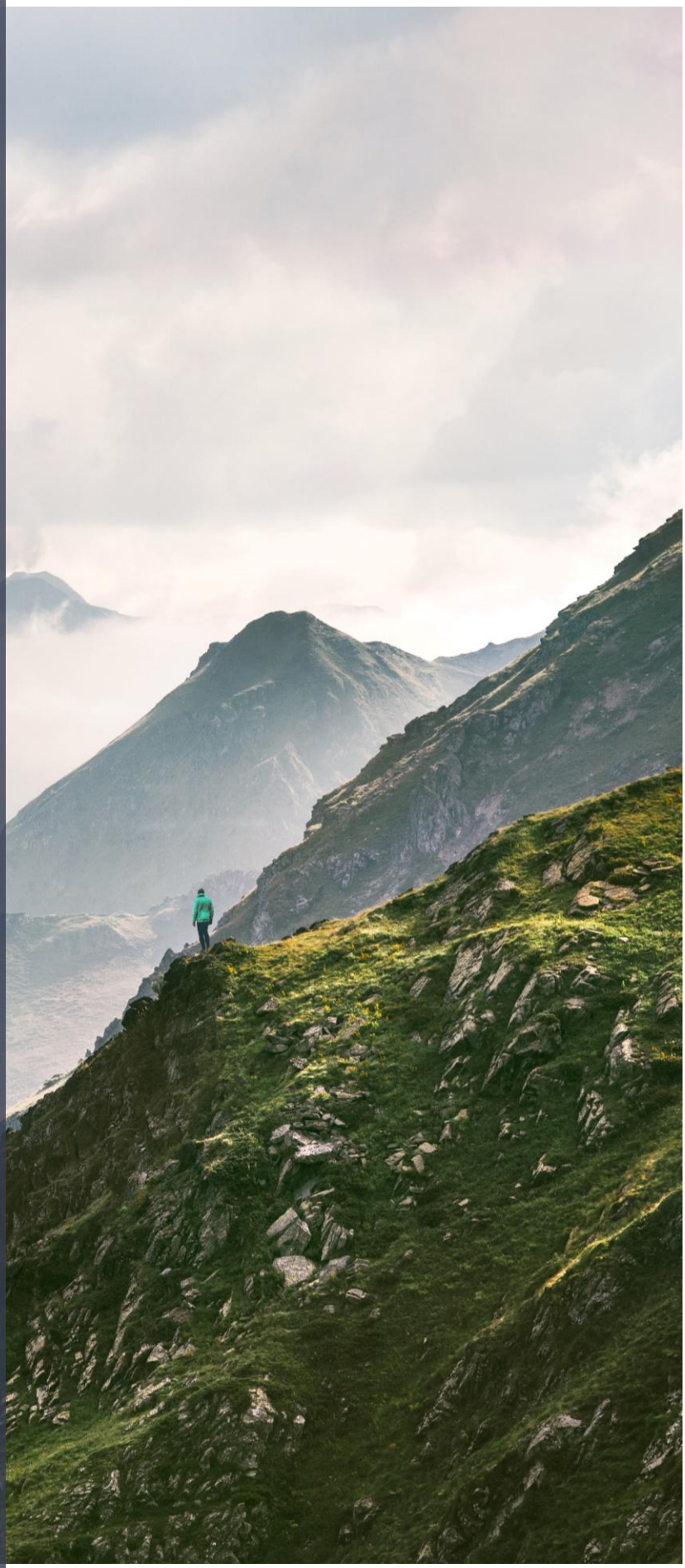
COVA™

CANNABIS RETAIL SOFTWARE

Sales Orders

Explanations and examples

2026



COPYRIGHT BY COVA SOFTWARE©

Cova Sales Orders API

The Cova Sales Order APIs are a way to push simple external orders to the Cova system. The most common audience is an Ecommerce provider showing a retailer's catalog online and allowing the end customer to place an order for pick-up or delivery, and optionally pre-pay for the order. Due to the simplicity, complex concepts like Promotions, Discounts, and Compliance are not part of the current solution and may need to be handled manually by the integrator.

Definitions

Company: A business, firm, or corporation that may operate one or many locations as physical brick-and-mortar stores and/or virtual ecommerce presence. Most calls to Cova APIs require a *CompanyId*, which is included in every integration onboarding package we supply.

