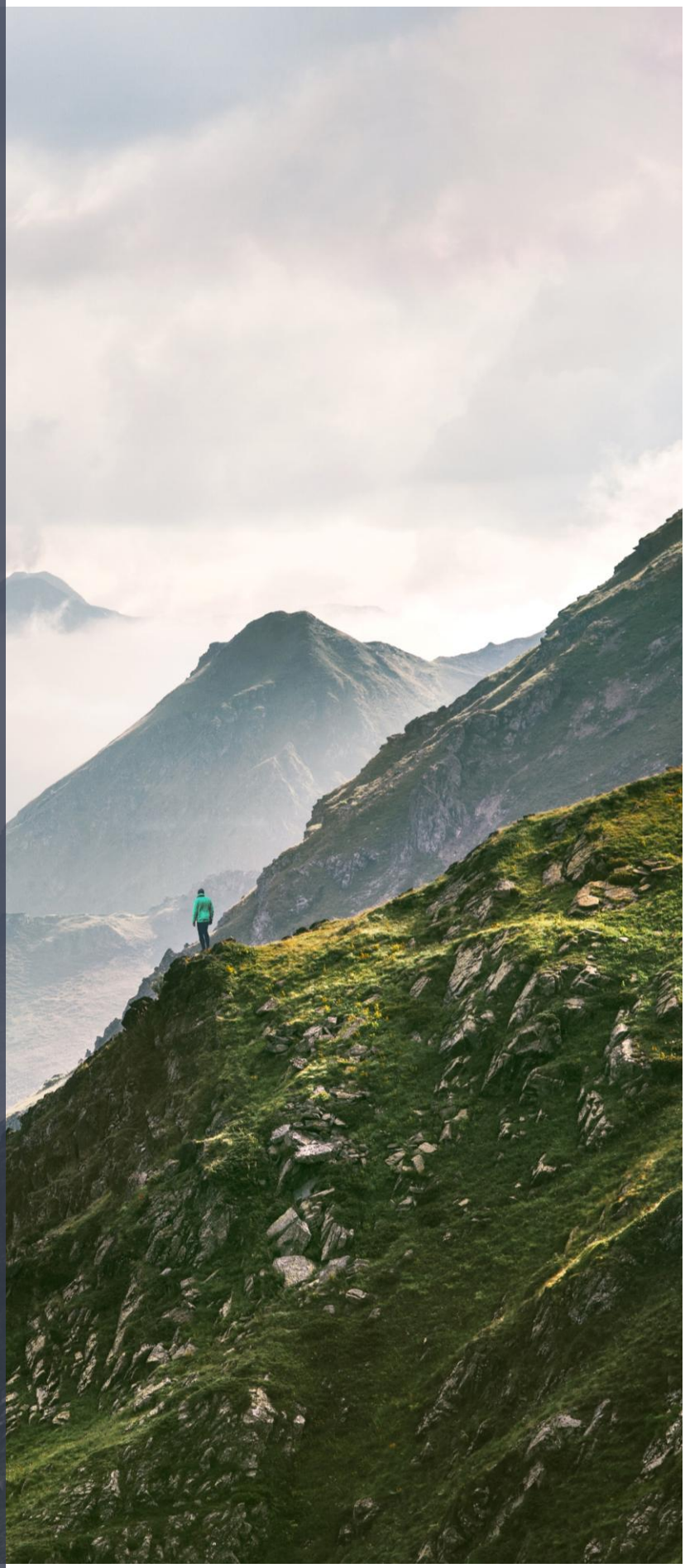
COVA™

CANNABIS RETAIL SOFTWARE

Promotions

Explanations and examples

2026



COPYRIGHT BY COVA SOFTWARE©

Cova Promotions API

This next generation of advanced promotions enables Cova customers to create and modify complicated promotions. The endpoints outlined in this document allow for querying of these promotions for external use.

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*. Promotions do not require EntityIds for queries, but there's a collection of them on each promotion that indicates the location(s) the promotions are active.

Entity: A generic term for locations, divisions, groups, and the company itself. Generally, a multi-location company will have their so-called *EntityTree* setup in a way that reflects the real world. Where each location lives under a group/division, and all lives under the *Company* entity. The *Company* acts as the root, with divisions and/or groups as branches connected to the root, and *Locations* connected to either acting as leaf nodes. All of these come together to create the EntityTree, which is a hierarchically organized tree of EntityIds that is used by promotions to configure which locations the promotion in question is enabled at.

Cart: A term for a list of products, their quantities and prices. Represents a sale before being tendered. Each line item in the cart has the same price per-unit, catalogId, and package number. Adding a duplicate product increases the quantity of the line item and does not add another line item.

CatalogId: Also known as ProductId. It is a GUID/UUID value.

Endpoint details

The promotions for a given company are available through the following endpoints.

For the first, the caller is required to provide two bitmask fields that represent the conditions and types the caller has implemented. The API will then filter the promotions based on the given bitmasks and only return promotions that are handleable by the caller. Details about these bitmasks can be found later in this document (see Promotion Capability Bitmask, ProductCondition Bitmask, CartCondition Bitmask, LineCondition Bitmask) There should be no need to make multiple calls to this endpoint per day.

