price lists, or they can be provided for the individual items in the sales order.

We recommend that you group your sales transactions per customer per outlet per day. Sales without any customer data can be grouped under a general customer. For example, WALKIN.

## Special Fields in a Sales Order

### customerPickup

This field indicates that the sale is a customer pickup. For example, a sale from the cellar door where the customer walks out with the goods. If this option is on, vintrace automatically reduces stock levels for the items in the sales order once it's saved and approved.

### storageAreaId/storageAreaCode

When the customerPickup field is set to *true*, you must enter a storage area name or the vintrace ID from where the stock levels are reduced for the sale.

### disableAccountsSync

If this is set to *true*, it disables the syncing of the sales order with any accounting integrations. For example, if you have Xero accounting integration enabled, you cannot sync this sales order to Xero.

### customerId/customerName

The person or customer who bought the stock. You can enter the customer's vintrace ID, or the name as it's saved in vintrace.

### salesPriceListId/salesPriceListName

The price list determines prices for the item. You can configure a price list vintrace specifically for the cellar door.

### itemId/itemName

The name or ID of the stock item in vintrace.

### accountId/accountName

The revenue account tracks the revenue generated on the sale. You can enter the vintrace ID of the revenue account, or the name as it's saved in vintrace.

This is an optional field. If no value is entered, it defaults to the revenue account on the sales item/stock item category/sales region/price list of the item. You can enter the vintrace ID of the price list, or its name.

### taxRateId/taxRateName

Different tax rates can apply to different situations for items. For example, in Australia, the Wine Equalisation Tax (WET) applies to wine sales, but not to the merchandise which can also be on the same price list. You can configure a tax rate for wine sales that has a WET tax component and a different tax rate for merchandise sales where WET tax is not applied.

This is an optional field. If no value is entered, it defaults to the tax rate of the price list used. You can enter the vintrace ID of the tax rate, or its name in vintrace.

### discountPct/adjustment

For sales orders with discounts, and when accounting integration with Xero is enabled for the customer, enter the discount per item as a percentage in the discountPct field.

Otherwise, if accounting integration is disabled, enter the discount as a dollar value using the adjustment field.

## Creating or Updating a Sales Order

To update a sales order, you must provide the ID of the sales order that you want to update. Otherwise, a new sales order will be created.

When you want to deplete stock for a sale, you'll need to set the following:

- customerPickup - customerPickup should be set to *true*
- storageAreaCode - The location where the stock is to be depleted. If no value is specified, the API user's favorite pickup storage area will be used.

If you don't want to send the sales order to a third-party accounting integration such as Xero, set disableAccountsSync to *true*.

| POST http://<base_api_url>/api/v6/sales-order | | |
|---|---|---|
| Field<br>* indicates field is mandatory | Type | Description |
| id | Integer | The sales order to update. Otherwise, a new sales order is created. |
| code | String | Unique name for the sales order. If not provided, it is auto generated. |
| customerId * | Integer | Provide either the vintrace ID, or vintrace customer name. |
| customerName * | String | |
| orderDate * | Date | Order date |
| invoiceDate | Date | Invoice date |
| salesType | String | Possible values:<br>● Retail<br>● Wholesale<br>● Staff |
| salesPriceListName * | String | Price list to be used for the sales order. You can provide either the name, or the ID. |
| salesPriceListId * | Integer | |
| salesOrderStatus | String | Possible values:<br>● New<br>● Approved<br>● Payment in progress<br>● Paid |
| description | String | |

| | | |
|---|---|---|
| reference | String | |
| wineryId | Integer | Provide the winery ID or name. |
| wineryName | String | |
| fulfillment | String | Auto resolved as *Fully sent*. |
| fulfillmentDate | Date | The date that the stock is reduced or dispatched. |
| salesRegionId | Integer | Provide either the sales region ID, or name. |
| salesRegionCode | String | |
| notes | String | |
| customerPickup | Boolean | Set as *true* if stock levels are to be reduced. |
| disableAccountsSync | Boolean | Set as *true* if you do not want to sync the sales order to an accounting integration (e.g., Xero). |
| ignoreStockError | Boolean | To ignore error/warning on insufficient stock. |
| storageAreaId | Integer | Provide either the storage ID, or name. Otherwise, this is resolved as the default storage area configured in vintrace.<br><br>This field is mandatory if the customerPickup field is set to *true* and there is no default storage area configured in vintrace. |
| storageAreaCode | String | |
| salesOrderItems * | Array | Array of items in the sales order |
| itemName * | String | The vintrace stock item code, or ID. |