Location: A physical store or dispensary customers can visit to purchase cannabis and accessories. A company may have many locations. Each location has its own inventory and may define its own prices. Many Cova API endpoints require a location id, commonly represented as *EntityId*.

Room: Cova supports the concept of rooms. This is a way to segregate inventory within a location. A typical setup would consist of a "Back Room" where new stock is unboxed and received into the system, and a "Sales Floor" used to fulfill sales to end customers. Note: a location can have any number of rooms, but only one room can be designated for processing sales.

Groups, Divisions, and Entity: A company can define a structure within Cova. We often refer to this as the Company Tree. The tree may have many branches of Groups and/or Divisions. Figure 1 below shows a Company (COVA Café), with a Division (Canada), and many Groups (Alberta, Ontario, etc.). Locations can exist at any level within the tree. The tree is dynamic and can have any number of Divisions and Groups nested within it, or none. An Entity is any node along the tree. "Canada" is an entity, "Ontario" is an entity, and "Hamilton" is also an entity which we commonly refer to as a Location. Only Location entities can have inventory and process sales.

TIP, TEP: Tax Inclusive pricing (TIP), Tax Exclusive Pricing (TEP). Retailers may configure their locations (separately) to include taxes in pricing (out-the-door pricing), or to calculate and add taxes to product prices at the time of sale. The default pricing scheme is *exclusive* pricing (taxes are *not* included in the price)

IntegratorId: Required for v2 and v3 order creation and TIP/TEP orders. Set up by Cova and supplied to the integrator. An integrator may have multiple *IntegratorIds*. We recommend two, one for "online" (ecommerce sales) and one for "kiosk" (self-serve checkout, for example). Figure 2 shows the Orders view within the Cova POS app. The Order Source is the name provided when an *IntegratorId* is created. In this example, the integrator has an id that maps to the name "LOTR Online"

Cova Dev Portal: Documentation for integration developers regarding Cova APIs. These can be found at https://api.covasoft.net/Documentation.

Figure 1 Company Tree example

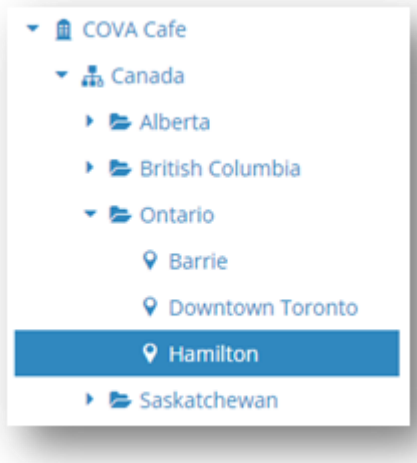
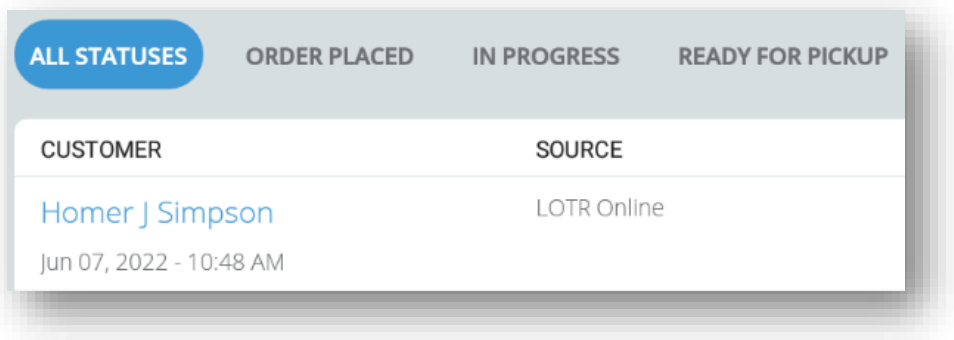


Figure 2 Order view, Cova POS app



Endpoints - TIP, and TEP

Use the TIP and TEP order endpoints and discontinue use of the original /CovaOrder endpoints (v1, v2, and v3). This new generation of endpoints are explicit and enable more accurate pricing calculations, particularly when tiered pricing is used. For more information on item pricing within Cova and example calculations, see the Item Pricing on the Cova Dev Portal.

In the following API endpoint examples, values enclosed in double curly brackets represent a variable that must be supplied. I.e. {{*CompanyId*}}.

