Tax Inclusive Pricing

The intent of this feature is to represent orders with taxes already included in the prices of each line item. I will refer to Tax Inclusive Pricing as TIP and the historical Tax Exclusive Pricing as TEP throughout this document.

Key Concepts

1. This functionality will be added to the Cova platform through the introduction of new properties and APIs to preserve existing behavior.
2. Tax Inclusive Pricing will be a feature flag set at the location level. This allows companies the flexibility to have some locations setup with TIP while others can be setup as TEP.
3. Pricing APIs will represent either TIP or TEP prices depending on this feature flag. We expect this to change infrequently for any given location. There will be no indication from the existing pricing APIs to indicate what type of price is being returned.
4. There will be changes to Tier Pricing. New properties will allow the user to set the line price of a product at the tier quantity. This property should be used to determine the per/unit price
5. The previous order models used a per/unit pricing model where prices for each line item were entered as a per/unit price. A new order endpoint will use a line pricing model, where the price is the total due for the full quantity of the line item.

Pricing endpoints

Once TIP pricing is enabled for a location, the retailer will be expected to update existing prices to include taxes for that location. This means any display of these properties where taxes are calculated and added to the display price will be incorrect behavior once TIP is enabled and prices are updated.

Tier pricing changes

You'll see the addition of AtTierPrice for tier prices. Tier prices are prices that become enabled once a sufficient quantity of the product has been added to the cart. To some degree, they are like a bulk discount. For prices that have an `AtTierPrice` attribute, this should be used instead of the `RegularPrice` field. The intention is that if the quantity being sold is exactly the `TierQuantity`, it will be the final price for the line. Otherwise, you will need to calculate a line price by taking the `AtTierPrice`, dividing it by the `TierQuantity` and multiplying it by the actual quantity included on the order. There will be rounding differences between the two approaches, especially when the selling quantity is exactly the same as the Tier Quantity. The same principle applies to the `AtTierOverridePrice` and the `OverridePrice`

eg.

The Quantity of a tier is 3

The `AtTierPrice` is $1.00
The `RegularPrice` will be $0.33

If the quantity being sold is exactly 3, the Line price should be exactly $1.00. Using the RegularPrice approach, you would arrive at $0.33 X 3 = $0.99.

If the quantity being sold is exactly 4, you would calculate the Line price as
(AtTierPrice / 3) X 4
= ($1.00 / 3) X 4
= $1.33333 and round half up to $1.33.

Using the RegularPrice approach you would have arrived at $0.33 X 4 = $1.32

```
/v1/Companies(94447)/Entities(94447)/CatalogItems

returns collection of

{
        "Id": 647007616,
        "CompanyId": 94447,
        "EntityId": 94447,
        "CatalogItemId": "4ef0f176-b5a4-4e28-aa7d-2de8dfa09367",
        "PricingTermId": null,
        "RegularPrice": 7.50,
        "AtTierPrice": 15.00, 🡐 this is new!
        "IsDiscountable": false,
        "FloorPrice": null,
        "OriginalPrice": null,
        "PricingTierId": 119,
        "PricingGroupId": null,
        "PricingShelfId": null,
        "OverridePriceId": 2605961,
        "OverridePrice": 7.00,
        "AtTierOverridePrice": 14.00, 🡐 this is new!
        "OverrideStartDateUtc": null,
        "OverrideStopDateUtc": null,
        "LastModifiedDateUtc": "2019-07-01T21:22:59.023"
    }
```

```
/v1/Companies(94447)/PricingTiers

No actual pricing.  No changes.
```

Order Intake Endpoints

There are 2 new endpoints: to take advantage of tier pricing changes described above, for tax exclusive orders (TEP) use:

```
TEP OrderEndpoint
/covaorderintake/v1/TEPCovaOrder({{SalesOrderId}})
```

and for tax inclusive orders (TIP) use:

```
TIP OrderEndpoint
/covaorderintake/v1/TIPCovaOrder({{SalesOrderId}})
```

The new TEP/TIP order endpoints are based upon the v2 endpoint as described below. Please note the differences between v1 and v2 as they are prior to the introduction of TIP.

