MyDeal Universal API Documentation

Owner	MyDeal Tech team
Version	3.4
Last Updated date	30/05/2025
Last Update by	AB

Revision History

S.No.	Revision Description	Version	API version number	Revised By	Revised Date
1	Initial Version	1.0	1	HY	29/01/2018
2	Major Changes to Products endpoint: - Shipping related fields - ProductUnlimited (allow backorders) - Variations - Product listing status update Major Changes to Order endpoint – - Unfulfilled orders endpoint - Cancel Order endpoint - Fulfill order changes Added Pending Responses API Added integration process & API validation section	2.0	1	HY	16/04/2018
3	Added MyDeal Category ID Mapping Instructions in (0.12.1 Product models)	2.1	1	HY	27/08/2018
4	Added IsDirectImport flag to ProductGroup to allow sellers to specify product is direct imported or local	2.2	1	HY	10/10/2018
5	● IsDirectImport flag is now made mandatory ● ShippingCostCategory “FreeShipping” Enum is deprecated. Instead, Seller choose “Flat” with $0 ShippingCostStandard ● ShippingCostExpedited is deprecated. Currently, express shipping is not supported but it will be added in future releases ● Below optional fields are added to ProductGroup. - DeliveryTime - MaxDaysForDelivery - Has48HoursDispatch	2.3	1	HY	11/02/2019
6	1. /products/listingStatus endpoint is updated to allow only to	2.4	1	HY	30/05/2019

	discontinue product (making product not live) 2. New ShippingCostCategory option “FlatAnyQty” is added				
7	New field “UnitPriceExcCommission” added to OrderItems in all orders endpoints	2.5	1	HY	19/09/2019
8	New Field “CustomerDateOfBirth” added to Order in all order retrieval endpoints	2.6	1	HY	25/11/2019
9	Order Refunds API is implemented for partial and full refunds	2.7	1	HY	27/11/2020
10	Added RefundReason for Order Refunds API.	2.8	1	TT	10/02/2020
11	Updated section 0.5.4 Update Products Price and Quantity with an important note: For variant type products, all available variants (Buyable Products) must be present in the Productgroup when updates are sent	2.9	1	HY	03/09/2020
12	Updated: ● 0.3 removed ‘Different shipping for variant in a product’ from out of scope list ● 0.8 - 0.10 with refined integration process instructions ● 0.11.6 Other Models - Field: allow clients to send the image, weight and dimensions at variant level (Buyable Products).	3.0	1	TT	20/09/2021
13	1. New section ‘0.7 Categories’ added 2. Updates : 0.6 Orders -> ● added sub-section on OrderItems \ combined shipping ● Updated description for all order end-points to include behavior for combined orders 0.11 Recommended frequency for API calls ● Updated order fetch frequency from 60 mins to 15 mins 0.12.1 Product Models	3.1	1	KD\SB	10/06/2022

	● Removed support for RequestFreightQuote shipping option ● Removed support for CategoryName option in the categories[] ● Updated description for ProductUnlimited in the BuyableProducts model ● Removed height,width fields from the image[] 0.12.6 Other Models ● Added gtin and mpn fields to BuyableProducts metainfo (allowed names and values) Other sections : Refined content for better clarity - no technical changes				
14	Updated hyperlinks in 0.7 Categories 0.14.1 Integration Flow Diagram	3.2	1	KD & SB	22/10/2022
15	Added “OrderSource” to the Order Model Section 0.12.2	3.3	1	KD & SB	23/11/2023
16	Section 0.12.1 Product models; ● Added details on GTIN validity to ProductGroup Data Overview Section 0.12.6 Other models; ● Added details on GTIN validity to to Allowed Names and Values for BuyableProduct level Metainfo	3.4	1	AB	30/05/2025

Table of Contents

0.1 Overview

MyDeal Universal API solution is a service based infrastructure exposed via REST Web API for enabling business to business communication. Predominantly, these services are used to integrate external e-commerce platforms/channels to onboard their sellers who can sell their products, manage orders, and order fulfillments on MyDeal Marketplace.

External e-commerce platforms build their system to the API specifications mentioned in this document to integrate their system that enables below operations for each seller –

- Query products
- Create new Products
- Update existing products content
- Update Quantity and Price
- Update Publish status of a product to discontinue to sell on marketplace
- Retrieve orders placed in marketplace
- Retrieve Unfulfilled orders
- Fulfill orders with Shipment information
- Full Cancellation of unshipped orders
- Full or Partial Refund of orders
- Retrieve the list of MyDeal category IDs

0.2 General API requirements

MyDeal Universal APIs ensure specific requirements to meet for successful external platform integration.

- All API endpoints support API versioning. Sellers need to pass version header in request headers as “api-version” (default version is “1”), please refer API versioning
- Data communicated to and from API should be in JSON format
- Date and time fields should be UTC format in request and same communicated in response as well
- All APIs endpoints should require TLS secured
- APIs enforce Token based Authentication, hence external platforms should ensure valid bearer token generated before requesting endpoint operations. For detailed Authentication flow, please refer Authentication
- Any API endpoint operation requires each SellerID and SellerToken to be passed in RequestHeaders in order to verify their claims to allow specific operation. Please refer Integration Process section on the process of obtaining SellerID and SellerToken as part of integration
- Product Key Identifier - External Product ID or Product SKU needs to be the primary identifier for all product API calls (Product SKU is always mandatory but External Product ID is optional to pass based on availability in seller system) The MyDeal team will request sellers to confirm the primary identifier during the initial onboarding phase.
- Product SKU must be present on order line items
- The Product SKU must support a maximum length of 50 characters and containing all printable ASCII characters, including spaces
- It is advised to create products with a "MyDeal Category ID" in the product data, so that products get created in the right categories and get maximum exposure.

0.3 Out of Scope

Currently, below functionality is out of scope of this API document. However, integrating platforms can check with the MyDeal Integration Support team on handling those scenarios to complete full integration either by developing custom API or achieve using manual process.

- Partial Shipment of an OrderItem
- Partial Cancellation of an unshipped OrderItem
- Custom freight calculations (this is done by MyDeal support team upon seller requests as part of Customization/Validation phase)
- Onboarding Sellers onto MyDeal (this is done by MyDeal onboarding team, refer Integration Process)

0.4 Authentication & Authorization

All APIs are secured with TLS and token-based authentication.

Token Based Authentication:

All MyDeal APIs mentioned in this document are secured with Token based Authentication. Hence, consumers should generate their bearer tokens with their client credentials and pass the token in subsequent API requests.

0.4.1 External platform/Channel Authentication

External platforms/channels who want to communicate with MyDeal Universal APIs to onboard their sellers must do a one-time registration with MyDeal as an external platform and get their Client Credentials such as **Client Id, Client Secret key.**

These client credentials can be used in future to authenticate the API transactions.

Tokens can be generated as following request:

```
POST /mydealaccesstoken
```

Request Headers:

```
{
        "Content-Type":"application/x-www-form-urlencoded”
}
```

Request body:

```
{
        grant_type: client_credentials,
        client_Id: <clientId>,
        client_secret: <secret>
}
```

Response:

```
{
    "access_token": "1i4QfL8II-FR0W-Q_ x9TVDUf50gE…..",
     "token_type": "bearer",
```

```
        "expires_in": 3599
}
```

Use above access_token in subsequent API requests by passing a token in the Authorization Header.

In case authentication fails, API response should be sent as **BadRequest (400)** and returned with error.

ErrorID	*Example error message*
`AuthenticationFailure`	*ClientId Invalid/Client secret Invalid*

0.4.2 API Operation Level Authorization

Once Authentication token generated using client id and secret, authorization happens at API request level to authorize specific seller operation.

Each API operation is limited to a single Seller. So, in each API request for products, orders, fulfillments updates, below fields should be present in request headers along with bearer token to authorize the request for that specific seller.

HTTP Header	*Description*
SellerID	A unique, seller-specific identifier assigned by your system.
SellerToken	A unique, seller-specific token assigned by your system.

Note: SellerID, SellerToken can be obtained from MyDeal Integration team while onboarding specific Sellers. For more details, please refer to the below Integration process.

In the case of authorization failure, the API response should return with status code UnAuthorized (401) and one of the following ErrorID values.

ErrorID	*Example error message*
4000	Authorization Failed
4001	Invalid seller Token
4002	Invalid Seller ID

0.5 Products

Products API endpoints are used to manage (create new or update existing) products in the marketplace. In addition, product price & quantity and listing status can be updated on a frequent basis as agreed with MyDeal.

Each product is represented as a ProductGroup with one or more BuyableProducts associated with it. ProductGroup models can be found in Models.

MyDeal supports 2 types of Products - Standalone and Variant. Standalone products are single products without any variations like size or colour. Variants are products having multiple options like different sizes or colours.

ProductGroup contains key fields which are **ExternalProductId** and **ProductSKU.** *During Onboarding, integrating partners will be asked to confirm whether ExternalProductId(Id used to store product in seller system) can be supplied/used to identify a product or ProductSKU can be used when their platform can't send ExternalProductID.*

ProductGroup also contains other information common to product groups such as title, description, specifications and brand etc.

Each product group should have one or more BuyableProducts items to support standalone or variations to provide sku, price, quantity,product unlimited and shipping cost etc. Each BuyableProduct contains key fields which are **ExternalBuyableProductID** and **SKU.** ExternalBuyableProductID and SKU should follow the same rules mentioned above for ProductGroup.

More detailed description and data rules can be found in Models.