| | | |
|---|---|---|
| itemId * | Integer | |
| unitPrice * | Integer | Unit price of the item. |
| quantity * | Integer | Quantity sold. |
| taxRateId | Integer | Auto resolved from the price list. If you know the vintrace ID of the tax rate, you can provide it. |
| taxRateName | String | Auto resolved from the price list. If you know the vintrace code of the tax rate, you can provide it. |
| accountId | Integer | Revenue account. |
| accountCode | String | Revenue account. |
| discountPct | Integer | For a sales order with discounts and if accounting integration is enabled for the customer, enter the discount as a percentage value. |
| adjustment | Integer | If accounting integration is disabled, enter the discount adjustment as a dollar amount. |

## Example

```
{
      "customerName": "ABC Wine Company",
      "salesPriceListName": "Retail",
      "salesType": "Retail",
      "salesOrderStatus": "Approved",
      "invoiceDate": 1507860000000,
      "customerPickup": true,
      "storageAreaCode": "Warehouse",
      "disableAccountsSync": true,
      "salesOrderItems":
      [
      {
```

```
            "itemName": "2013 Gloria GSM 750ml",
            "unitPrice": 13.223,
            "quantity": 5,
            "accountCode": "Revenue"
        },
        {
            "itemName": "2014 BSE Cabernet Sauvignon 750ml",
            "unitPrice": 15.5,
            "quantity": 10,
            "discountPct": 10,
            "accountCode": "Revenue"
        }
        ],
        "orderDate": 1507860000000,
        "ignoreStockError": true
}
```

## Viewing Sales Orders

You can view the created sales order using the following endpoints:

- All sales orders: http://<base_api_url>/api/v6/sales-orders/list/
- Sales order by ID: http://<base_api_url>/api/v6/sales-orders/<id>
- Sales order by code: http://<base_api_url>/api/v6/sales-orders/?code=<code>

### Query Options

You can add the following parameters to the above endpoints to filter results.

| | | |
|---|---|---|
| startDate | Format yyyy-MM-dd | Order start date. |
| endDate | Format yyyy-MM-dd | Order end date. |
| invStartDate | Format yyyy-MM-dd | Invoiced start date. |
| invEndDate | Format yyyy-MM-dd | Invoiced end date. |
| startsWith | String | Sales order codes that begin with the string. |
| status | | Sales orders with the given status. |
| customerName | | Sales orders with the given customer account. |

| | | |
|---|---|---|
| first | Integer | Fetches first n results. |
| max | Integer | Maximum results fetched. |

For more details, see the vintrace API workspace.

## Refunds

You can use the vintrace API to process refunds, return stock, and create a refund for any approved sales order.

You can create multiple refunds for a sales order. Each line item in the sales order corresponds to an equivalent refund line item. When creating the refund, you'll need to:

- Set the sales order name or ID
- Specify the item name
- Specify the return quantity per refund line item

The unit price in teh refund line item is automatically calculated from the corresponding sales order line item. If a discount is specified for the line item in the sales order, the unit price in the refund uses the discounted price.

The following must be specified for a refund:

- salesOrderName - The sales order on which the return is processed
- refundDate - The refund date.
- refundLineItems - The list of items to be returned. Specify itemName or itemID, and returnQuantity.

## Creating or Updating a Refund

To update a refund, you must provide the ID of the refund that you want to update. Otherwise, a new refund will be created.

| POST http://<domain_url>/api/v6/refund | | |
|---|---|---|
| Field<br>* indicates field is mandatory | Type | Description |

| | | |
|---|---|---|
| id | Integer | If provided, the refund is updated. |
| code | String | Unique name for refund. Auto generated if not entered. |
| refundDate * | Date | Date of the refund |
| salesOrderName * | String | The sales order you are creating a refund for. This is mandatory if you are creating a refund. |
| stockReturned | Boolean | Set as *true* if the stock is returned. |
| storageAreaId | Integer | Provide either the vintrace ID or name of the storage area to which stock is returned. |
| storageAreaName | String | |
| refundStatus | String | Possible values :<br>• *Approved*<br>• *Awaiting approval* |
| disableAccountsSync | Boolean | Set as *true* if you do not want to sync the refund to an accounting integration (e.g., Xero). |
| reference | String | |
| notes | String | |
| refundLineItems * | Array | |
| itemName | String | The vintrace code for the item. |
| returnQuantity | Integer | |