GET promotions/v1/Companies/{companyId}/ConditionCapabilities/{conditionCapabilities}/PromotionTypeCapabilities/{promotionTypeCapabilities}/Promotions

Returns a list of promotions that are filtered by the given capability bitmasks

Response Body Properties:

Promotion

Name	Type	Description
PromotionId	guid	A unique identifier for this promotion
CompanyId	int32	The CompanyId this promotion belongs to. Will always match the CompanyId provided in the URL
Name	string	The name of the promotion
Status	string	The status of the promotion. Either 'Active' or 'Deleted'. Will always be 'Active' on this endpoint. Does not reflect the scheduled begin/end of the promotion
HumanReadablePromotionType	string	A human readable label for the type of promotion. One of: 'BOGO', 'Buy X Get Y', 'Bundle', 'Dollar Off' or 'Percent Off'
EnabledAtLocationIds	int32[]	A list of the Location identifiers where this promotion is applicable
ICalVEventSchedule	string	A ICal formatted string representing the schedule for this promotion. See https://icalendar.org/ for details
PromotionType	PromotionType	An object containing most of the details about how to apply the promotion and when it's applicable to do so
CartCondition	CartCondition	A nestable tree-based object representing the requirements a cart needs to meet to apply this promotion
LineCondition	LineCondition	A nestable tree-based object representing the requirements a line needs to meet to be considered for this promotion
CreatedByUserId	int32	The UserId that created this promotion
CreatedDateTimeUTC	DateTime	The UTC date/time this promotion was created

Version	int32	The version of this promotion. Incremented by 1 for each change made to this promotion

The second endpoint does not require the use of capability bitmasks and is therefore simpler to consume. It returns all promotions for a company with a list of catalogIds that *may be* qualified for discount if all conditions are met. This endpoint is intended for integrators who wish to display items as "on promotion". No information about how to apply the promotion nor the applicable discount are available through this endpoint. If an integrator wants to apply a promotion, they must use the first endpoint and not this one. The catalogId data is calculated and cached for approximately on hour (subject to change without notice).

GET promotions/v1/Companies/{companyId}/PromotionsForMenuBoard

Returns a list of promotions with a list of CatalogIds that the promotion can apply to

Response Body Properties:

PromotionForMenuBoard

Name	Type	Description
PromotionId	guid	A unique identifier for this promotion
CompanyId	int32	The CompanyId this promotion belongs to. Will always match the CompanyId provided in the URL
Name	string	The name of the promotion
Status	string	The status of the promotion. Either 'Active' or 'Deleted'. Will always be 'Active' on this endpoint. Does not reflect the scheduled begin/end of the promotion
HumanReadablePromotionType	string	A human readable label for the type of promotion. One of: 'BOGO', 'Buy X Get Y', 'Bundle', 'Dollar Off' or 'Percent Off'
EnabledAtLocationIds	int32[]	A list of the Location identifiers where this promotion is applicable
ICalVEventSchedule	string	A ICal formatted string representing the schedule for this promotion. See https://icalendar.org/ for details
CreatedByUserId	int32	The UserId that created this promotion
CreatedDateTimeUTC	DateTime	The UTC date/time this promotion was created
Version	int32	The version of this promotion. Incremented by 1 for each change made to this promotion
CatalogIds	Guid[]	A list of CatalogIds that this promotion can apply to

Promotion Types

The promotion type contains details about the discount provided by the promotion as well as conditions that define when it can be applied.

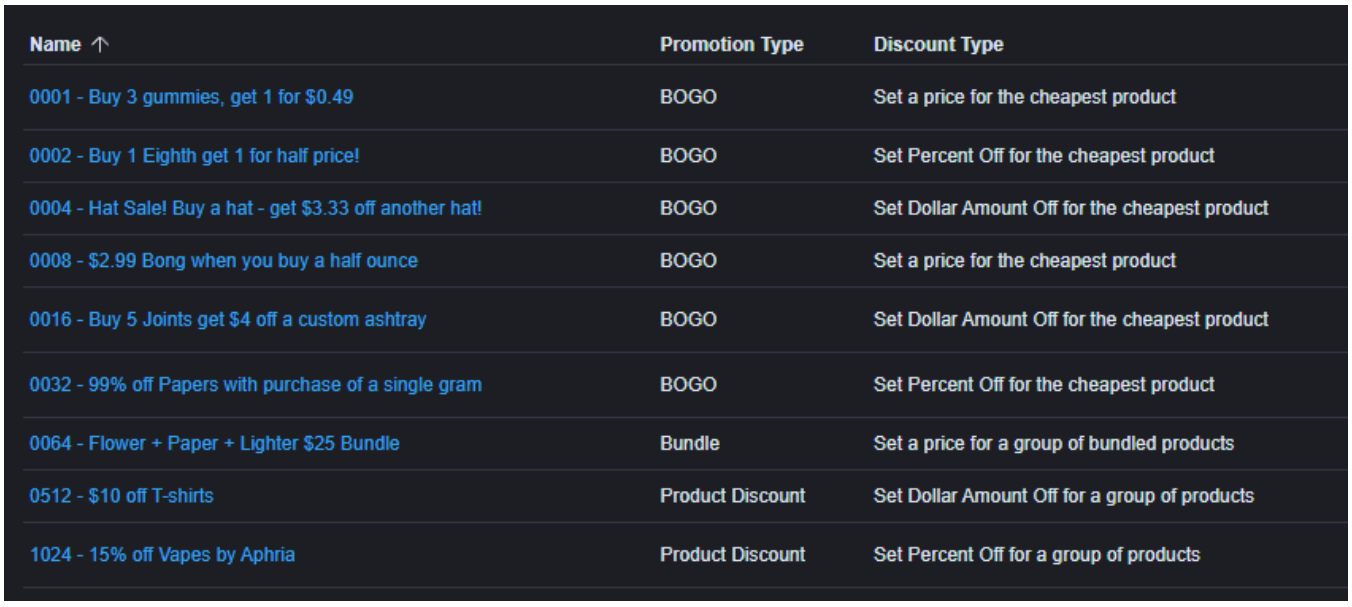
Promotion Capability Bitmask

Here is a table of each PromotionType and their corresponding bit.