Standalone ProductGroup:

Standalone ProductGroup should satisfy below rules –

- **Only one** BuyableProducts item should be present in ProductGroup.
- ExternalProductID should be same as ExternalBuyableProductID at Buyableproducts level if available
- ProductSKU should be same as SKU at Buyableproducts level
- There should not be any options available at Buyableproducts level

For E.g.,

Variation ProductGroup:

A Product can be considered as a variation when below rules met –

If One Buyable Product available -

- ExternalProductID should be different from ExternalBuyableProductID at Buyableproducts level if available
- ProductSKU should be different from SKU at Buyableproducts level
- At least one option is mandatory at the Buyableproducts level (e.g. size or colour)

If more than one Buyable Product is available -

- Buyable Product SKU different from ProductSKU in the ProductGroup
- There should be options available at each Buyableproducts level
- OptionName should be consistent across all buyable products

Each BuyableProduct item represents a variant of the product containing SKU, price, quantity, variation Options.

For E.g.,

```
[
{
"ExternalProductId": "1234" ,
"ProductSKU":"POLO-SHIRT",
 "Title: "Sample Title",
"Description":"Sample Description",
"BuyableProducts" : [
                                                                                    {
                                                                                    "ExternalBuyableProductId": 2345,
                                                                                     "SKU" : "POLO-SHIRT-SMALL",
                                                                                    "Price" : 100,
                                                                                    "Quantity" : 10,
                                                                                    "Options":[ { "OptionName":"Size",
          "OptionValue":"Small", "Position":1 }]
                                                                                     },
                                                                                    {
                                                                                    "ExternalBuyableProductId": 2346,
                                                                                     "SKU" : "POLO-SHIRT-MEDIUM",
                                                                                    "Price" : 100,
                                                                                    "Quantity" : 10,
                                                                                    "Options":[ { "OptionName":"Size",
          "OptionValue":"Medium", "Position":1 }]
                                                                                        }
                                                                                    ]
 }
]
```

0.5.1 Get Products

```
GET /products
```

This Endpoint can be used to retrieve a list of products for a specific seller based on the filter criteria passed.

Endpoint:/products

Method: Http GET

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
listingStatus	String (comes from ListingStatus Enum values)	All (allowed values : Live, NotLive)
page	integer	1
Limit	Integer	100 **(max 250, but can be configured at the time of integration**)
Fields	String	

Fields parameters will be used to filter the fields and give a partial response. Fields follow below format:

`Field1, Field2, Field3(ChildField1, ChildField2)`

For e.g.,

`ExternalProductId,ProductSKU,Title,Description,Images(Src),BuyableProducts(ExternalBuyableProductID,SKU,Price,Quantity)`

Note: If the field parameter is empty, the response will be empty.

Request Body: None

Response:

Response will be **ActionResponse** , where Data in ActionResponse will be an Array of **ProductGroupResponse** items that can be standalone or variant product groups.

Please refer to the ProductGroupResponse Model.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": [ProductGroupResponse Array]
  "Errors":[]
}
```

0.5.2 Get Single Product

GET /products/{idorsku}

This Endpoint can be used to retrieve a single product based on ExternalProductID or ProductSKU. Sellers should pass the appropriate value based on the configuration set for Product Key Identifier.

Endpoint:/products/{idorsku}

Method: Http GET

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

If Primary Key is ExternalProductID :

Parameter Name	datatype	Default value
id	string	mandatory to pass

If Primary Key is ProductSKU:

Parameter Name	datatype	Default value
sku	string	mandatory to pass

Request Body: None

Response:

Response will be **ActionResponse**, where Data in ActionResponse will be a single **ProductGroup** item that can be standalone or variant type.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": ProductGroup
  "Errors":[]
```

}

0.5.3 Create or Update Products

```
POST /products
```

This Endpoint can be used to create new products or update existing products content based on their availability in the marketplace database.

Existence of the product is checked based on Product Key identifier, which is ExternalProductID or Product SKU at ProductGroup level and ExternalBuyableProductID or SKU at BuyableProducts level respectively.

Long running batch flow:

This endpoint will not immediately create/update products but it will put the batch of requested items into a workflow, which is a long running batch process that may include manual verification process.

Hence, response status from this endpoint always will be ' **AsyncResponsePending'** with pending Uri that consists of batch work item id for e.g. /pending-responses?workItemId=100

Sellers may need to poll this **PendingUri** to check for the latest status of work item to find the status of those products as part of the work item and find whether they are live or not. If any product fails to create/update due to errors, the seller needs to rectify those errors and resend the products.

Please refer to the pending-responses endpoint for more information.

Important notes for updates:

- As this endpoint is used to update products as well, Seller needs to send delta product updates which are changed in their system with regards to product content, shipping related information, variations, images, brand, categories etc

- If products have only price stock updates, it is advised to use PriceStock endpoint '/quantityprice' to refresh price and stock because that endpoint performs real time updates to products and instantly reflect in marketplace, whereas /products endpoint goes through async batch workflow and that leads to delays in updates.

- **When a discontinued product receives an update via the /products API call, it will automatically be re-listed (discontinue flag becomes FALSE).**
- **In order to discontinue a product, '/products/listingstatus' endpoint should be used.**
- **Newly created products status will automatically be set to Live status and there is no need to send status update via '/listingstatus' endpoint**

For detailed fields and their significance, please refer to the ProductGroup.

Endpoint:/products

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters: None

Request Body: request body should contain an array of ProductGroup items. Items can be standalone or variant product groups. Please refer to the ProductGroup Model for examples.

Note: It is recommended that products are sent in batches. Maximum number of products per request is 250 but is configurable at time of integration. If request exceeds count, it will not be processed and API will return 'BatchCountExceeded' Error.

```
[{
"ExternalProductId": "1234",
"ProductSKU": "POLO-SHIRT",
"Title: "Sample Title",
"Description":"Sample Description",
"BuyableProducts" : [
{
"ExternalBuyableProductId":2345,
"SKU" : "POLO-SHIRT-SMALL",
"Price" : 100,
"Quantity" : 10,
"Options":[ { "OptionName":"Size",
"OptionValue":"Small", "Position":1 }]
},
{
"ExternalBuyableProductId":2346,
"SKU" : "POLO-SHIRT-MEDIUM",
"Price" : 100,
"Quantity" : 10,
"Options":[ { "OptionName":"Size",
"OptionValue":"Medium", "Position":1 }]
}]
}]
```

Response:

Response will be **ActionResponse**, with status “AsyncResponsePending” along with PendingUri to poll by seller.

HttpStatusCode - 200

```
{
  “ResponseStatus”: “AsyncResponsePending”,
  "Data": null,
  “Errors”:null,
  “PendingUri”:”http://<APIURL>/pending-responses?workItemId=<id>”
}
```

0.5.4 Update Products Price and Quantity

POST /products/quantityprice

This Endpoint can be used to update the price and quantity of existing products based on their availability on the marketplace.

Existence of the product is checked based on Product Key identifier, which is ExternalProductID or Product SKU at ProductGroup level and ExternalBuyableProductID or SKU at BuyableProducts level respectively.

Important Note: Whole product with all of its variants’ quantity and price information should be sent to this endpoint. For variant type products, ALL available variants(Buyable Products) must be present in the Productgroup even if there are changes to only a few variants’ price or stock. If any variants are missing from the product group, that variant will be treated as Out of Stock and not available to purchase in MyDeal.

Endpoint: /products/quantityprice

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to ‘1’)

Request Query string parameters: None

Request Body: request body should contain an array of ProductGroup items. Items can be standalone or variant product groups.

Seller doesn’t need to pass all info in ProductGroup item for this endpoint. Instead, below fields are ONLY expected to be present in request and other fields will be ignored automatically.

At ProductGroup level, below fields are required to be passed in.

- ExternalProductID
- ProductSKU

At BuyableProducts level, below fields are required to be passed in.

- ExternalBuyableProductID
- SKU
- **Price**
- RRP
- **Quantity**
- **ProductUnlimited** (If ProductUnlimited is true, Quantity will be ignored. This means that the seller has an unlimited quantity of the product. If ProductUnlimited is false, quantity is mandatory and quantity gets updated in the system when the API call is received. It is recommended that sellers provide correct stock numbers and use ProductUnlimited = False)

Note: The number of products in a batch can be configured at the time of the integration and the maximum number of products per request is 250. If request exceeds count, it will be not processed and 'BatchCountExceeded' error sent.

Response:

Response will be **ActionResponse**, where Data in ActionResponse will be an array of **ProductGroupResponse** items. Please refer to the **ProductGroupResponse** model.

HttpStatusCode - 200

{

```
"ResponseStatus": "Complete",
"Data": [ProductGroupResponse array],
```

```
  "Errors":[]
}
```

0.5.5 Update product listing status

POST /products/listingstatus

This Endpoint can be used to update products' listing status to unpublish or discontinue them on marketplace.

Either the whole product or a single variant of the product can be discontinued through this endpoint.

Note: This endpoint does not support continuing\re-listing a discontinued product. So, to relist a discontinued product, sellers need to send that product in the POST /products endpoint as an update to the product.

Endpoint:/products/listingstatus

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters: None

Request Body: request body should contain an array of ProductGroup items. Items can be standalone or variant product groups.

Seller doesn't need to pass all info in ProductGroup item for this endpoint. Instead, below fields are ONLY expected to be present in request and other fields will be ignored.

At ProductGroup level, below fields are required to be passed in.