## Example

```
{
      "refundDate": 1507860000000,
      "salesOrderName": "JXSO234",
      "refundStatus": "Awaiting approval",
      "notes": "Created from API",
      "disableAccountsSync": true,
      "stockReturned": true,
      "refundLineItems":
      [
      {
            "itemName": "2013 Gloria GSM 750ml",
            "returnQuantity": 5
      },
      {
            "itemName": "2014 BSE Cabernet Sauvignon 750ml",
            "returnQuantity": 10
      }
      ]
}
```

# Viewing Refunds

You can view the created refunds using the following endpoints:

- All refunds: http://<base_api_url>/api/v6/refund/list/
- Refund by ID: http://<base_api_url>/api/v6/sales-orders/<id>
- Refund by code: http://<base_api_url>/v6/sales-orders/?code=<code>

## Query Options

You can add these parameters to the endpoints above to filter the results:

| | | |
|---|---|---|
| startDate | Format yyyy-MM-dd | Refund start date. |
| endDate | Format yyyy-MM-dd | Refund end date. |
| startsWith | String | Refund codes that begin with the string. |
| status | | Refunds with the given status. |
| customerName | | Refunds with the given customer account. |

| | | |
|---|---|---|
| first | Integer | Fetches first n results. |
| max | Integer | Maximum results fetched. |

For more details, see the vintrace API workspace.

# Inventory

You can get available stock levels using the vintrace API.

If you want to view the breakdown of costs associated with the stock items:

- Set the breakoutCosting field to *true*
- The API user needs to have the *Can view costs* permission.

GET: http://<base_api_url>/api/v6/inventory

## Query options

| | | |
|---|---|---|
| startDate | Format yyyy-MM-dd | Refund start date. |
| endDate | Format yyyy-MM-dd | Refund end date. |
| startsWith | String | Refund codes that begin with the string. |
| status | | Refunds with the given status. |
| customerName | | Refunds with the given customer account. |
| first | Integer | Fetches first n results. |
| max | Integer | Maximum results fetched. |
| date | Format yyyy-MM-dd | The date to report stock up to, excluding stock changes after it. |
| stock | String | The code of the stock item. |
| stockType | String | The stock type of the stock items. |
| ownerName | String | The owner's name on the stock items. |
| showEquivalentType | | Displays the ratio of |

| | | |
|---|---|---|
| | | what the inventory item's volume/qty is equivalent to the given showEquivalentType. |
| breakoutCosting | | If true, retrieves each costing's category of costs and details each type's cost. The user must have the *Can View Costs* permission. |
| disableCommitHeaders | | If true, the item's committed stock amount details are not displayed. |

For more details, see the vintrace API workspace.