Type	Bit
CheapestMatchedForDollar	1
CheapestMatchedForPercentOff	2
CheapestMatchedForDollarOff	4
MatchThenCheapestOtherForDollar	8
MatchThenCheapestOtherForDollarOff	16
MatchThenCheapestOtherForPercentOff	32
BundleForTotalDollarDistributed	64
BundleForTotalDollarOffDistributed	128
BundleForPercentOff	256
EachMatchedDollarOff	512
EachMatchedPercentOff	1024

Different fields will be non-null depending on the value in 'Type'

The screenshot below shows examples of the promotion types supported as of writing this document. The first four characters of the name are the bitmask values from above, added here for illustration purposes only. They were zero-padded to assist with sorting.



Name	Promotion Type	Discount Type
0001 - Buy 3 gummies, get 1 for $0.49	BOGO	Set a price for the cheapest product
0002 - Buy 1 Eighth get 1 for half price!	BOGO	Set Percent Off for the cheapest product
0004 - Hat Sale! Buy a hat - get $3.33 off another hat!	BOGO	Set Dollar Amount Off for the cheapest product
0008 - $2.99 Bong when you buy a half ounce	BOGO	Set a price for the cheapest product
0016 - Buy 5 Joints get $4 off a custom ashtray	BOGO	Set Dollar Amount Off for the cheapest product
0032 - 99% off Papers with purchase of a single gram	BOGO	Set Percent Off for the cheapest product
0064 - Flower + Paper + Lighter $25 Bundle	Bundle	Set a price for a group of bundled products
0512 - $10 off T-shirts	Product Discount	Set Dollar Amount Off for a group of products
1024 - 15% off Vapes by Aphria	Product Discount	Set Percent Off for a group of products

Here is a table representing all fields that could possibly be supplied. See later tables to determine which fields are supplied in what case, as well as their descriptions

Name	Type	Description
Type	string	The type of PromotionType this object represents.
DollarValueOfCheapest	decimal?	
PercentOffOfCheapest	decimal?	

DollarOffOfCheapest	decimal?	
DollarValueOfOther	decimal?	
DollarOffOfOther	decimal?	
PercentOffOfOther	decimal?	
DollarValueOfAll	decimal?	
DollarOffOfAll	decimal?	
PercentOffOfAll	decimal?	
DollarOffOfEach	decimal?	
PercentOffOfEach	decimal?	
ItemsToMatch	ProductCondition?	
OtherItemConditions	ProductCondition?	
MatchConditions	ProductCondition?	
BundleItemsToMatch	BundleMatch []?	
NumberToMatch	int32?	
GramsPerMatchUnit	decimal?	
MaxApplicationCount	int32?	

BundleMatch

Name	Type	Description
ProductCondition	ProductCondition	The ProductCondition tree used to determine which product(s) can be considered as an element in this bundle
QuantityToMatch	decimal	The Quantity of products matching the ProductCondition required to be considered an element in this bundle. 'GramsPerMatchUnit' is used to convert gram-based quantities into each-based quantities.

CheapestMatchedForDollar

Name	Type	Description
Type	string	'CheapestMatchedForDollar'
DollarValueOfCheapest	decimal	The dollar value to be charged for the cheapest product applicable to this promotion.
ItemsToMatch	ProductCondition	The ProductCondition tree used to determine which product(s) apply to this promotion
NumberToMatch	int32	The number of items that must pass the 'ItemsToMatch' filter for this promotion to apply
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'NumberToMatch'
MaxApplicationCount	int32?	An optional field representing the maximum number of times this promotion can be applied to a single cart. Null is equivalent to no restriction

CheapestMatchedForPercentOff

Name	Type	Description
Type	string	'CheapestMatchedForPercentOff'
PercentOffOfCheapest	decimal	The percentage discount that should be applied to the cheapest product applicable to this promotion. Between 0.0 and 1.0
ItemsToMatch	ProductCondition	The ProductCondition tree used to determine which product(s) apply to this promotion
NumberToMatch	int32	The number of items that must pass the 'ItemsToMatch' filter for this promotion to apply
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'NumberToMatch'
MaxApplicationCount	int32?	An optional field representing the maximum number of times this promotion can be applied to a single cart. Null is equivalent to no restriction

CheapestMatchedForDollarOff

Name	Type	Description
Type	string	'CheapestMatchedForDollarOff'
DollarOffOfCheapest	decimal	The dollar amount the cheapest product applicable to this promotion should be discounted by
ItemsToMatch	ProductCondition	The ProductCondition tree used to determine which product(s) apply to this promotion
NumberToMatch	int32	The number of items that must pass the 'ItemsToMatch' filter for this promotion to apply
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'NumberToMatch'
MaxApplicationCount	int32?	An optional field representing the maximum number of times this promotion can be applied to a single cart. Null is equivalent to no restriction

MatchThenCheapestOtherForDollar

Name	Type	Description
Type	string	'MatchThenCheapestOtherForDollar'
DollarValueOfOther	decimal	The dollar value of the cheapest item passing the 'OtherItemConditions' filter
OtherItemConditions	ProductCondition	The ProductCondition tree used to determine which product(s) are available for this discount
MatchConditions	ProductCondition	The ProductCondition tree used to determine which product(s) apply to this promotion

NumberToMatch	int32	The number of items that must pass the 'ItemsToMatch' filter for this promotion to apply
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'NumberToMatch'
MaxApplicationCount	int32?	An optional field representing the maximum number of times this promotion can be applied to a single cart. Null is equivalent to no restriction