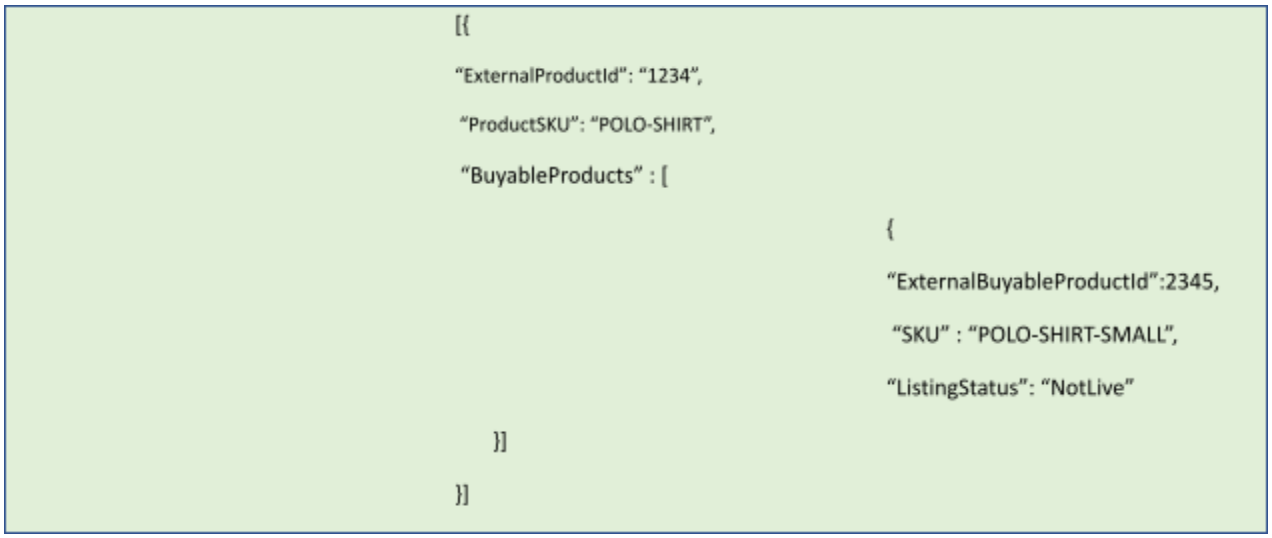
- ExternalProductID
- ProductSKU

For those BuyableProduct items which need to be discontinued or unpublished, below fields are required to be passed in.

- ExternalBuyableProductID
- SKU
- ListingStatus ('**NotLive' to discontinue on marketplace**)

Note : To discontinue the whole ProductGroup, all buyableproducts should be included in the API call.

Note: Maximum number of products per request is 100 but is configurable at time of integration. If request exceeds count, it will be not processed and BatchCountExceeded error will be returned.



Response:

Response will be **ActionResponse**, where Data in ActionResponse will be an array of **ProductGroupResponse** items. Please refer to the **ProductGroupResponse** model.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": [ProductGroupResponse Array],
  "Errors":[]
}
```

0.5.6 Pending Responses

```
GET /pending-responses?workItemId=<workitemid>
```

This Endpoint can be used to query the status of a long running batch that is submitted by a seller , especially CreateorUpdateProducts endpoint (/products) to create or update products.

If the Work Item is still processing or under reviewal workflow, it gives back a response with "AsyncResponsePending" along with PendingUri until the reviewal process completes.

If the work item is finished processing either success or failed, it gives a response accordingly.

Endpoint:/pending-responses?workItemId=<id>

Method: Http GET

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters: WorkItemId

Request Body: None

Response:

If the work Item is still processing, the response below will be sent.

{

"ResponseStatus": "AsyncResponsePending",

"PendingUri" :"pollingURl"

}

If the work Item is complete or complete with errors, below response will be sent. Detailed errors will be given at each ProductGroup level.

Response will be **ActionResponse**, where Data in ActionResponse will be an array of **ProductGroupResponse** items. Please refer to the **ProductGroupResponse** model.

HttpStatusCode - 200

{

```
"ResponseStatus": "Complete",
"Data": [ProductGroupResponse Array],
"Errors":[errors if any]
```

}

0.6 Orders

Orders API endpoints are used for retrieving unfulfilled orders, order acknowledgement, order fulfillment and order cancellation.

Below are the operations exposed by Order API.

- Retrieve successfully purchased orders from marketplace based on filtering criteria
- Retrieve ready to fulfill orders for further fulfillment by seller
- Acknowledge those orders to market place, which are successfully retrieved for fulfillment
- Update the fulfillment status and shipping information
- Full Cancellation of unshipped order that seller can't fulfill as a whole
- Full or partial refund of Orders

Each order in the marketplace has the following OrderStatus.

- **ReadytoFulfill :** Orders that are successfully processed by the marketplace and are ready to be fulfilled by the seller. External platforms who want to retrieve new orders that needs fulfillment, need to query /Orders/unfulfilled GET API

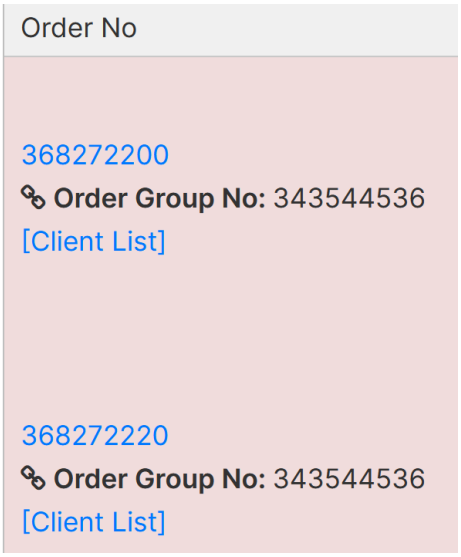
- **SellerAcknowledged:** Orders that are successfully retrieved and acknowledged by Seller that they are in process of fulfillment
- **Shipped:** Orders that are fulfilled successfully and shipping information is updated for all OrderItems in the order
- **Refunded:** Orders that are fully refunded to customer
- **All:** All Orders that match any of the above four statuses

OrderItems:

By default, an order contains 1 OrderItem. If sellers have configured either Combined Shipping or Shipping Offers , orders that satisfy the combined shipping \ shipping offer rules will contain more than 1 orderItem.

For example: Assume the seller has configured a shipping offer of “free shipping'' when the customer buys products from his store totalling $100.

1. If the customer purchased 2 products from the seller totalling $100, then it will translate to **1 order having 2 OrderItems**.
   Example : OrderItems 368272200 and 368272220 belonging to the same order 343544536



Terminology on Marketplace Portal: On the MyDeal marketplace portal (snapshot shown above), the OrderID in the API response is called the “Order group No” and the OrderItemID in the API response is called the “Order No”

2. If the customer purchased 2 products, but the total price is less than $100, then it will translate to **2 separate orders with 1 orderItem** each.
   Example : OrderID X with OrderItemID Y and OrderID M with OrderItemID N

Note: When a customer does a purchase with multiple items in the cart, it could translate to a

- single order with multiple orderItems OR
- multiple orders with single orderItems OR
- a combination of these.

An order with multiple orderItems will be created only when :

1. sellers have configured combined shipping and/or shipping offers AND
2. the products added to cart meet the combined shipping / shipping offer rules set by the seller.

0.6.1 Retrieve Orders

```
GET /orders
```

This Endpoint can be used to retrieve orders of a specific seller based on Order status that is passed in.

Please refer to the OrderStatus Enum in Models.

Endpoint:/orders

Method: Http GET

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
orderStatus	String (comes from OrderStatus Enum values)	All
Page	Integer	1
Limit	Integer	100 (max 250, can be configurable during integration)

Request Body: None

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be an array **of Order** items.

Each Order will contain at least one Order item. Please refer to the **Order** model.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": [Order array],
  "Errors":[]
}
```

0.6.2 Retrieve Single Order

```
GET /orders/{id}
```

This Endpoint can be used to retrieve orders of a specific seller based on Order id passed in. This OrderId is a unique id generated by the marketplace.

Endpoint:/orders/{id}

Method: Http GET

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
id	numeric	NoneIt's a Mandatory field

Request Body: None

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be a single **Order** with one or more OrderItems. Please refer to the **Order** model.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": [Order],
  "Errors":[] }
```

0.6.3 Retrieve Unfulfilled Orders

```
GET /orders/unfulfilled
```

This Endpoint can be used to retrieve orders that are ready to be fulfilled but not acknowledged by the seller. This endpoint retrieves unfulfilled orders sorted by their order placed date.

This endpoint will always retrieve unfulfilled orders if orders are not acknowledged accordingly. Sellers need to import these orders into their system and acknowledge each of them.

Note: Seller has to acknowledge each order that is retrieved from this endpoint using ACK endpoint

Endpoint:/orders

Method: Http GET

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
Limit	Integer	100 (max 250, can be configurable during integration)

Request Body: None

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be an array of orders.

Each Order will contain at least one Order item. Please refer to the **Order** model.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": [Order array],
  "Errors":[]
}
```

0.6.4 Acknowledge Order

POST /orders/{id}/acknowledge

This Endpoint can be used to acknowledge an order which is successfully retrieved by the seller for further fulfillment.

Once this endpoint is called and the API response is a success, status of all OrderItems belonging to that order will be updated to "**SellerAcknowledged**" and hence it will not be retrieved from '/orders/unfulfilled' endpoint in the next pull by the seller.

Endpoint: /orders/{id}/acknowledge

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
id	numeric	None. It's a Mandatory field

Request Body: None

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be True /False.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": True (or) False
  "Errors":[]
}
```

0.6.5 Create/Update Order Fulfillment(s)

```
POST /orders/fulfill
```

This Endpoint can be used to update fulfillment status and shipping details of orders that are successfully shipped by the seller.

In the request, each order can contain one or more OrderFulfillment Items with shipping details.

Note: Currently partial shipment of an order with 1 orderitem is not supported.

However, for orders with multiple OrderItems, fulfillment of individual OrderItems is supported.

Endpoint:/orders/fulfill

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters: None

Request Body:

Requests can contain one or more OrderFulfillment Items. Each OrderFulfillment Item represents one order with one or more Order fulfillment line items in it with shipping information.

Note: Maximum number of orders per request is 100 but is configurable at time of integration. If request exceeds count, it will be not processed and BatchCountExceeded error will be returned.