Current endpoints:

POST /covaorderintake/v1/CovaOrder
or PUT /covaorderintake/v2/CovaOrder({orderId})

```
{
  "Id": {{orderId}}, 🡐 only in v2
  "IntegratorId": "{{IntegratorId}}", 🡐 only in v2
  "CompanyId": {{CompanyId}},
  "EntityId": {{EntityId}},
  "FirstName": "{{FirstName}}",
  "LastName": "{{LastName}}",
  "customerId": "{{CustomerId}}",
  "IsDelivery": "{{IsDelivery}}",
  "Items": [
    {
      "Name": "{{ItemName}}",
      "CatalogItemId": "{{CatalogItemId}}",
      "Quantity": {{Quantity}},
      "SellAtPricePerUnit": {{SoldAtPrice}},
      "PackageId": "{{OptionalPackage}}"
    }
  ]
}
```

NOTE the request allows optional discounts and tax collections after the Items collection:

```
"Discount": {
    "InvoiceLevel": [
      {
        "Name": "Test",
        "Value": 1,
        "ValueType": "Amount"
      }
    ]
  },

"Taxes": [
  {
   "TaxId": "3236dc55-78e6-4bfb-9582-85d57ff15347",
   "TaxAmount": 1.11
  },
  {
   "TaxId": "d3f966df-215b-4a1d-adb5-876e0a0f1565",
   "TaxAmount": 2.22
  }
 ]

Both new endpoints expect properties as described below
TIP/TEP OrderEndpoints
PUT/covaorderintake/v1/TEPCovaOrder({{SalesOrderId}})
or PUT/covaorderintake/v1/TIPCovaOrder({{SalesOrderId}})
{
  "Id": "{{SalesOrderId}}",
  "CompanyId": {{CompanyId}},
  "EntityId": {{EntityId}},
  ~"~~FirstName~~": "{{FirstName}}",~ <- Removed
  ~"~~LastName~~": "{{LastName}}",~ <- Removed
  "OrderReference": "{{OrderReference}}
  "customerId": "{{CustomerId}}",
  "IntegratorId": "{{IntegratorId}}",
  "IsDelivery": "{{IsDelivery}}",

  ~"~~Items~~": []~ <- Renamed to Lines

  "Lines": [
    {
      "Name": "{{ItemName}}",
      "CatalogItemId": "{{CatalogItemId}}",
      "Quantity": {{Quantity}},
      ~"~~SellAtPricePerUnit~~": {{SoldAtPrice}},~ <- Removed
      "LineDollarAmount": {{LineDollarAmount}}, <- Added
      "PackageId": "{{OptionalPackage}}"
    }
  ]

  ~"~~Taxes~~": [],~ <- Removed
  ~"~~Discount~~: []~ <- Removed
}
```

Notes

- \`OrderReference\` this property replaces \`FirstName\` and \`LastName\`. It represents the text shown within Cova to identify an order. This is what the budtender will be presented with when attempting to tender the order within Cova. It is recommended that an appropriate scheme such as \`"$FirstName $LastName"\` be used. If a valid customerId is supplied for the order, the customer name is displayed on POS *instead* of the OrderReference.
- LineDollarAmount is the total amount for the line item instead of the price for each unit.
  Example:
  ```
    "Quantity": 3,
    "LineDollarAmount": 10.0
  ```
  instead of
  ```
    "Quantity": 3,
    "SellAtPricePerUnit": 3.33
  ```
- There is the introduction of an IntegratorId which must be provided by Cova. Anyone using the current v2 CovaOrder endpoint will have this already.
- The integrator must generate the idempotent request SalesOrderId (GUID) for each request
- The HTTP verb is changed to PUT to better capture the request ID being generated externally
- Response from both TIP and TEP order endpoints return a 202 – Accepted and an empty response body.