MatchThenCheapestOtherForDollarOff

Name	Type	Description
Type	string	'MatchThenCheapestOtherForDollarOff'
DollarOffOfOther	decimal	The dollar value of the discount applied to the cheapest item passing the 'OtherItemConditions' filter
OtherItemConditions	ProductCondition	The ProductCondition tree used to determine which product(s) are available for this discount
MatchConditions	ProductCondition	The ProductCondition tree used to determine which product(s) apply to this promotion
NumberToMatch	int32	The number of items that must pass the 'ItemsToMatch' filter for this promotion to apply
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'NumberToMatch'
MaxApplicationCount	int32?	An optional field representing the maximum number of times this promotion can be applied to a single cart. Null is equivalent to no restriction

MatchThenCheapestOtherForPercentOff

Name	Type	Description
Type	string	'MatchThenCheapestOtherForPercentOff'
PercentOffOfOther	decimal	The percentage discount that should be applied to the cheapest product item passing the 'OtherItemConditions' filter. Between 0.0 and 1.0
OtherItemConditions	ProductCondition	The ProductCondition tree used to determine which product(s) are available for this discount
MatchConditions	ProductCondition	The ProductCondition tree used to determine which product(s) apply to this promotion
NumberToMatch	int32	The number of items that must pass the 'ItemsToMatch' filter for this promotion to apply
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'NumberToMatch'
MaxApplicationCount	int32?	An optional field representing the maximum number of times this promotion can be applied to a single cart. Null is equivalent to no restriction

BundleForTotalDollarDistributed

Name	Type	Description
Type	string	'BundleForTotalDollarDistributed'
DollarValueOfAll	decimal	The total dollar amount for all items defined in 'BundleItemsToMatch'
BundleItemsToMatch	BundleMatch[]	A list of BundleMatch objects which define each element of this bundle. An element can be more than a single quantity
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'QuantityToMatch'

BundleForTotalDollarOffDistributed

Name	Type	Description
Type	string	'BundleForTotalDollarOffDistributed'
DollarOffOfAll	decimal	The total dollar amount all items defined in 'BundleItemsToMatch' should be reduced by
BundleItemsToMatch	BundleMatch[]	A list of BundleMatch objects which define each element of this bundle. An element can be more than a single quantity
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'QuantityToMatch'

BundleForPercentOff

Name	Type	Description
Type	string	'BundleForPercentOff'
PercentOffOfAll	decimal	The percentage discount that should be applied to all items defined in 'BundleItemsToMatch'. Between 0.0 and 1.0
BundleItemsToMatch	BundleMatch[]	A list of BundleMatch objects which define each element of this bundle. An element can be more than a single quantity
GramsPerMatchUnit	decimal	The number of grams that are considered a single unit for use with 'QuantityToMatch'

EachMatchedDollarOff

Name	Type	Description
Type	string	'EachMatchedDollarOff
DollarOffOfEach	decimal	The dollar amount discount each item passing the 'ItemsToMatch' filter should be given.
ItemsToMatch	ProductCondition	The ProductCondition tree used to determine which product(s) are available for this discount
GramsPerMatchUnit	decimal	The number of grams that should be considered a single unit when applying the 'DollarOffOfEach' discount

EachMatchedPercentOff

Name	Type	Description
Type	string	'EachMatchedPercentOff
PercentOffOfEach	decimal	The percentage discount that should be applied to all items passing the 'ItemsToMatch' filter. Between 0.0 and 1.0
ItemsToMatch	ProductCondition	The ProductCondition tree used to determine which product(s) are available for this discount

ProductCondition

A ProductCondition is an object that represents a node in a tree of other ProductConditions that is used to filter products applicable to a promotion. An entire ProductCondition tree must evaluate to True for a line item to be applicable to a promotion.

Like the PromotionType, there is a Type string field on each object that determines which other fields are populated

Each node type of ProductCondition has a corresponding bit set in the ConditionCapabilities bitmask that is required to be supplied to the endpoint when retrieving promotions. The caller is required to supply a bitmask representing the types of nodes that it supports. If a promotion has a node in its ProductCondition tree that isn't '1' in the given bitmask it will not be returned in the response. Some nodes require information about a product that may or may not be available to the caller, so the bitmask is a way for the caller to filter out promotions that it cannot support.

Note: ConditionCapabilities bitmask contains the bitmask for CartCondition, LineCondition and ProductCondition.

ProductCondition Bitmask

Here is a table of each ProductCondition node type and its corresponding bit.
Note how some nodes don't have a bit. Those are required to be implemented by all callers

Node	Bit
AllOf	N/A
AnyOf	N/A
NoneOf	N/A
None	N/A
Classification	16
CatalogId	N/A
NonStock	32
BatchTracked	64
GiftCard	128
Regular	256
ContainsCannabis	512
IsGram	1024
IsEach	2048
SupplierId	4096
SpecificationValue	16384

To determine if products match the conditions listed above, you will need product data from Cova. The assumption here is that you are already caching this data from Cova's DataPlatform APIs. This table shows the correlation between the promotion ProductConditions and where to find the data in a DetailedProductData response. See DataPlatform on the Cova API portal for more information.