```
[
 {
  "OrderId": 1111,
  "FulfillmentItems": [
   {
    "OrderItemId": 1111,
     "SKU": "POLO-SHIRT-SMALL",
    "DispatchedDate": "2018-01-16 11:19:53",
    "DispatchCarrier": "AUPost",
    "TrackingCode": "AU12121"
   }
  ]
 }
]
```

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be a single **OrderFulfillmentResponse** item with result. Please refer to the **OrderFulfillmentResponse** model.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": [OrderFulfillmentResponse],
  "Errors":[]
}
```

0.6.6 Cancel Order

```
POST /orders/{id}/cancel
```

This Endpoint can be used to cancel an order which cannot be fulfilled by the seller as a whole. Hence, this endpoint expects the orderId and quantity(no significance as of now because system cancels full order)

Note:

a. **Sellers must use this endpoint for canceling orders which are not shipped at all (due to out of stock or any other issues). This request will be internally processed as a full refund to the customer.**
b. **For undispatched orders with a single OrderItem, the system currently supports Full Order Cancellation only.**
c. **For orders with multiple OrderItems, some of the OrderItems can be canceled. The cancellation reason should be consistent for all the OrderItems that are to be canceled.**

Once this endpoint is called, the specified OrderItems will be canceled and order will be forwarded to the internal refund queue.

Endpoint: /orders/{id}/cancel

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
id	numeric	It's a Mandatory field

Request Body:

Requests contain OrderCancellation Item that have one or more orderItemCancellation items.

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be OrderCancellationResponse.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": OrderCancellationResponse
  "Errors":[]
}
```

0.6.7 Refund Order

```
POST /orders/{id}/refund
```

This Endpoint can be used to refund a dispatched order fully or partially.

Note: This endpoint accepts Refund Amount instead of Quantity of the order. Hence, sellers preferably use this endpoint for refunding an order based on amount instead of quantity, mostly in

case of fully or partially shipped Orders. To refund unshipped Orders, sellers must use /cancel endpoint.

Once this endpoint is called, based on the refund amount for the product and/or shipping amount, MyDeal system will calculate the refund and mark the order as Fully Refunded or Partially Refunded. So, for the seller, it is just the amount that they want to refund to the customer.

In the case of orders with multiple OrderItems,

1. Individual OrderItems can be refunded separately if needed
2. If the API request contains multiple OrderItems, the RefundReason should be consistent across all the OrderItems

Endpoint: /orders/{id}/refund

Method: Http POST

Request Headers: SellerID, SellerToken & api-version(optional and default to '1')

Request Query string parameters:

Parameter Name	datatype	Default value
id	numeric	It's a Mandatory field

Request Body:

Requests contain OrderRefund Item that have one or more OrderItemRefund items.

Response Body:

Response will be **ActionResponse**, where Data in ActionResponse will be OrderRefundResponse.

HttpStatusCode - 200

```
{
  "ResponseStatus": "Complete",
  "Data": OrderRefundResponse
  "Errors":[]
}
```

0.7 Categories

```
GET /categories
```

This endpoint returns the list of MyDeal categories and their IDs. Products can be tagged to those categories which have "IsAssignable: TRUE" in the response. The categoryID obtained from this API call

can be used in the /products call (Categories -> CategoryID field) to categorize the products while they are being created.

Endpoint: /categories

Method: Http GET

Request Headers: None (no authentication required)

Request Query string parameters: None

Request Body: None

Response Body: Data response in the following format