Creation/Status API calls

Endpoint	Verb	Notes
covaorderintake/v1/TEPCovaOrder({{SalesOrderId}})	PUT	Correct endpoint for TEP locations
covaorderintake/v1/TIPCovaOrder({{SalesOrderId}})	PUT	Correct endpoint for TIP locations
covaorderintake/v2/Companies({{CompanyId}})/ CovaOrder({{SalesOrderId}})/Status	GET	Replaces v1 – provides more detail

Order workflow

The full Sales Order workflow may take many paths through loading a retailer's catalog, pricing, availability, customer data, etc., and the details of each possible path will not be included in this document. The focus here will be on the specific Order creation/status calls.

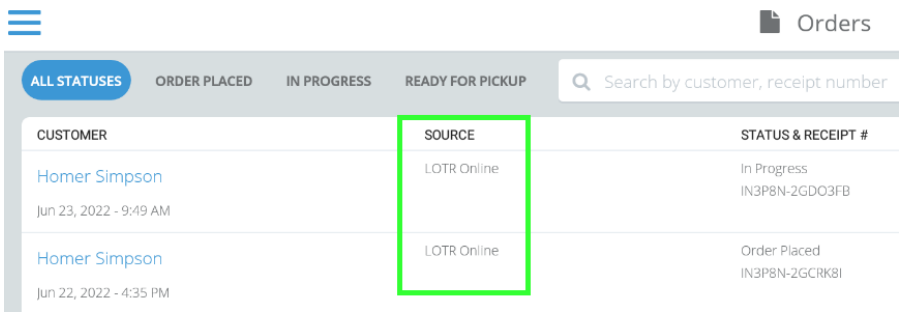
In general, assuming an Ecommerce site is responsible for creating orders, all that is required is knowledge of the location tax pricing configuration (is it TIP or TEP), a list of Cova Catalog items, and the relevant item pricing. A customer/patient is optional but highly recommended. See the Cova Dev Portal for more information on *Catalog, Customer,* and *ItemPricing*.

IntegratorId and Order Source

An *IntegratorId* will be required to create a Sales Order. This value is created by Cova and supplied for your use. The purpose of the *IntegratorId* is to identify the source of an order. The *Name* associated with the ID will display in Cova POS as shown in Figure 3 below. Two Ids are typically recommended to help track order source. In the example below, our "Lord of the Rings" integrator has:

- LOTR Online
- LOTR Kiosk

Figure 3 Cova POS order source



While onboarding, when Cova creates the association between an Integrator and a Retailer, our automation also sets up Online Payment types that match the source name. For example, the LOTR Online order source/id will also create an associated LOTR Online Payment for the retailer. If you process payments in your solution and submit the payment info to Cova to indicate a paid order, we will associate the payment to this type. If the source and payment names do not match within Cova, the default payment type will simply be Online Payment.

Tax pricing configuration

To determine which Cova Order endpoint to use, get the TaxPricingConfiguration resource:

GET taxes/v1/Companies({{CompanyId}})/TaxPricingConfiguration

Figure 4 tax pricing configurations



In the example above, location 94451 is configured with "Exclusive" pricing, which means the /TEPCovaOrder endpoint should be used to properly create tax exclusive sales orders. The only other valid result is "Inclusive", where taxes are included in the prices.

This configuration will rarely change. It is not trivial to switch between the tax schemes as all pricing for a location must be properly adjusted. Cova will communicate to each concerned integrator before a location configuration is to be changed. Refreshing a cache of this data once daily is sufficient (do not call before creating each order). Figure 4 shows the result at the company level (all configurations). If a location is *not* listed in this configuration, it is by default a TEP location.

PUT Order

The input body is the same for both TIP and TEP orders. The difference is a sale created via /TIPCovaOrder will use the *LineDollarAmount* to reverse calculate the included taxes, while an order created via /TEPCovaOrder will have taxes added to the *LineDollarAmount*. Figure 5 shows the input body used to create one of each order type at a single location for demonstration purposes. The example location is configured for tax exclusive pricing (TEP) which is the default configuration. As an integrator, you are responsible for generating the Id (GUID/UUID). If your system generates an Order Identifier that is visible by the retailers, please also submit this value in the "ExternalOrderId" property.