ProductCondition Node	Bitmask	Property in DetailedProductData	Notes
CatalogId	n/a	ProductId	These are always returned in promotion API calls
Classification	16	ClassificationId	Promo has ParentCategoryOrClassificationId which *may* be a category, not classification. Ensure IncludeClassifications: true on DetailedProductData requests and check the Classifications collection to see category structure
NonStock	32	IsNonStock	= TRUE
BatchTracked	64	IsBatchTracked	= TRUE. Subject to state traceability. *Usually* means "contains cannabis"
GiftCard	128	ProductSpecifications[]	a field exists with DisplayName equal to "Integrated Gift Card" and value is true/yes. Ensure IncludeProductSpecifications:true on DetailedProductData requests
Regular	256	IsNonStock	= FALSE
ContainsCannabis	512	ProductSpecifications []	a field exists with DisplayName starting with "Retail Marijuana Product Type" and value is not null/empty. Ensure IncludeProductSpecifications:true on DetailedProductData requests
IsGram	1024	MeasurementType	= Mass
IsEach	2048	MeasurementType	= SingleUnit
Supplier	4096	SupplierSkus[]	SupplierId. Ensure IncludeProductSkusAndUpcs:true on DetailedProductData requests
SpecificationValue	16384	ProductSpecifications[]	match FieldId and exact match on Value. Ensure IncludeProductSpecifications:true on DetailedProductData requests

Here's a table with all the possible fields. See later tables for which specific fields are populated for each Type, as well as their descriptions

Name	Type	Description
Type	string	The type of ProductCondition this object represents
Conditions	ProductCondition[]?	
ParentCategoryOrClassificationId	int32?	
Id	Guid?	
SupplierId	int32?	

StringId	string?	
Value	string?	
FieldId	int32?	

AllOf

Name	Type	Description
Type	string	'AllOf'
Conditions	ProductCondition[]	A list of other ProductConditions

The inner list of ProductCondition nodes must *all* evaluate to True for this node to also evaluate to True.

AnyOf

Name	Type	Description
Type	string	'AnyOf'
Conditions	ProductCondition[]	A list of other ProductConditions

If *any* of the inner ProductConditions evaluates to True, then this node also evaluates to True

NoneOf

Name	Type	Description
Type	string	'NoneOf'
Conditions	ProductCondition[]	A list of other ProductConditions

The inner list of ProductConditions must *all* evaluate to False for this node to evaluate to True

None

Name	Type	Description
Type	string	'None'

A node representing a no-op. Always evaluates to True. Can be a node on a tree, or a whole tree in and of itself.

Classification

Name	Type	Description
Type	string	'Classification'
ParentCategoryOrClassificationId	int32	A ParentCategory or ClassificationId

This node evaluates to True when the line item belongs to the given ParentCategory or ClassificationId.

CatalogId

Name	Type	Description
Type	string	'CatalogId'
Id	guid	The CatalogId

This node evaluates to True when the line item has the provided CatalogId

NonStock

Name	Type	Description
Type	string	'NonStock'

This node evaluates to True when the line item is a non-stock product

BatchTracked

Name	Type	Description
Type	string	'BatchTracked'

This node evaluates to True when the line item is batch-tracked

GiftCard

Name	Type	Description
Type	string	'GiftCard'

This node evaluates to True when the line item is a gift card

Regular

Name	Type	Description
Type	string	'Regular'

This node evaluates to True when the line item is a regular product

ContainsCannabis

Name	Type	Description
Type	string	'ContainsCannabis'

This node evaluates to True when the line item contains cannabis

IsGram

Name	Type	Description
Type	string	'IsGram'

This node evaluates to True when the line item is gram based

IsEach

Name	Type	Description
Type	string	'IsEach'

This node evaluates to True when the line item is each based

Supplier

Name	Type	Description
Type	string	'Supplier'
SupplierId	int32	A SupplierId

This node evaluates to True when the line item belongs to the given SupplierId

SpecificationValue

Name	Type	Description
Type	string	'SpecificationValue'
StringId	string	The StringId of the specification value this node matches
Value	string	The value of the specification value this node matches

FieldId	int32	The FieldId of the specification value this node matches

This node evaluates to True when the StringId/FieldId as well as the Value matches (case-insensitive) a specification value of the product in question.

CartCondition

A CartCondition is an object that represents a node in a tree of other CartConditions that is used to filter carts appliable to a promotion. A CartCondition tree must evaluate to True to be considered for this promotion.

Like the PromotionType, there is a Type string field on each object that determines what other fields are populated.

Each node type of CartCondition has a corresponding bit set in the ConditionCapabilities bitmask that is required to be supplied to the endpoint when retrieving promotions. The caller is required to supply a bitmask representing the types of nodes that it supports. If a promotion has a node in its CartCondition tree that isn't '1' in the given bitmask it will not be returned in the response. Some nodes require information about customers and other things that the caller may or may not be aware of. This bitmask is a way for the caller to filter out promotions that it cannot support. Note: ConditionCapabilities bitmask contains the bitmask for CartCondition, LineCondition and ProductCondition.

CartCondition Bitmask

AllOf	N/A
AnyOf	N/A
NoneOf	N/A
None	N/A
MedCustomer	1
RecCustomer	2
CustomerInPricingGroup	4
CustomerNotInPricingGroup	8192
NoInvoiceLevelDiscountOnCart	262144
NoLoyaltyDiscountOnCart	524288

AllOf

Name	Type	Description
Type	string	'AllOf'
Conditions	CartCondition []	A list of other CartConditions

The inner list of CartCondition nodes must *all* evaluate to True for this node to also evaluate to True.

AnyOf

Name	Type	Description
Type	string	'AnyOf'
Conditions	CartCondition []	A list of other CartConditions

If *any* of the inner CartConditions evaluates to True, then this node also evaluates to True

NoneOf

Name	Type	Description

Type	string	'NoneOf'
Conditions	CartCondition []	A list of other CartConditions

The inner list of CartConditions must *all* evaluate to False for this node to evaluate to True

None

Name	Type	Description
Type	string	'None'

A node representing a no-op. Always evaluates to True. Can be a node on a tree, or a whole tree in and of itself.