```
[
  {
    "ParentID": "NULL",
    "CategoryID": 2608,
    "CategoryName": "Appliances",
    "IsAssignable": false
  },
  {
    "ParentID": 2608,
    "CategoryID": 2609,
    "CategoryName": "Air Conditioners",
    "IsAssignable": true
  }
]
```


It is recommended that the categories are refreshed daily. MyDeal team will also send an update to channels/sellers when there are major changes to category hierarchy.

Alternatively, the MyDeal category list is available via an excel file that can be downloaded from https://assets.mydeal.com.au/content/marketplace/MyDeal_Product_Category_List.xlsx

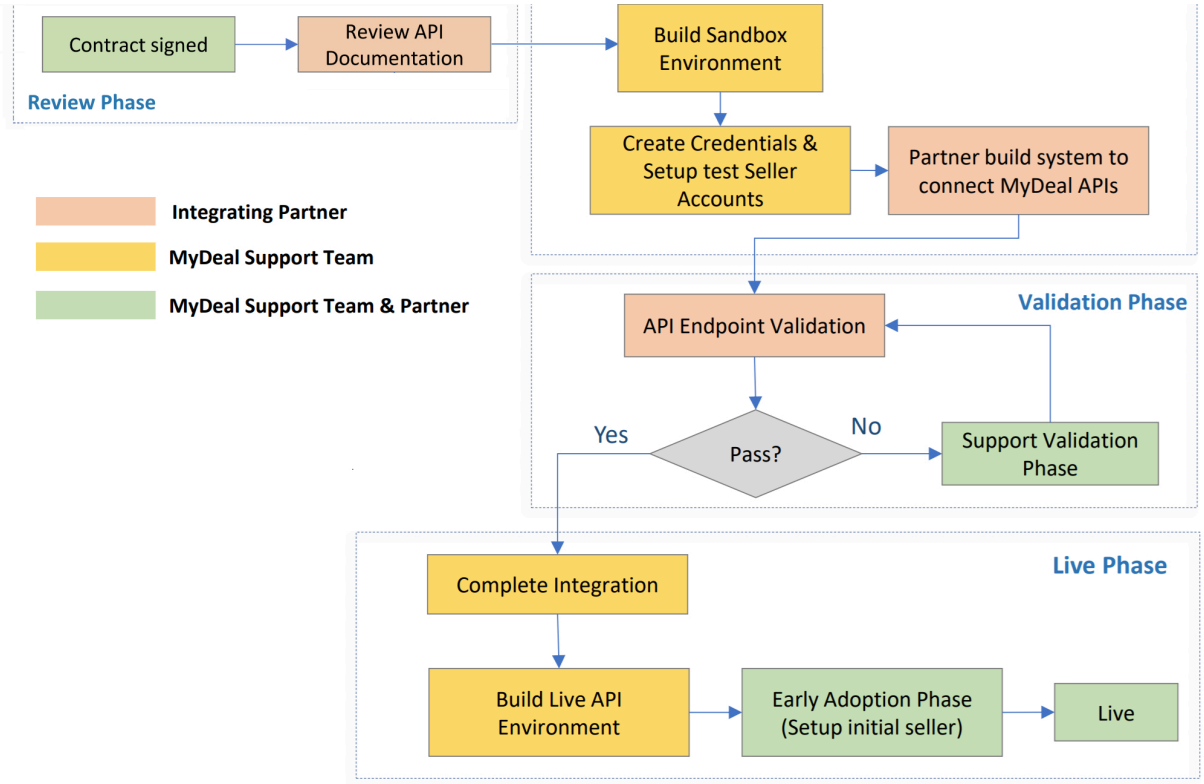
0.8 API versioning

All API endpoints support versioning. “api-version” header needs to be passed in request headers along with other authentication headers. By default, “api-version” defaults to “1”.

For each new API upgrade/release, a new version may be released and it will be communicated to all sellers so that sellers can make amendments in their system to support the new version. However, the current version will be supported until the seller uses the new version.

If an API release has any breaking changes, It will be communicated to the seller to quickly make their changes to support the new version and test it in the sandbox environment. Once the seller confirms, a new version will be rolled out to live and the old version will be deprecated.

0.9 Integration process



Here are the steps involved in successfully integrating your store via the MyDeal API.

Step 1 - Select your Product Key Identifier: ExternalProductID or ProductSKU

- This value must be unique per product and will be used for all API calls
- If you have a unique SKU for each product, use ProductSKU as Product Key Identifier for the MyDeal API Integration
- If you may have duplicate SKUs across your products, use ExternalProductID as Product Key Identifier for the MyDeal API Integration

Step 2 - Confirm your Shipping Arrangement on MyDeal

- Find all available shipping options described in this article: Setting up Shipping for your Products

Step 3 - MyDeal will set up your store in the Sandbox Environment and provide your Sandbox API credentials.

- Sandbox API URL = [*to be provided by the MyDeal Team]*
- Sandbox ClientID = [*to be provided by the MyDeal Team]*
- Sandbox ClientSecret = [*to be provided by the MyDeal Team]*
- Sandbox SellerID = [*to be provided by the MyDeal Team]*
- Sandbox SellerToken = [*to be provided by the MyDeal Team]*
- Sandbox Store URL = [*to be provided by the MyDeal Team]*

Step 4 - Once you have received your Sandbox API credentials, please test each of the following API Integration Flows in the Sandbox Environment. See section 0.9 for more details on product and order validations.

- Authentication (Endpoint: '/mydealaccesstoken')
- New product creation and content update (Endpoint: '/products')
- Product price and stock update (Endpoint: '/products/quantityprice')
- Fetch and acknowledge orders (Endpoint: '/orders', '/orders/acknowledge')
- Fulfill orders (Endpoint: '/orders/fulfill')
- Order cancellation and refund (Endpoints: '/orders/{orderId}/cancel', '/orders/{orderId}/refund')

NOTE:

- Pre-filled Postman Scripts to test the integration flows in our Sandbox Environment will be provided via email.
- Refer this spreadsheet to see all the test scenarios that have to be run on the sandbox. Please enter answers to the questions in each of the test scenarios as you run the tests and return the spreadsheet to the MyDeal team, so that the MyDeal team can verify your test results.
- Test Credit Card Details for *Order Integration Testing* will be provided by email. This step can only be done once you've created Products via the API. Please ensure you are only purchasing products from your store, otherwise, you will encounter errors.
- To fast track testing please set up your products with 'Free' / 'Flat' shipping arrangement in the Sandbox Environment. *To list your products with a Freight Calculator, you can create your freight schemes via the marketplace portal when you are ready to integrate to our Live Environment.*

Step 5 - Please notify the MyDeal team once you have tested all the API integration flows in the Sandbox Environment. The MyDeal team will verify your test results and provide any feedback.

Step 6 - Once all tests have been validated, MyDeal will create your store in the Live Environment and provide your Live API credentials

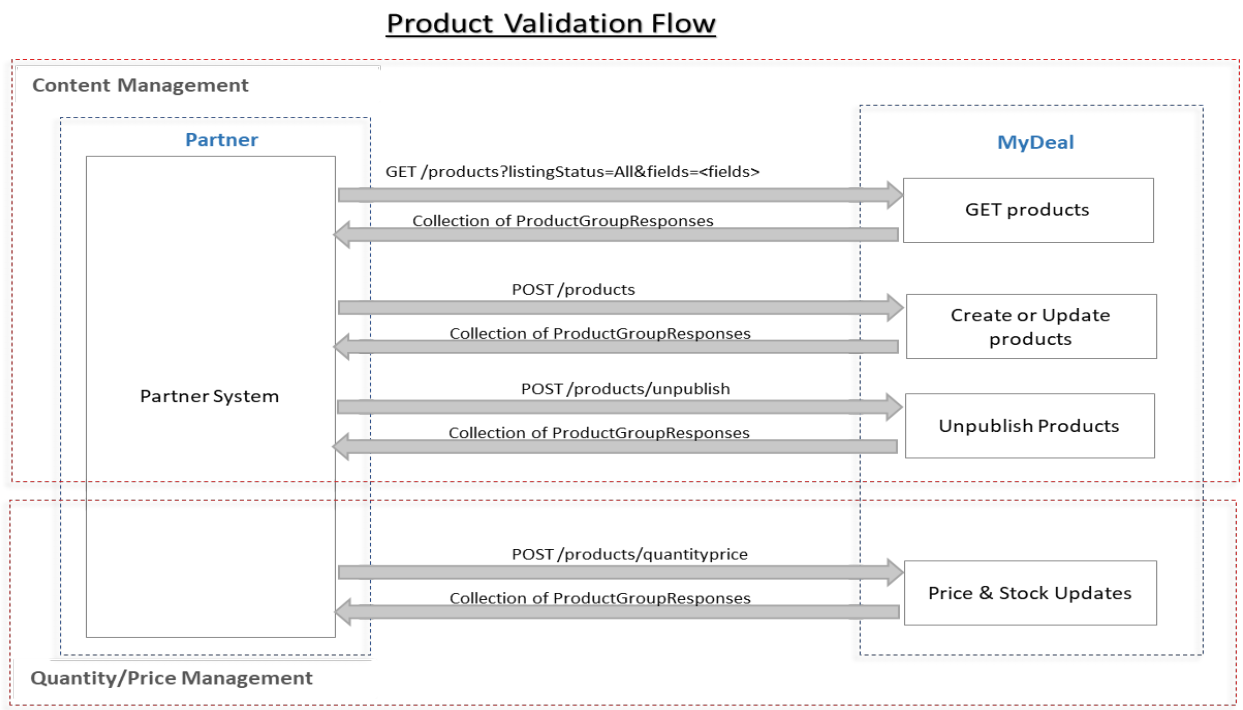
- Live API URL = [*to be provided by the MyDeal Team]*
- Live ClientID = [*to be provided by the MyDeal Team]*
- Live ClientSecret = [*to be provided by the MyDeal Team]*
- Live SellerID = [*to be provided by the MyDeal Team]*
- Live SellerToken = [*to be provided by the MyDeal Team]*
- Live Store URL = [*to be provided by the MyDeal Team]*

Step 7 - Publish your products in the Live Environment to start selling on MyDeal!

- To list your products with your *Shipping Rate Table Calculator*, apply the following arrangements per product:
    - ShippingCostCategory = 'Custom'
    - CustomFreightSchemeID = <will be auto generated when you create your freight scheme via the marketplace portal >

0.10 Product and order validation flows

0.10.1 Product validation flow



Content Management process:

Content management refers to keeping the product details (title, images, description and other text fields) up to date and in-sync between MyDeal and the integrating system. Content management process should happen on a schedule (say every 6 hours) at the partner system or as agreed with MyDeal.

1. **GET Products** - The integrating system should first request a full catalog of products that exist in the marketplace in case they want to reconcile differences between the two systems based on their system logic. This step can be skipped when onboarding for the first time with new products.

This step can also be skipped if the integrating system maintains delta updates based on their own system logic and does not have the need to pull a full catalog of products from MyDeal.

Note: All products belonging to the seller (whether live or not) will be returned in the response.

2. **Create/Update Products** – The integrating system can push new products and\or update existing via the /products endpoint. Existing products should be updated ONLY when there has been changes to the products since the last time the content management process ran. MyDeal system will process the API request and changes will be reflected on MyDeal website if the API call succeeds. Should any of the products fail validation, API will return a list of failed products with the associated errors.
   !!! IMPORTANT NOTE : Products should be sent in batches. Maximum batch size is 250.

   Note: Create/Update endpoint creates an asynchronous batch process in the MyDeal system. Hence, immediate response would be the work item id. This Id needs to be saved in the partner system to check the status of the work item to find whether products were pushed successfully or if any error has been returned.

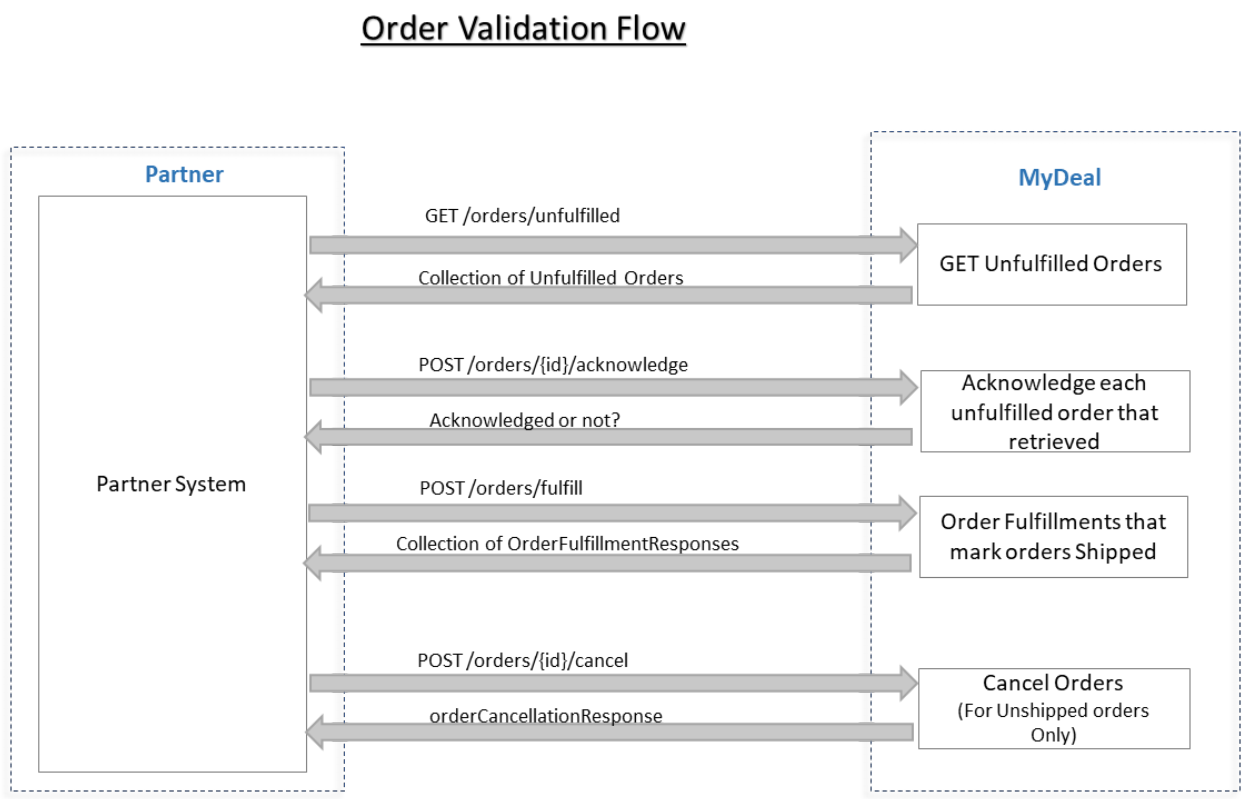
3. **Unpublish Products -** Upon reconciling differences between MyDeal and their system, if products have to be discontinued on MyDeal (for example, due to out of stock conditions), the integrating system should send a list of products that have become unavailable in their system. Those products would be marked as discontinued in the marketplace and will no longer be available for buying.

Quantity and Price Management

Quantity and price should be updated on a more frequent basis (say every 30 mins) due to the critical need for keeping quantity and price up-to-date. Frequency can be configured as per partner's preference.

Integrating system should send a list of only those products that have price and stock updates from the last update. Upon successful response, price and stock are directly updated in the product listing on MyDeal. Should there be any errors, the API will return a list of failed products with the associated errors.

0.10.2 Orders validation flow



Order Management

1. **Retrieve Unfulfilled Orders –** Integrating systems should first request a list of orders with the status "ReadyForFulfillment". These orders are successfully placed orders that are ready for shipment by sellers.
2. **Acknowledge order -** For each order that is imported into the partner system, they should send an order acknowledgement to MyDeal, indicating a successful import.
   Note: If order is not acknowledged, it will continuously appear in Unfulfilled Orders retrieved in step 1 above.
3. **Update Shipment Info –** Integrating system can send shipment updates to MyDeal for all shipped orders. This will update orders as “Dispatched” along with tracking code and dispatch carriers .
4. **Order Cancellation –** the integrating system can send cancellation requests for orders that the seller cannot fulfill.
5. **Order refund -** the integrating system can partially or fully refund ‘shipped’ orders

0.11 Recommended frequency for API calls

Operation	Suggested frequency \ values	Details
Product content Update Frequency	6 hours	Frequency to push product content updates. It is expected that the integrating platform sends only delta updates and not the entire product catalog.
Quantity and Price Update Frequency	30 mins	Frequency to update quantity and price information. It is expected that the integrating platform sends price\stock updates for only those products whose price\stock on MyDeal are not the same as that on the integrating platform.
Unfulfilled Orders Retrieval frequency	15 mins	Frequency to fetch 'ready to fulfill' orders from marketplace
Batch Size – Create/Update Products	250	Maximum number of products per batch that should be sent in a /products API call to create new products or update existing ones. **It is highly recommended that products are sent in batches.**
Batch Size – Order Fulfillments	100	Maximum number of orders per batch that should be sent via order fulfillment API
Batch Size – Unfulfilled Orders Retrieval	250	Maximum number of orders per batch that will be returned by MyDeal when the integrating platform fetches unfulfilled orders

0.12 API Models

Below are API Models that are used in Request or Response.

Products:

ProductGroup	BuyableProducts	Category
Option	Image	MetaInfo
ProducGroupResponse	BuyableProductResponse	

Orders:

Order	OrderItem	Address
OrderFulfillment	OrderFulfillmentItem	OrderFulfillmentResponse
OrderCancellation	OrderItemCancellation	OrderCancellationResponse

Other:

ActionResponse	Error	Field

0.12.1 Product models

ProductGroup:

This is a single product group with one or more BuyableProduct items to sell on the marketplace. ProductGroup contains at least one BuyableProduct item irrespective of type of the product.

If **Product is standalone without any variations**, only one BuyableProduct item is available in a product group.

If **Product is Variant** with more than one variation, each BuyableProduct's ExternalBuyableProductId and SKU will be unique and different from product group level Id and SKU.

Important note:

ExternalProductId or ProductSKU is used to uniquely identify a product. Similarly, BuyableProductId or SKU should be used at BuyableProducts level.

Shipping Fields:

ShippingCostCategory:

This field is mandatory and specifies the type of shipping for a product. Please refer to this help article to know about the various shipping options available on MyDeal.

ShippingCostStandard:

1. If ProductGroup ShippingCostCategory is "Flat" or "FlatAnyQty", ShippingCostStandard field is required else product fails validation step.
2. "FlatAnyQty" should be used only in case the product shipping cost is a flat amount irrespective of the quantity of units ordered.
3. FreeShipping - Product can be set as "free shipping" when "ShippingCostCategory" is "Flat" and "ShippingCostStandard" as "0". **Currently, we support Standard Shipping ONLY.**
4. If ProductGroup ShippingCostCategory is "Custom", ShippingCostStandard is ignored.

CustomFreightSchemeID:

CustomFreightSchemeID is required when ShippingCostCategory='Custom'. *If you want to list your products with a Freight Calculator, you can create your freight schemes via the marketplace portal when you are ready to integrate to our Live Environment. CustomFreightSchemeID can be obtained from the marketplace when the freight scheme is created.*

a. For sandbox testing, it is recommended to use FREE or FLAT shipping for simplicity.

ShippingCostCategory	ShippingCostStandard	Description
Flat	0	Product has Free Shipping
Flat	greater than 0	Product has flat rate shipping Australia wide per-item

FlatAnyQty	greater than 0	Product has flat rate shipping Australia wide irrespective of quantity ordered
Custom	-	Product has a freight calculator. CustomFreightSchemeID should be provided. ShippingCostStandard will be ignored.

Categories

Sellers can categorize their products by sending a MyDeal CategoryID in the /products API call. The list of MyDeal categories can be obtained from the /categories API call. The categoryID should be sent in the Categories[] -> CategoryID field of the /product API call.

Note :

a. It is required to assign products with a MyDeal Category ID when they are created. Please select the most appropriate category as far down the hierarchy as possible and the product will automatically be tagged up the hierarchy. Refer to our category tagging guide here.
   i. For example, if categoryID is 3213 (Appliances > Kitchen Appliances > Mixers > Hand Mixers) , the product will get tagged to Appliances, Kitchen Appliances, Mixers and Hand Mixers automatically.
b. Once categorized, the category of the product cannot be updated.All category updates are ignored. Sellers must contact MyDeal team for any changes required to the category after creation.

ProductGroup Data Overview:

Field	Data type	Required	Description
ExternalProductId	string	Conditional	External product Id stored as part of the seller channel, which is conditional. Typically, this field should be the same for a single product throughout its lifetime. If there is a change in ExternalProductId, the marketplace will treat it as a new product. This field is mandatory when the seller chooses “ExternalProductID" as the unique identifier for the products
ProductSKU	String	Required	Product Unique SKU
Title	string	Required	Product title or product name. Maximum 200 chars
Description	string	Required	Product Description. Can contain Html. PDFs and images included in the description will be downloaded and stored in the MyDeal servers.
Specifications	string	Optional	Product specifications/fine print. This can contain html.
Brand	string	Optional	brand name of the product if applicable

Tags	string	Optional	comma delimited tags used to filter or search products, say search keywords
Condition	string	Optional	new / used / refurbished. Default value is 'New' if not provided
Images	Image[]	Required	Image urls collection For variant products, all variant images including the ones specified in variationimageurl must be included in this collection.
Categories	Category[]	Required	Category collection Note : - Though category is a collection, categoryID is the only supported option - Currently CategoryID is an array but the system considers only the first value sent in the "CategoryID" field.
Weight	Decimal	Optional	Product weight
WeightUnit	String	Optional	Weight unit. Default to kilograms
Length	Decimal	Optional	Product length
Height	Decimal	Optional	Product height
Width	Decimal	Optional	Product width
DimensionUnit	String	Optional	Product dimension unit. Default to cm. Applies to Length, Height and width fields.
GTIN	String	Optional	Global trade item number, include UPC, EAN (in Europe), JAN (in Japan), and ISBN. Must be a valid 8, 12, 13 or 14 digit GTIN.
MPN	String	Optional	Manufacturer part number
ShippingCostCategory	String	Required	Used to specify how to calculate shipping cost. Shipping cost category enum
CustomFreightSchemeID	Numeric	conditional	This is required when ShippingCostCategory="Custom". Seller gets FreightSchemeID when they create a freight scheme on MyDeal marketplace
RequiresShipping	Boolean	Optional	**Defaults to true.** If true, the product needs shipping. Products with "RequiresShipping: FALSE" will not be accepted, as this is not supported at this time.

ShippingCostStandard	numeric	Conditional	Specify shipping cost in this field, when ShippingCostCategory is "Flat"or "FlatAnyQty" See description above
IsDirectImport	Boolean	Required	The direct import status of a product. FALSE indicates that the product is shipped from within Australia. TRUE indicates that the product is shipped from outside Australia.
MaxDaysForDelivery	Numeric	Required	The maximum number of calendar days estimated for delivery of the product to the customer.
DeliveryTime	String	Required	The estimated delivery time frame for delivery of the product to the customer. This will be displayed on the listing. Preferred format is "_ business days", eg. "5-10 business days".
Has48HoursDispatch	Boolean	Optional	If TRUE, then a label "24-48 hour dispatch" will be shown on the listing and you confirm that the product will be dispatched from the warehouse within 24 to 48 hours of purchase. If FALSE, then no label will show.
ProductSpecifics	Field[]	Optional	Used to send any supporting product information in key value pair that gets used in product search filters/refinements such as age group, material etc
BuyableProducts	BuyableProducts[]	Required	BuyableProducts Collection, which contains variant or standalone product details. At least one item should be available

BuyableProducts:

BuyableProducts is part of ProductGroup and represents an individual product or its variant. Product price, stock, status are always considered from this model.

Variant Options:

Variant options are required only for variant products. Variant options should NOT be defined for stand alone products.

If the product is a variant :

1. At least 1 variant option is required
2. Maximum number of variant options is 3.
3. The variant option counts and OptionName of the options should be consistent across all variants of a product.

These options will be shown as drop-down options on the product details page on MyDeal. For standalone products, information such as colour, materials etc can be sent via the ProductSpecifics.