Figure 5 Order PUT Body



```
{
  "Id": "{{SalesOrderId}}",
  "CompanyId": 539009,
  "EntityId": 539010,
  "OrderReference": "Homer Simpson",
  "CustomerId": "ea93917e-8d2b-4d0d-b026-c9ab320583f9",
  "IntegratorId": "{{IntegratorId}}",
  "IsDelivery": "true",
  "ExternalOrderId":"379947-KLM373", //recommended
  "Notes":"Please use side door", //optional
  "TimeSlotDisplayString": "2 PM - 4 PM", //optional
  "Lines": [
    {
      "Name": "Purple Rain - 3.5g",
      "CatalogItemId": "6f3456c9-ebf2-4640-be26-032f4dfbed81",
      "Quantity": 1,
      "PackageId":null,
      "LineDollarAmount": 22.49
    }
  ]
}
```

When supplying Notes and TimeSlotDisplayString, please be brief as these are shown on Cova POS where space is limited. Also note that the time slot property is for informational purposes and does not trigger any further events or business logic around delivery or pickup orders.

The response is an empty body with an HTTP status of 202 – Accepted.

Fetch Order Status

The PUT call above does not return errors for validation failures. Only a malformed body, authorization failure, or downstream service outage would cause anything but a 202 response. Once the call returns, fetch from the v2 /Status endpoint to see the state of the order. If it was successful, the order will be “ReadyForPayment”, if it was not successful an error status/message is returned instead. See OrderStatusMessage for other states. The following table demonstrates the result of creating both order types with an identical input body to Figure 5 above.

TIP	TEP
Order Status `{ "companyId": 94447, "orderId": "b1cf6669-1f56-49a9-8b7f-1dc80f8528dc", "orderStatus": "ReadyForPayment", "orderMessage": "", "paymentStatus": "ReadyForPayment", "paymentMessage": "", "saleTotal": 25.00 }`	Order Status `{ "companyId": 94447, "orderId": "0b3b2151-5cc8-42aa-8416-29c2c0d16270", "orderStatus": "ReadyForPayment", "orderMessage": "", "paymentStatus": "ReadyForPayment", "paymentMessage": "", "saleTotal": 31.31 }`
Sale total detail:	Sale total detail:

```
"Total": {
    "Products": 19.96,
    "Taxes": 5.04
```

```
"Total": {
    "Products": 25.0,
    "Taxes": 6.31
```


Receipt Number Reservation

ReceiptNumbers can be reserved for a given OrderId ahead of time. This will allow callers to determine what the ReceiptNumber will be for an order before the order is created. Which can be useful for order correlation and traceability

PUT v1/Companies({{companyId}})/CovaOrderReceiptNumberReservation({{orderId}})


```
{
    "orderId": "6c0449e8-8af7-4cda-a9fc-6860eb4d23ad",
    "receiptNumber": "IN8A6J-13IC8PK"
}
```


This receiptNumber will be used for this orderId when it's eventually created. There is no expiration for these reserved receiptNumbers, so they can be reserved any amount of time before the order is created.

If there is no reserved receiptNumber for an orderId, a new one will be generated when the order is created

Calculating LineDollarAmount

Item pricing is covered in detail in a document like this. It can be found under ***ItemPricing*** on the Cova Dev Portal. An important distinction between legacy order endpoints and the current TIP/TEP endpoints is the introduction of *LineDollarAmount* which replaces *SellAtPricePerUnit.* This allows more accurate pricing, particularly when tiered pricing is involved. In a simple case, assume a customer wishes to purchase 3 of "product X". The regular price of that item is $4. The *LineDollarAmount* is then $12 (3 x $4).

Let us now assume that the retailer wishes to offer these products at 3 for $10 by using Tiered Pricing within Cova. Data from the pricing API will return a *RegularPrice* of $3.33 (per unit, $10.0 / 3, rounded) but will also have an *AtTierPrice* of $10.00.

Using *RegularPrice*: $3.33 X 3 = $9.99
Using *AtTierPrice*: ($10 / 3) * 3 = $10.00 ([*atTier* / *lowerBound*] * qty)

Always use the *AtTierPrice* if it exists.

Discounts & Promotions

Discounts and Promotions are not fully supported in the Order system. A discount can be achieved by supplying a *LineDollarAmount* that is lower than current Cova pricing. This will apply a generic line-level discount to the sale that

does not tie back to a discount reason nor promotion. Any promotions need to be manually calculated and applied to the applicable products on the order. We do not currently have an API available to determine discounts and promotions based on a collection of items. All this work is done by the Cova POS App.

If there is a sale price in effect when an order is submitted via API, the Order services will automatically adjust any higher supplied price to match the sale price.