MedCustomer

Name	Type	Description
Type	string	'MedCustomer'

This node evaluates to True when the customer is a medical customer.

RecCustomer

Name	Type	Description
Type	string	'RecCustomer'

This node evaluates to True when the customer is not a medical customer.

CustomerInPricingGroup

Name	Type	Description
Type	string	'CustomerInPricingGroup'
PricingGroupId	int32	The pricing group Id

This node evaluates to True when the customer belongs to the given pricing group.

CustomerNotInPricingGroup

Name	Type	Description
Type	string	'CustomerNotInPricingGroup

This node evaluates to True when the customer does not belong to any pricing group. Or if there is no customer associated to the sale.

NoInvoiceLevelDiscountOnCart

Name	Type	Description
Type	string	'NoInvoiceLevelDiscountOnCart'

This node evaluates to True when the cart doesn't have an invoice-level discount applied.

NoLoyaltyDiscountOnCart

Name	Type	Description
Type	string	'NoLoyaltyDiscountOnCart'

This node evaluates to True when the cart doesn't have a loyalty discount applied.

LineCondition

A LineCondition is an object that represents a node in a tree of other LineConditions that is used to filter line items appliable to a promotion. A LineCondition tree must evaluate to True to be considered for this promotion.

Like the PromotionType, there is a Type string field on each object that determines what other fields are populated.

Each node type of LineCondition has a corresponding bit set in the ConditionCapabilities bitmask that is required to be supplied to the endpoint when retrieving promotions. The caller is required to supply a bitmask representing the types of nodes that it supports. If a promotion has a node in its LineCondition tree that isn't '1' in the given bitmask it will not be returned in the response. Some nodes require information about pricing and other things that the caller may not be aware of. This bitmask is a way for the caller to filter out promotions that it cannot support.
Note: ConditionCapabilities bitmask contains the bitmask for CartCondition, LineCondition and ProductCondition.

LineCondition Bitmask

AllOf	N/A
AnyOf	N/A
NoneOf	N/A
None	N/A
NoSalePricing	8
NoTierPricing	32768
NoGroupPricing	65536
NoManualLineDiscount	131072

AllOf

Name	Type	Description
Type	string	'AllOf'
Conditions	LineCondition []	A list of other LineConditions

The inner list of LineCondition nodes must *all* evaluate to True for this node to also evaluate to True.

AnyOf

Name	Type	Description
Type	string	'AnyOf'
Conditions	LineCondition []	A list of other LineConditions

If *any* of the inner LineCondition evaluates to True, then this node also evaluates to True

NoneOf

Name	Type	Description
Type	string	'NoneOf'
Conditions	LineCondition []	A list of other LineConditions

The inner list of LineCondition must *all* evaluate to False for this node to evaluate to True

None

Name	Type	Description
Type	string	'None'

A node representing a no-op. Always evaluates to True. Can be a node on a tree, or a whole tree in and of itself.

NoSalePricing

Name	Type	Description
Type	string	'NoSalePricing'

This node evaluates to True when the line item isn't using sale pricing.

NoTierPricing

Name	Type	Description
Type	string	'NoTierPricing'

This node evaluates to True when the line item isn't using tier pricing.

NoGroupPricing

Name	Type	Description
Type	string	'NoGroupPricing'

This node evaluates to True when the line item isn't using group pricing.

NoManualLineDiscount

Name	Type	Description
Type	string	'NoManualLineDiscount'

This node evaluates to True when the line item doesn't have a manually-applied line discount.

Promotion Application

Promotions must be filtered out based on their conditions and time schedule.

The time schedule that the promotion is active during is specified in the ICalVEventSchedule field. Which must be active during the defined time at the location the sale is happening.

The CartConditions, LineConditions, and ProductConditions filters must all pass before the promotion can be applied.

See below sections for details

iCalendar

iCal is an open standard for transferring calendar information. There are a variety of libraries for many languages to parse these strings and to make use of the calendar information.

See https://icalendar.org/

The iCal standard supports a wide range of scheduling options, the vast majority of which aren't used by Cova Promotions.

Below is an outline of what to expect in a typical ICalVEventSchedule field.

```
    "ICalVEventSchedule": "BEGIN:VEVENT\r\nUID:a2875ac9-4b56-42c0-b2ac-
28406b092f0a\r\nSEQUENCE:0\r\nDTSTAMP:20240916T185552Z\r\nDTSTART:20240916T180000\r\nDTEND:20
240916T200000\r\nRRULE:FREQ=DAILY;UNTIL=20300916T200000\r\nSUMMARY:18a509c5-82fa-4b97-83ee-
c20037d26e5b\r\nEND:VEVENT\r\n",
```

Parsing the string, and applying the '\r' and '\n' yields:

BEGIN:VEVENT
UID: a2875ac9-4b56-42c0-b2ac-28406b092f0a
SEQUENCE:0
DTSTAMP:20240916T185552Z
DTSTART:20240916T180000
DTEND:20240916T200000
RRULE:FREQ=DAILY;UNTIL=20300916T200000
SUMMARY:18a509c5-82fa-4b97-83ee-c20037d26e5b
END:VEVENT

The 'SUMMARY' field contains the GUID of the promotion in question. This can be helpful when importing many VEVENT schedules into a calendar to determine which promotion is currently active

UID is a generated GUID that uniquely identifies this schedule, it isn't related to anything.