```
[{
 "OptionName":"Color",
"OptionValue":"White",
"Position":1
},
{
 "OptionName":"Size",
"OptionValue":"S",
"Position":2
}]
```

BuyableProducts Data Overview:

Field	Data type	Required	Description
ExternalBuyableProductId	String	Conditional	External Buyable product Id stored as part of the seller channel, which is conditional. This should be passed if the seller chose to pass ExternalProductID as the unique product identifier.
SKU	string	Required	SKU of a buyable product and it should be unique for each BuyableProduct item
Price	numeric	Required	Standalone product/variant item price
Quantity	numeric	Conditional	Standalone product/variant item Quantity. If ProductUnlimited = true, Quantity will be ignored.
ListingStatus	Enum	Optional	This is ignored in all endpoints except /listingstatus endpoint. **Note: ListingStatus can be Live, NotLive. This field is used only in update product listing status endpoint to update status of product. Otherwise, it is READ ONLY field by seller in other endpoints.**
Options	Option[]	Required	For Standalone product, this must be empty For variants, Options field must have one or more options. **Note: Options limited to 3**
RRP	numeric	Optional	Standalone product/ variant item RRP price
ProductUnlimited	Boolean	Optional	Defaults to FALSE. When set to TRUE, it means that the seller has unlimited quantity of the product and the "Quantity" will be ignored.

			It is recommended to set this to FALSE and provide the correct quantity of available stock in the "Quantity" field
MetaInfo	Field[]	Optional	Meta Info field contains additional optional supporting details of the product to show in UI other than description and specifications Refer allowed named and values for metainfo

Option:

Option model represents variant options. Position field decides the order of appearance of options in the MyDeal product detail page. Position of a given option name should be consistent across all variants. If no positions are provided, default positions are calculated by MyDeal.

Field	Data type	Required	Description
OptionName	string	Required	Option Name for e.g., Size or Colour If this exceeds 10 characters, the OptionName will be displayed as "Option" on the MyDeal product details page.
OptionValue	string	Required	Option value for e.g., S, M, L, XL
Position	numeric	Optional	Unique OptionName position. for e.g., Size position as 1, Colour position as 2 etc.

Image:

Images can range from 1 to 30 max. At least one image should be available to make the product Live on MyDeal.
Src should be an external image Url and position can also be passed in the request.

Field	Data type	Required	Description
Id	numeric	Required	Image ID
Src	string	Required	External image Url
Position	numeric	Optional	Position to show in the marketplace. If not specified, default positions will be created.

Category:

Field	Data type	Required	Description
CategoryId	Numeric	Required	MyDeal predefined Category ID that can be found in the above link.

ProductGroupResponse:

This response will be sent in create/update products, update product status or price operations.

Field	Data type	Required	Description
ExternalProductId	string	Optional	Unique Product Id
ProductSKU	String	Required	Unique product SKU
BuyableProductResponses	`BuyableProductResponse[]`	Required	BuyableProductResponses Collection
Result	Enum	Required	RequestResult Enum (Success/Fail)
Errors	Error[]	Optional	Error collection added if any errors in the process

BuyableProductResponse:

Item level Errors will be present in the Errors array if applicable.

Field	Data type	Required	Description
ExternalBuyableProductId	String	Optional	Unique id
SKU	string	Required	Unique SKU
Result	Enum	Required	RequestResult Enum (Success/Fail)
Errors	Error[]	Optional	Error collection added if any errors in the process

0.12.2 Order models

Order:

Order is a group of one or more order line items purchased by a customer. Order Id is uniquely generated by the marketplace.

Field	Data type	Required	Description
OrderId	numeric	Required	Unique OrderID from marketplace

PurchaseDate	Datetime	Required	Order created date in UTC
OrderStatus	Enum	Required	OrderStatus enum
SubTotalPrice	numeric	Required	Total items price inclusive of taxes but excludes shipping
TotalPrice	numeric	Required	Total items price including Shipping, taxes
TotalShippingPrice	numeric	Required	Total shipping price includes taxes
TaxInclusive	boolean	Required	Always true
Currency	string	Optional	Default is "AUD"
CustomerEmail	String	Required	Customer email address
ShippingAddress	Address	Required	Customer shipping address
PaymentMethod	String	optional	Standard text that set up by MyDeal onboarding team during seller set up in system
PaymentReference	string	Optional	Invoice reference number if order is invoiced and paid to seller
CustomerDateOfBirth	Date	Optional	Customer Date Of Birth provided in dd/mm/yyyy (If order requires it)
OrderSource	string	Optional	The marketplace banner from which the customer purchased the product. Ex: MyDeal, BigW etc
LineItems	OrderItem[]	Required	Order line items collection

OrderItem:

Field	Data type	Required	Description
OrderItemId	numeric	Required	unique Order Line Item Id generated by marketplace
SKU	string	Required	SKU of the product
Quantity	numeric	Required	Quantity of the line item
UnitPrice	numeric	Required	One unit price of line item
UnitPriceExcCommission	numeric	Required	One unit price of line item excluding MyDeal commission
TotalPrice	numeric	Required	Unit Price * Quantity
TotalShippingPrice	numeric	Required	Total shipping price of line item
ProductId	numeric	Required	product Id in marketplace

ProductTitle	numeric	Required	
VariantOption	string	Optional	If variant, this has all options of this variant comma separated
ShippingRequired	string	Required	False - Voucher, True - Non-Voucher item to be posted
ShippingMethod	Enum	Optional	Standard text that set up by MyDeal onboarding team during seller set up in system
PrivateNotes	string	Optional	Notes to supplier if any
FulfillmentStatus	boolean	Optional	True - Shipped, false - yet to be shipped
SellerAcknowledged	Boolean	Optional	True – Seller Acknowledged False- seller not acknowledged
DispatchDate	Date	Optional	Dispatched date in UTC
DispatchCarrier	String	Optional	Shipping carrier
TrackingCode	String	Optional	Tracking code of shipment

Address:

Field	Data type	Required	Description
FirstName	string	Required	
LastName	string	Required	
Phone	string	Required	
CompanyName	string	Optional	
Address1	string	Required	
Address2	string	Optional	
Suburb	string	Required	
State	string	Required	
PostalCode	string	Required	
CountryCode	string	Required	Default is “AU”

0.12.3 Order Fulfillment models

OrderFulfillment:

OrderFulfillment is a group of one or more order line items.

Field	Data type	Required	Description
OrderId	numeric	Required	Unique OrderID from marketplace
FulfillmentItems	OrderFulfillmentItem[]	Required	Fulfillment Items.

OrderFulfillmentItem:

Field	Data type	Required	Description
OrderItemId	numeric	Required	unique Order Line Item Id generated by marketplace
SKU	string	Required	SKU of OrderItem
DispatchedDate	string	Optional	ShippedDate of OrderItem in UTC. Defaults to request sent date.
DispatchCarrier	string	Optional	DispatchCarrier of OrderItem
TrackingCode	string	Optional	TrackingCode of OrderItem

OrderFulfillmentResponse:

Field	Data type	Required	Description
OrderId	numeric	Required	Ordered ID
Result	Enum	Required	RequestResult Enum (Success/Fail)
Errors	Error[]	Optional	Item level Error collection added if any errors in the process

0.12.4 Order Cancellation models

OrderCancellation:

OrderCancellation is a group of one or more order line items.

Field	Data type	Required	Description
OrderId	numeric	Required	Unique OrderID from marketplace
Items	OrderItemCancellation []	Required	Cancellation Items

OrderItemCancellation:

Field	Data type	Required	Description
Id	numeric	Required	unique Order Line Item Id generated by marketplace

SKU	string	Required	SKU of OrderItem
Reason	string	Required	Reason for cancellation

OrderCancellationResponse:

Field	Data type	Required	Description
OrderId	numeric	Required	Ordered ID
Result	Enum	Required	RequestResult Enum (Success/Fail)
Errors	Error[]	Optional	Item level Error collection added if any errors in process

0.12.5 Order Refund models

OrderRefund:

OrderRefund is a group of one or more order refund line items.

Field	Data type	Required	Description
OrderId	numeric	Required	Unique OrderID from marketplace
Items	OrderItemRefund[]	Required	Refund items

OrderItemRefund:

Field	Data type	Required	Description
Id	numeric	Required	Unique Order Line Item Id generated by marketplace
Reason	Enum	Required	RefundReason Enum
RefundAmount	decimal	Required	Refund amount for the product
RefundShippingAmount	decimal	Optional	Refund amount for the shipping

OrderRefundResponse:

Field	Data type	Required	Description
OrderId	numeric	Required	Order ID

Result	Enum	Required	RequestResult Enum (Success/Fail)
Errors	Error[]	Optional	Item level Error collection added if any errors in process

If Order Refund is failed due to any reason, ErrorCode, ErrorID will be returned along with the response. Refer Errors for more information.

0.12.6 Other models

ActionResponse: **ActionResponse** is a standard response sent from any MyDeal API endpoint.

ResponseStatus is the status of the operation. If the operation succeeded, it gives either Complete or CompleteWithErrors. In case of failure, status will be Failed.

If the Operation is exposed as async long processing batch(for e.g., create or update products), ResponseStatus will be "AsyncResponsePending" along with PendingUri, which consists of the URL to poll by external platforms for the status of that async batch for further processing decisions.

Data is the actual response body, which will be populated in case of **Complete** or **CompleteWithErrors** status. CompleteWithErrors will contain Errors at each item level.

Field	Data type	Required	Description
ResponseStatus	String	Required	Enum values from ResponseStatus enum. **AsyncResponsePending,** **Complete,** **CompleteWithErrors,** **Failed**
Data	Generic Type	Required	Actual response body, this is generic type and varies for each API endpoint. For details please refer specific endpoint response section
Errors	Error[]	Optional	Parent level Error collection added if any errors in the process
PendingUri	String	Optional	This is available for async long running batches. In this case, ResponseStatus is "AsyncRespnsePending"

Error:

Field	Data type	Required	Description
ID	String	Required	Unique ErrorID say, InvalidToken. Refer ErrorID allowed values in ErrorCodes
Code	String	numeric	Error code

Message	String	Required	Error Message

Field:

Field	Data type	Required	Description
Name	String	Required	Name of the field
Value	String	Required	Value of the field

Allowed Names and Values for BuyableProduct level Metainfo (applicable only for variant products):

Name (exact match)	Value (must be in double quote)	Description
variationimageurl	“{your absolute image url}”	To specify the image at variant level. This field supports only one image URL. This should be the main image of the “variant” product.
shippingweight	“3”	To specify the weight at variant level.
shippingheight	“30”	To specify the height at variant level.
shippingwidth	“20”	To specify the width at variant level.
shippinglength	“40”	To specify the length at variant level.
gtin	“886691186281”	Global trade item number, include UPC, EAN (in Europe), JAN (in Japan), and ISBN at variant level. Must be a valid 8, 12, 13 or 14 digit GTIN.
mpn	“HSC0424PP”	Manufacturer Part Number at variant level

0.12.7 Enums

Enums will be communicated in string representation as part of request and response. So, please consider Enum string values.

ListingStatus:

Integer Value	String Value	Description
0	NotLive	Products which are not live and customer can’t purchase

1	Live	Products which are live and customer can purchase
2	Pending	Products which are under review by MyDeal team

OrderStatus:

Integer Value	String Value	Description
0	All	All orders except Delivery On Hold orders such as canceled/refunded
1	ReadytoFulfill	Successfully placed orders require fulfillment by seller
2	SellerAcknowledged	Orders acknowledged by seller for further fulfillment
3	Shipped	Order which has been fulfilled
4	Refunded	Order that is fully refunded to customer

FulfillmentStatus:

Integer Value	String Value	Description
0	All	All orders irrespective of shipment status
1	unShipped	Orders which are not shipped
2	Shipped	Orders which are shipped

ShippingCostCategory:

Integer Value	String Value	Description
0	FreeShipping	This is Obsolete in API V2.3 doc. Instead, please use ShippingCostCategory = 'Flat' and ShippingCostStandard= 0
1	Flat	Flat Rate. Standard rate or express rate per unit Ex: If customer orders 2 quantities of a product which has FLAT $10 shipping, the total shipping cost = $20
2	Custom	Use Custom, if a freight calculator has been set up by the seller via the MyDeal marketplace portal
3	FlatAnyQty	Similar to “Flat” category however the flat amount applies for any quantity of the product ordered Ex: If customer orders 2 quantities of a product which has FLAT_ANY_QTY $10 shipping, the total shipping cost = $10

RequestResult:

Mainly this enum is used to send item level results such as BuyableProductResponse, OrderFulfillmentResponse.

Integer Value	String Value	Description
0	Success	Operation succeeded
1	Fail	Operation failed

ResponseStatus:

Mainly this enum is used to send global level response status in the ActionResponse model.

Integer Value	String Value	Description
0	`Success`	Operation succeeded
1	`Fail`	Operation failed

RefundReason:

Integer Value	String Value	Description
1	CANCELLED_CHANGE_OF_MIND	Change of mind
6	COMPENSATION	Compensation
7	DAMAGED_ON_ARRIVAL	Damaged item on arrival
9	DISPATCH_ERROR	Incorrect product shipped
11	FAULTY	Faulty item
13	FREIGHT_DISCOUNT	Freight discount
15	LOST_IN_POST	Lost in post
16	NOT_AS_DESCRIBED	Not as described
17	OUT_OF_STOCK	Out of stock
18	OVERSEAS_ADDRESS	Outside delivery area
19	PRICE_ERROR	Price error
23	RETURN_TO_SENDER	Return to sender
32	MISSING_PARTS	Missing parts
33	DELIVERY_ADDRESS_NOT_CONFIRMED	Delivery address not confirmed

0.13 Errors

All errors, warnings, and other alerts will be communicated using Error object. This object specifies an ID and a code to match the error to an internal error identifier, and a message providing details on the error. Since all error feedback is communicated in this manner, every API response is returned with HTTP Status OK, unless there is any unrecoverable exception on our system. In case of an unhandled exception, it returns HTTP Status InternalError(500).

There are two types of Errors can be sent in response.

1. System Errors

   System errors indicate that a problem has occurred as a result of an outage, a bug, or other system issue. The exact details of system errors are not communicated to sellers; the error message will be generic and will indicate that there is a communication issue that is being addressed. All ErrorID values in the 3000, 7000, and 8000 ranges are treated as system errors