Important: If an end-user edits an order on the point-of-sale device, the POS will apply discounts and promotions to the cart, which will duplicate any existing discounts on the order. A location-specific setting exists to minimize the possibility of discount duplication. See Remove Integrator Discounts on the Cova Support and Community site.

Delivery and customer addresses

To create an order for delivery, simply set the *IsDelivery* property to "true" in the Order body. A *customerId* is not required, but if one is not provided the sale cannot be completed at POS until a customer with a valid delivery address is added to the sale.

If a valid *customerId* is added to the order, the first "Shipping" address if any exist will be chosen as the shipping address, then will fall back to the first address that exists on the customer entity. Currently there is no way to provide an address identifier or choose the delivery address when creating an Order.

Any Customer Address changes after order creation are not reflected on existing sale/order.

See the Cova Dev Portal for more information on available Customer APIs

Payment

If the integrator solution includes accepting payment for orders online, the Cova Sales Order can be marked as paid by calling covaorderintake/v1/CovaOrderPayment

Figure 6 CovaOrderPayment body

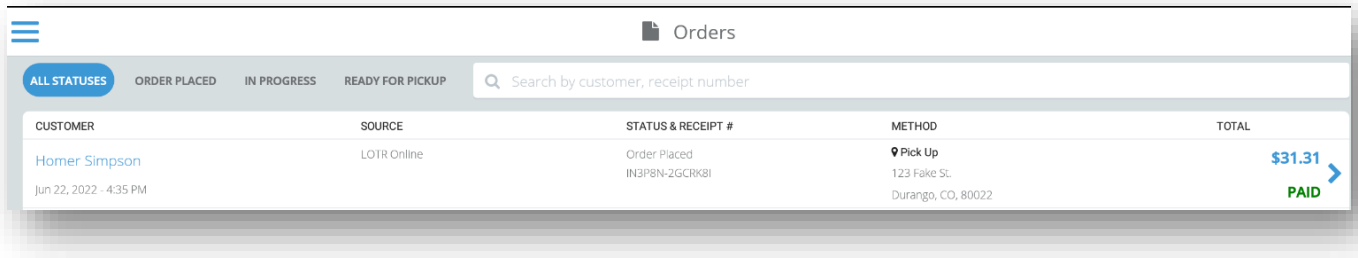


The response is an empty body with an HTTP status of 202 - Accepted

The *Amount* submitted must exactly match the *saleTotal*, as shown in the /status call above. The *TrackingNumber* is required. The purpose of this field is to hold a delivery identifier like a driver name, id, or phone number, FedEx tracking number, etc. If no tracking number is applicable, supply some text to signify this, such as "n/a" or "none". The same is true for the *PreAuthorizationToken.* You may supply a value from your payment system, or some "not applicable" text.

When an order is paid this will be reflected in the Cova POS in the Orders section. The /status endpoint will also show the status of the payment in the *paymentStatus* field. See PaymentStatusMessage.

Figure 7 Orders area in Cova POS



Paid orders cannot be modified or cancelled via POS or API.

Cancelling

An unpaid order can be cancelled within the Cova POS or via API by calling:

DELETE /pointofsale/Companies({{CompanyId}})/Sales({{SalesOrderId}}).

The retailer can also cancel an order by opening the order and selecting the trash can icon. They will be prompted to confirm removal/cancellation. Paid orders cannot be cancelled.

Completing

Completing an order is typically done via Cova POS once the budtender has packaged the items, taken payment (if not pre-paid), and given the customer their purchase. However, the Order could in fact be completed via API. A common scenario would be for delivery orders. Once the customer has received their order, the driver may initiate an action via an integrated process that would mark the order as complete. The API call for completing an order is:

POST /pointofsale/Companies({{CompanyId}})/Sales({{SalesOrderId}})/Complete

Once an order is marked as Complete, it is no longer available in the Orders section of the Cova POS. It is converted into an Invoice, which is a finalized sale.

Rooms

For an explanation of what a Room is in Cova, see the top of this document.