DTSTAMP is the DateTime when this schedule was created. It has no relation to the active time of a promotion
DTSTART is the DateTime that the promotion starts
DTEND is the DateTime that the promotion ends
RRULE is the recurrence rules for this promotion. In this example it recurs daily until 2030-09-16

DTSTART contains both the start date and time. In this example it starts at 18:00:00 on 2024-09-16
DTEND contains both the end date and time. In this example it ends at 20:00:00 on 2024-09-16
With the inclusion of RRULE, the end date is overwritten by the DateTime given in the UNTIL field

This promotion starts at 18:00:00 on 2024-09-16 until 20:00:00 and repeats during that time frame daily until 2030-09-16

Another example, with different recurrence rules:

BEGIN:VEVENT
UID:d5c79244-d455-4b72-8127-385633df205c
SEQUENCE:0
DTSTAMP:20240909T175013Z
DTSTART:20240806T000000
DTEND:20240806T235959
RRULE:FREQ=WEEKLY;UNTIL=20310802T235959;BYDAY=TU,TH
SUMMARY:1b3c14eb-1f89-4c2c-b2f2-d2d9af8743f7
END:VEVENT

DTSTART begins on 2024-08-06 at 00:00:00
DTEND ends on 2024-08-06 at 23:59:59
RRULE repeats weekly on Tuesday and Thursday until 2031-08-02 at 23:59:59

This promotion starts at 00:00:00 on 2024-08-06 and ends at 23:59:59, and repeats every Tuesday and Thursday until 2031-08-02. This promotion will run all-day every Tuesday and Thursday

All Cova Promotions will use VEVENT schedules that look like the above. Their specific dates and times will be different, but the general shape will be the same.
The iCal standard allows for the creation of much more complex schedules, but anything more complex than what's outlined above is not currently supported.
For example, RRULE can be used to specify yearly or monthly recurrences. There are other fields in the iCal standard that can include information about geographic locations, there are also specifications for different formats of DateTime strings. None of these are used by Cova Promotions.

Example Scenarios

Assume a retailer wants to give 20% off for a selection of products they choose by hand while building the promo.

That's PromotionTypeCapability bitmask 1024 (EachMatchedPercentOff)
"Product selection" is the default ConditionCapability and are always returned no matter which bitmask you supply. For this example, let's assume your solution can only currently process is a list of products. Your final URL with bitmasks will be:

~/ConditionCapabilities/1/PromotionTypeCapabilities/1024/Promotions

This will return only promotions of type EachMatchedPercentOff that contain a list of product identifiers

---
Now, let's assume a retailer wants to give 20% off for all items they've classified as "Pre-roll" and come from Suppler "Hi-Roller"

That's still PromotionTypeCapability bitmask 1024 (EachMatchedPercentOff)
ConditionCapability now needs to include bitmasks:

Capability	Bitmask
Classification	16
SupplierId	4096

Sum the integers: 4112

Final URL:
~/ConditionCapabilities/4112/PromotionTypeCapabilities/1024/Promotions

This will return only promotions of type EachMatchedPercentOff that contain conditions: list of product identifiers, and/or classification, and/or supplier

As you can see from this example the promotions rely on many data points from the Cova product catalog. If you are not currently capturing values like SupplierId, then you could not calculate promotions that have SupplierId conditions.

This is an example of how some of the data would look:

```
"PromotionType": {
        "Type": "EachMatchedPercentOff",
        "PercentOffOfEach": 0.2,
        "ItemsToMatch": {
                "Type": "AllOf",
                "Conditions": [
                        {
```

```
                        "Type": "AnyOf",
                        "Conditions": [
                                {
                                        "Type": "Classification",
                                        "ParentCategoryOrClassificationId": 12345 //Preroll
                                }
                        ]
                },
                {
                        "Type": "AnyOf",
                        "Conditions": [
                                {
                                        "Type": "Supplier",
                                        "SupplierId": 54321 //Hi-Roller
                                }
                        ]
                }
        ]
    }
}
```

How to apply

There are a variety of ways to setup promotions to apply different discounts. Generally, there's a 'bucket' of products that are 'consumed' when applying a promotion, and a 'bucket' that gets the promotion discount.
The 'bucket' may or may not be the same. For example, a 'CheapestMatchedFor_' promotion has a single 'bucket' of products where the cheapest product in the 'bucket' will get the discount. Whereas a 'MatchThenCheapestOtherFor_' promotion has a 'bucket' of products that are required to fulfill the promotion requirements, but there's a different 'bucket' of products that gets the discount.

A 'bucket' in this case is conceptualization of a product(s) at a quantity that are grouped together to be included as part of the promotion.
These could be many quantities of a single product type, or many product types at a single quantity each. It will depend on the promotion which product(s) are included in the promotion.

It's important to know that once a product is used to apply a promotion, it cannot be used to apply another promotion.
This is where the concept of 'consumption' comes into play. Once a product is determined to be used, or discounted by a promotion, it is 'consumed' and cannot be used for other promotions that may apply.

There are many discounts that promotions can apply. Knowledge about how to apply each PromotionType is implicit in the name of the PromotionType.
For example, a CheapestMatchedForDollar PromotionType can be considered in two sections 'CheapestMatchedFor' and 'Dollar'.
Where the first part is the type of promotion. The second part is the type of discount that applies.