```
{
 "ID": "SystemUnavailable",
 "ErrorCode": "701",
 "Message": "The API is currently down for maintenance."
}
```


2. Custom Errors

   Custom errors indicate that a problem can be resolved through sellers' action (usually because they have submitted invalid data). This is the primary mechanism for providing feedback to the seller, so custom errors are generally communicated back to the seller seconds after they are received.

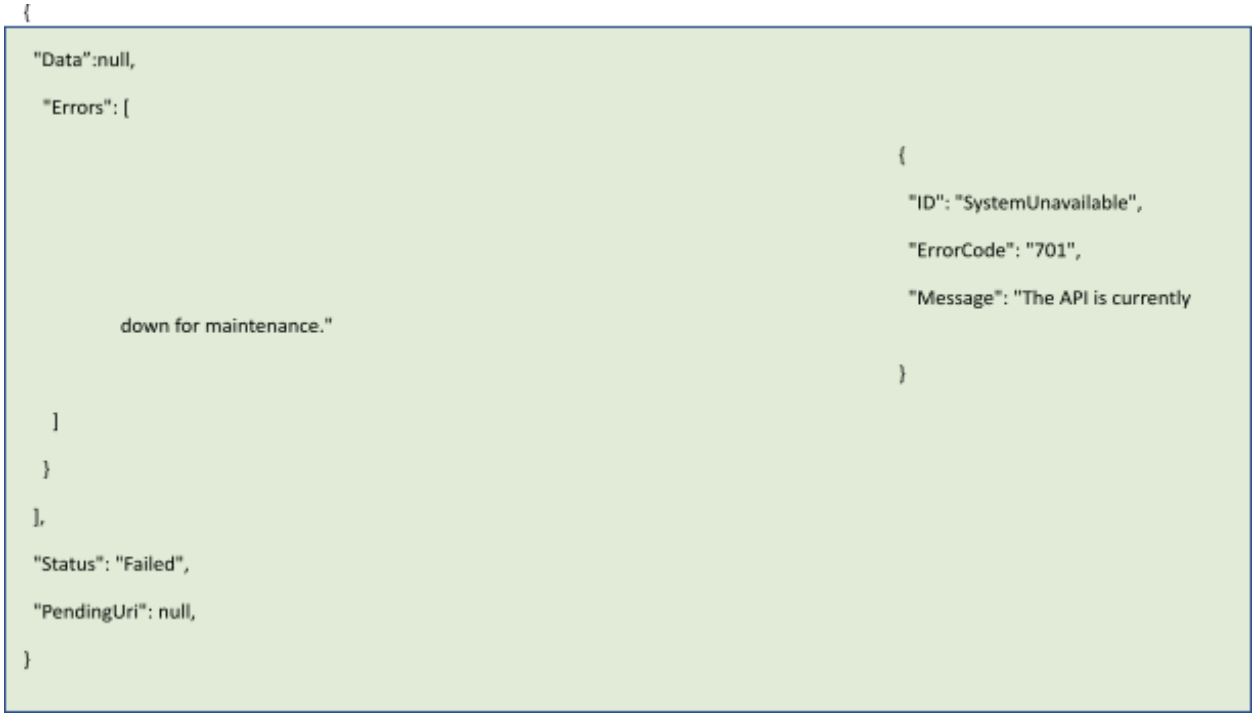