Sales can only be processed from the “Selling Room”. Some retailers do not use rooms, so the default receiving and selling rooms are one and the same. However, when retailers use separate rooms, they must move inventory to the Selling Room before it is accessible to the Order system. Creating an Order for an item that is in stock in the store, but not in the Selling Room will result in a message like this when checking the Order /Status

```
"orderStatus": "NonTransientProcessingFailure",
"orderMessage": "The order could not be created because availability for product 1060[redacted]d620 is not available.",
"paymentStatus": "NotReadyForPayment",
```

Tax Calculation

If you wish to display the applicable taxes prior to the customer submitting the order, you can call a Tax Calculation endpoint, depending on the Tax Pricing Configuration (see above) for the location.

For Tax Exclusive locations (default behavior):
POST /taxes/v2/TaxCalculation

```
{
    "CompanyId": {{CompanyId}},
    "EntityId": {{LocationId}},
    "LineItems": [
        {
            "LineNumber": 1,
            "ProductCatalogId": "{{CatalogItemId}}",
            "Quantity": 3,
            "Price": 3.33, //SoldAt unit price
            "BasePriceListPrice" : 3.33 //the "regular" unit price
        }
    ]
}
```

The response is detailed for each "LineItem" but there is also a TaxTotals section, so you can show every tax for every product, or just the tax totals. Choice is yours.

For Tax Inclusive locations:

POST /taxes/v2/TaxInclusiveCalculation

```
{
    "CompanyId": {{CompanyId}},
    "EntityId": {{LocationId}},
    "LineItems": [
        {
            "CatalogId": "{{CatalogItemId}}",
            "Quantity": 4,
            "LineDollarAmount": 40.60, //total line price, not unit price
            "PerUnitBasePriceListPrice" : 10.15 //"regular" unit price
        }
    ]
}
```

The response is detailed for each "LineItem". There is no TaxTotals section

Resource and property definitions

Order

Property	Type	Notes
Id	GUID	Value required to be supplied by caller
CompanyId	integer	
EntityId	Integer	Location identifier of the store accepting the order
OrderReference	Text	Value shown on POS if a valid customerId is not supplied
CustomerId	GUID	Cova Customer ID – the end consumer placing the order. Optional
IntegratorId	Guid	Identifier supplied by Cova for all orders placed by an integration partner
IsDelivery	Boolean	Is this order for Delivery
ExternalOrderId	Text	Identifier for the order from your system. Only persisted in the PointOfSale service, is not reflected downstream on the sale or in reporting.
Notes	Text	Any notes relevant to the order. Please be brief
TimeSlotDisplayString	Text	Text describing a delivery time slot. Optional. Please be brief.
Lines	Array	See OrderLine

OrderLine

Property	Type	Notes
Name	Text	Name of the item
CatalogItemId	GUID	Cova catalog item id (product id)
Quantity	Decimal	Quantity being ordered

PackageId	Text	Optional. Chosen by system based on highest availability if no value is supplied. Omit by supplying null, not empty string.
LineDollarAmount	Decimal	Total price for the line. Not a per-unit price.

OrderStatus

Property	Type	Notes
companyId	Int	
orderId	GUID	The order id supplied during creation
orderStatus	Text	Limited values, see OrderStatusMessage below
orderMessage	Text	Additional information about the status
paymentStatus	Text	Limited values, see PaymentStatus below
paymentMessage	Text	Additional information about the status
saleTotal	Decimal	Total amount to be paid by customer

OrderStatusMessage

Text	Notes
SubmittedForProcessing	The api has accepted the order and placed it in the queue for processing
TransientProcessingFailure	An internal error happened within Cova, and Cova was unable to recover from it. Please resubmit the order or payment
NonTransientProcessingFailure	The order could not be processed, usually due to invalid input, or another validation issue. Refer to the message property to get error details
SubmittedForFinalProcessing	Validation and pre processing are complete, and the order is being written to the order system.
ReadyForPayment	The order is in a confirmed status, and ready to accept payment
Cancelled	The order is cancelled and cannot be interacted with further
Completed	

PaymentStatusMessage

Text	Notes
NotReadyForPayment	Payment cannot yet be submitted
TransientProcessingFailure	An internal error happened within Cova, and Cova was unable to recover from it. Please resubmit the payment.
NonTransientProcessingFailure	The payment could not be processed, usually due to invalid input, or another validation issue. Refer to the message property to get error details.
ReadyForPayment	The order is in a confirmed status, and ready to accept payment
PaymentCannotBeApplied	The payment could not be processed, usually due to invalid input, or another validation issue. Refer to the message property to get error details.
PaymentSubmittedForProcessing	Payment is being processed

PaymentApplied	Payment has been applied to the order