## Example

Get Inventory levels for 2012 BSE Merlot 750ml

```
GET :
http://<base_api_url>/api/v6/?breakoutCosting=true&stockType=Single
x1&stock=2012 BSE Merlot 750ml
```


```
{
      "status": "Success",
      "message": null,
      "inventorySummaries":
      [
      {
            "date": 1508125708144,
            "dateAsText": "16/10/2017",
            "winery": "King Estate Winery",
            "quantity": 58,
            "committed": 0,
            "onOrder": 0,
            "available": 58,
            "unit": "Single x1",
            "type": "Single x1",
            "code": "2012 BSE Merlot 750ml",
            "name": "2012 BSE Merlot bottle",
            "description": "",
            "lotBatch": null,
```

```
        "lotExpiryDate": null,
        "lotExpiryDateAsText": null,
        "lotManufactureDate": null,
        "lotManufactureDateAsText": null,
        "reorderCode": null,
        "location": "Main Warehouse / Main Warehouse",
        "stockCategory": "",
        "price": null,
        "priceAsText": null,
        "taxClass": null,
        "taxState": null,
        "bond": null,
        "sizeRatio": null,
        "finalProducts": "",
        "vintage": null,
        "variety": null,
        "region": null,
        "program": null,
        "owner": "JX2 Winery",
        "unitCost": 11.22,
        "unitCostAsText": "$11.22",
        "totalCost": 651.02,
        "totalCostAsText": "$651.02",
        "fruitCost": 0,
        "fruitCostAsText": "$0.00",
        "bulkCost": 0,
        "bulkCostAsText": "$0.00",
        "additiveCost": 0,
        "additiveCostAsText": "$0.00",
        "operationCost": 0,
        "operationCostAsText": "$0.00",
        "packagingCost": 0,
        "packagingCostAsText": "$0.00",
        "storageCost": 295.92,
        "storageCostAsText": "$295.92",
        "overheadCost": 355.1,
        "overheadCostAsText": "$355.10",
        "freightCost": 0,
        "freightCostAsText": "$0.00",
        "otherCost": 0,
        "otherCostAsText": "$0.00",
        "sku": "10071"
    },
    {
        "date": 1508125708144,
        "dateAsText": "16/10/2017",
        "winery": "JX2 Winery",
        "quantity": 40,
        "committed": 0,
        "onOrder": 0,
        "available": 40,
        "unit": "Single x1",
```

```
            "type": "Single x1",
            "code": "2012 BSE Merlot 750ml",
            "name": "2012 BSE Merlot bottle",
            "description": "",
            "lotBatch": null,
            "lotExpiryDate": null,
            "lotExpiryDateAsText": null,
            "lotManufactureDate": null,
            "lotManufactureDateAsText": null,
            "reorderCode": null,
            "location": "Barrel Shed / CELLAR",
            "stockCategory": "",
            "price": null,
            "priceAsText": null,
            "taxClass": null,
            "taxState": null,
            "bond": null,
            "sizeRatio": null,
            "finalProducts": "",
            "vintage": null,
            "variety": null,
            "region": null,
            "program": null,
            "owner": "JX2 Winery",
            "unitCost": 11.22,
            "unitCostAsText": "$11.22",
            "totalCost": 448.98,
            "totalCostAsText": "$448.98",
            "fruitCost": 0,
            "fruitCostAsText": "$0.00",
            "bulkCost": 0,
            "bulkCostAsText": "$0.00",
            "additiveCost": 0,
            "additiveCostAsText": "$0.00",
            "operationCost": 0,
            "operationCostAsText": "$0.00",
            "packagingCost": 0,
            "packagingCostAsText": "$0.00",
            "storageCost": 204.08,
            "storageCostAsText": "$204.08",
            "overheadCost": 244.9,
            "overheadCostAsText": "$244.90",
            "freightCost": 0,
            "freightCostAsText": "$0.00",
            "otherCost": 0,
            "otherCostAsText": "$0.00",
            "sku": "10071"
        }
        ]
}
```

# Customer

Customer refers to the purchaser of your stock in the sales order. A customer can either be an organisation or an individual.

To create a customer as an individual

- Set the isOrganization field to *false*.
- Set the primeName field to the surname (family name) of the person.
- Set the givenName field to the first (given) name of the person.

To create a customer as an organisation

- Set the  isOrganization field to *true*.
- Set the primeName field to the organisation's name.
- Set the givenName field to *null*.

## Creating or Updating a Customer

To update a customer, you must provide the ID of the customer that you want to update. Otherwise, a new customer will be created.

| POST http://<domain_url>/api/v6/party | | |
|---|---|---|
| Field<br>* indicates field is mandatory | Type | Description |
| ID | Integer | If an ID is provided, it updates the party |
| primeName * | String | If creating/updating an individual, this is the surname/family name. Otherwise, this is the name of the organisation. |
| givenName * | String | This is mandatory if the party is an individual and should be set to the first (given) name of the individual. Otherwise, if the party |

| | | |
|---|---|---|
| | | is an organisation, this can be set to *null*. |
| phone | String | Phone number |
| email | String | Email address |
| address | Address | |
| street1 | String | Street address line 1 |
| street2 | String | Street address line 2 |
| city | String | City |
| state | String | State |
| postalCode | String | Postal code |
| country | String | Country |
| isOrganization | Boolean | If the party is an organistion, set to *true*. Otherwise, if the party is an individual, set to *false*. |

## Example

```
{
	"primeName": "Tim",
	"givenName": "Smith",
	"email": "timsmith@abc.com",
	"phone": "(03) 8337 0444 ",
	"address":
	{
		"street1": "PO Box 5124",
		"street2": null,
		"city": "Napa",
		"state": "CA",
		"postalCode": "94558",
```

```
            "country": "USA"
      },
      "isOrganization": false
}
```

## Viewing Customers

You can view the created customer using the following endpoints:

- All customer: http://<base_api_url>/api/v6/party/list/
- Customer by id: http://<base_api_url>/api/v6/party/<id>
- Customer by name: http://<base_api_url>/v6/party/?name=<name>

### Query Options

| | | |
|---|---|---|
| first | Integer | Fetches the first n results. |
| max | Integer | Maximum results fetched. |
| startsWith | String | String that matches the start of the primeName field. For individuals, this is the Surname field in vintrace, and the Company Name field for organisations. |
| category | | The category of the party:<br>• *All*<br>• *Individuals*<br>• *Organisations* |