All PromotionTypes follow this naming convention:

PromotionType	Type Section	Discount Section
CheapestMatchedForDollar	CheapestMatchedFor	Dollar
CheapestMatchedForDollarOff	CheapestMatchedFor	DollarOff
CheapestMatchedForPercentOff	CheapestMatchedFor	PercentOff
MatchThenCheapestOtherForDollar	MatchThenCheapestOtherFor	Dollar
MatchThenCheapestOtherForDollarOff	MatchThenCheapestOtherFor	DollarOff
MatchThenCheapestOtherForPercentOff	MatchThenCheapestOtherFor	PercentOff
BundleForTotalDollarDistributed	BundleFor	TotalDollarDistributed
BundleForTotalDollarOffDistributed	BundleFor	TotalDollarOffDistributed
BundleForPercentOff	BundleFor	PercentOff
EachMatchedDollarOff	EachMatched	DollarOff
EachMatchedPercentOff	EachMatched	PercentOff

An explanation of each PromotionType and their corresponding discounts is below.
Starting with the simplest, and ending with the most complex:

EachMatched

The EachMatched promotion types are the simplest.
There is a single bucket in which all products associated with this promotion fall under, and the discount is applied to all products in that bucket.

This promotion type is equivalent to, or at least mappable from, the older promotion types within Cova.

For all EachMatched promotions, the conditions specified in the 'ItemsToMatch' field indicate which product(s) are included in this promotion. There is no restriction on the quantity of products on these promotions, unlike other promotions.

Once a product is determined to be on promotion with this promotion type, there are two possible discounts it can receive:
DollarOff and PercentOff

DollarOff indicates the dollar amount that each product is discounted by. GramsPerMatchUnit needs to be used to convert a gram-based product into an each-based for the dollar amount to be applied properly.

PercentOff is simply the percentage discount that each product should receive. A value of 0.3 indicates each product should receive a 30% discount. GramsPerMatchUnit is not needed here because the percentage applies regardless of the quantity in question.

CheapestMatchedFor

The CheapestMatchedFor promotion type is one that again specifies rules for a bucket of products. Once the bucket reaches the quantity specified in the 'NumberToMatch' field, in conjunction with 'GramsPerMatchUnit' for gram-based products, the promotion applies.

Once the promotion applies, the cheapest product in the bucket of consumed products gets the specified discount: Dollar, DollarOff, or PercentOff

Dollar indicates the dollar amount at which the product should be charged.
DollarOff indicates the dollar amount by which the product should be reduced.
PercentOff indicates the percentage discount by which the product should be reduced.

The most expensive LineItems on a transaction should be consumed by the promotion first, with the least expensive being consumed last. The discount always applies to the cheapest product in the bucket.

For example, if there is a transaction where all 5 line items match this promotion, with only 3 being consumed by this promotion, they should be consumed like this:

Product A : $10
Product B : $9
Product C : $8
Product D : $7
Product E : $6
^ all these products pass the 'ItemsToMatch' conditions on the promotion.

'NumberToMatch' in this case is 3, so only 3 of these products will be included in the promotion.
'DollarValueOfCheapest' in this case is $1, so the cheapest product will be sold for $1.

Product A, Product B, and Product E should be included in the promotion.
Product E will be sold for $1
Product A and Product B don't receive a discount.
Product A, Product B, and Product E are considered 'consumed' and cannot be used for other promotion calculations.

Say we change the 'NumberToMatch' to 2, from 3.
Now, the promotion applies twice, so there are going to be 2 products that are sold at a discount.
The promotion would not apply twice if the 'MaxApplicationCount' was 1.

For the first application, Product A and Product E are consumed. With Product E being sold for $1
For the second application, Product B and Product D are consumed. With Product D being sold for $1
Product C is not consumed and can be considered for other promotions.

As you can see, this promotion types consumes the most expensive products first when trying to find items that fulfill the 'ItemsToMatch' conditions. It then consumes the cheapest products when trying to find items that fulfill the 'ItemsToMatch' conditions to apply the specified discount.

MatchThenCheapestOtherFor

This PromotionType is like CheapestMatchedFor, except there are two condition trees. One for the products that enable the promotion, and another for the products that will get the discount specified by this promotion.

'MatchConditions' is a tree of conditions that must be met for a quantity of products to enable the discount specified by this promotion
'OtherItemConditions' is a separate tree of conditions that must be met by a product to receive the specified discount.

Once again, 'NumberToMatch' is used in conjunction with 'GramsPerMatchUnit' to determine what quantity of products need to meet the 'MatchConditions' before this promotion can apply.
Once that requirement is met, the cheapest product that's not already consumed by 'MatchConditions' that meets the 'OtherItemConditions' is given the discount.

The available discounts are the same as 'CheapestMatchedFor':

Dollar indicates the dollar amount at which the product should be charged.
DollarOff indicates the dollar amount by which the product should be reduced.
PercentOff indicates the percentage discount by which the product should be reduced.

'OtherItemConditions' only ever applies to a single quantity of a product. 'GramsPerMatchUnit' is used to convert a gram-based into an each-based product.

When consuming products for this promotion, the most expensive products are consumed first, with the least expensive being consumed first when attempting to find a product to receive the discount.

BundleFor

The 'BundleFor' PromotionTypes contain a list of 'BundleMatch' objects. Each 'BundleMatch' contains a ProductCondition tree and a corresponding quantity. This allows for the specification of bundles with many different requirements for each element in the bundle.

A bundle applies the discount to each element in the bundle. An element in a bundle is any product that matches the given 'ProductCondition' with the required 'QuantityToMatch'. 'GramsPerMatchUnit' can be used to convert a gram-based quantity into an each-based quantity.

Once there are enough quantities of each product for each element in the 'BundleItemsToMatch', the discount can be applied.

The discounts are as follows:
TotalDollarDistributed indicates the given discount amount should be distributed to all items in the bundle proportionally to their original price.
TotalDollarOffDistributed indicates the given discount amount should be taken from all items in the bundle proportionally to their original price.
PercentOff indicates the percentage amount by which each element in the bundle should be discounted.