```
{
 "ID": "ProductFailedDataValidation",
 "ErrorCode": "305",
 "Message": "Data validation failed."
}
```


Error Placement:

The placement of an error within the response depends on the type of request that was issued and the type of error that occurred.

Response-level errors

When an error occurs that prevents the request as a whole from being processed by your API, the ResponseStatus of the ActionResponse will be 3 / Failed, and all errors will be returned at the root level of the response. Example response-level errors : 3001 / SystemUnavailable, 4002 / InvalidSellerID, 4001 / InvalidToken, etc.



Item-level errors

When processing a request that contains a batch of items, an error may occur whose scope is limited to a specific entity in the batch. In these cases, errors will be returned in the element's response object for which it occurred.

For example, if 10 BuyableProduct items out of a 100-product batch failed our system's item validation, we would return the errors on the 10 corresponding BuyableProductResult objects in the response, but not at the response level or on any of the other 90 products.

The only time the ActionResponse should contain a ResponseStatus of 3 / Failed is when there are no elements in the response body because the entire request as a whole could not be processed.

If elements are returned, the ResponseStatus should be 1 / Complete or 2 / CompleteWithErrors, and contain errors at the element level.

```
{
 "ResponseStatus": "CompleteWithErrors",
 "Data": [
  {
   "ExternalProductId ": "string",
   "ProductSKU": "string",
   "BuyableProducts": [
                                                                                    {
                                                                                     "ExternalBuyableProductId" :
            "string",
                                                                                     "SKU": "string",
                                                                                      "Result": "Failed"
                                                                                     }
            ],
    "Errors": [{
                                                                                   "ID":
            "ProductFailedDataValidation",
                                                                                   "ErrorCode": "305",
                                                                                   "Message": "Data validation
            failed."
                                                                                  }],
    "Result":"Failed"
   }
  ],
 "Errors": []
}
```


0.13.1 Error ID

Below are generic error IDs used to communicate errors.

Integer value	string value
1000	Info
2000	Warning

3000	SystemFailure
3001	SystemUnavailable
3002	RateLimitExceeded
4000	AuthorizationFailure
4001	InvalidToken
4002	InvalidSellerID
4003	AuthenticationFailure
5000	ProductNotFound
5001	ProductMissingRequiredFields
5002	ProductFailedDataValidation
5003	ProductFailedCreate
5004	ProductFailedUpdate
5100	ProductMissingCategory
5101	ProductInvalidCategory
5200	ProductPendingReview
6000	OrderNotFound
6001	InvalidOrder
6002	InvalidOrderStatus

6100	ShipmentFailed
6101	InvalidTrackingNumber
6102	InvalidShippingCarrier
6103	InvalidShippingClass
6200	RefundFailed
6201	UnsupportedRefundReason
6300	CancellationFailed
6301	UnsupportedCancellationReason
6400	AcknowledgementFailed
7000	InvalidRequest
7001	MissingRequiredParameter
7002	InvalidRequiredParameter
8000	BatchNotFound
8002	BatchCountExceeded

0.14 Appendix

0.14.1 Integration flow diagram

Download Integration Flow Diagram PDF

0.14.2 Endpoint Examples

View postman script