This is part of the reference for the Bloomreach Catalog Management API, which customers can use to deliver catalog data to Bloomreach and index their catalogs. Catalogs must be properly formatted before delivery, and the sections included here detail the proper format for product catalogs.

Live at https://documentation.bloomreach.com/discovery/reference/format-your-data-product

Format your Data (Product)

This guide is intended for developers, so I supplement it with links to the JSON Patch specification and a validator.

Your catalog data must be properly formatted in order for Bloomreach to ingest it.

The catalog data format is a modified JSON Patch focusing on a subset of operations and path levels. You can test your data with this JSON Patch viewer and validator.

Product Catalog - Format your Data
Open Recipe ›

Products

I introduce new terms in bold (**product**), and italicize code terms (*product_id*).

A **product** is defined as an item that is sold on your site. Each catalog entry that contains all the data for a single product is called a **record**.

To add product data to your catalog, you must first format the data in the Bloomreach product record format. A single product record is modeled as a JSON (JavaScript Object Notation) object identified by a **product_id** that contains attributes, and optional *variants* and *views* properties. You are not required to declare variants and views if your catalog does not use them.

A single catalog cannot contain two products with the same *product_id* (that is, *product_id* values should be unique within a catalog). The *product_id* should be identical to the *pid* parameter in the pixel.

Here is a sample structure of a Bloomreach formatted product record:

The sample product record shows a complete structure, and I use snippets from this for later code samples.

Example product record

```
{
  "op": "add",
  "path": "/products/435898",
  "value": {
    "attributes": {
      "title": "Three-Pocket Straight Leg Jeans",
      "price": 70,
      "description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise.",
      "brand": "Acme",
      "example_bool": true,
      "keywords": ["straight leg", "mid-rise", "back pockets", "three pocket"],
      "url": "http://www.example.com/435898-Three-Pocket-Straight-Leg-Jeans.aspx",
      "thumb_image": "http://www.images.example.com/three-pocket-straight-leg-jeans-435898-thumbnail.jpg",
      "large_image": "http://www.images.example.com/435898/435-92-jj-202.jpg|three-pocket-straight-leg-jeans-435898.jpg",
      "reviews": 3.5,
      "review_count": 3,
      "category_paths": [
        [{
          "id": 42,
          "name": "apparel"
        },
        {
          "id": 96,
          "name": "pants"
        }]
      ]
    },
    "variants": {
      "43589801": {
        "attributes": {
          "size": "xlarge",
          "availability": true
        }
      },
      "43589802": {
        "attributes": {
          "size": "large",
          "availability": true
        }
      }
    },
    "views": {
      "viewID_1": {
        "attributes": {
          "price": 41
        }
```

op and path

Every product record requires the **op** and **path** properties.

- **op** defines the type of operation to be performed on the catalog with this record. Values allowed are "add" (which can also replace) and "remove".
- **path** identifies a specific product record, or a specific portion of a product record, to be operated on. To operate on an entire product record, the path value should be "/products/{product_id}". The product_id for a given product should be the same in both your catalog data and your pixel. To operate on a specific portion of a product record, refer to the Patch Operation Use Cases for the correct path value.

I use a callout to highlight "op" and "path" since their values are crucial to performing certain operations.

Patch operations

The PATCH operations possible on your product record are given below:

- Add/replace a product
- Remove a product
- Add/replace product attributes
- Remove product attributes
- Add/replace a product attribute
- Remove a product attribute

Attributes

Each product is defined by its **attributes**, which can include characteristics (such as color or size) and inventory data (such as price or description).

The attributes property specifies the product's **base product attributes**. If you have variants and/or views, base product attributes remain identical across all product variants or product views, unless they are overridden in a view.

Example attributes property

```
"attributes": {
  "title": "Three-Pocket Straight Leg Jeans",
  "price": 70.00,
  "description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise.",
  "brand": "Acme",
  "example_bool": true,
  "keywords": ["straight leg", "mid-rise", "back pockets", "three pocket"],
}
```

Reserved Attributes

There are reserved attributes that Bloomreach understands and treats specially. Reserved attributes have a preset name and need to be supplied in the format expected by Bloomreach.

Categories

To place a product in a category, add the reserved attribute: **category_paths**. If added, *category_paths* requires a name and id for each category to make sure Bloomreach can understand the data you're submitting.

> *category_paths* must be sent as an array of arrays, even if it only contains a single path.

In the example below, the product is assigned to categories apparel > petites > pants > denim and apparel > women > jeans. Categories should be declared in order from top to bottom, and the id should be unique and consistent with the corresponding category name.

Example category_paths property

```
"attributes": {
  "category_paths": [
    [{
      "name": "apparel",
      //"apparel" is the top-level category
      "id": "1"
    },
    {
      "name": "petites",
      //"petites" is a subcategory under "apparel"
      "id": "4"
    },
    {
      "name": "pants",
      //"pants" is a subcategory under "petites"
      "id": "15"
    },
    {
      "name": "denim",
      //"denim" is a subcategory under "pants"
      "id": "16"
    }],
    [{
      "name": "apparel",
      //"apparel" is the top-level category
      "id": "1"
    },
    {
      "name": "women",
      //"women" is a subcategory under "apparel"
      "id": "2"
    },
    {
      "name": "jeans",
      //"jeans" is a subcategory under "women"
      "id": "35"
    }]
  ]
}
```

I provided annotations to help clarify the category structure represented in the code.

Dynamic Attributes

If an attribute is not one of the reserved attributes, it is considered to be a **dynamic attribute**. Dynamic attributes may be given any name that is not reserved and any value of a supported value type.

- Dynamic attribute names - must consist of alphanumeric (A-Z, 0-9), space, or underscore (_) characters only, and should not start with a digit or space.
- Dynamic attribute values - supported types include string, integer, float, boolean, or a homogeneous array of those types. Objects are not currently supported, so if you have objects of arbitrary depths, you would need to flatten them out.

Sending irrelevant attributes or information will increase data ingestion and indexing time, and may impact request latency.

I had to specify restrictions for dynamic attributes as they could cause problems with catalog delivery or ingestion.

Max length of any single-value attribute value must be less than 1KB. For a multi-valued attribute (array), the max length of a single element's value must be less than 1KB.

Variants

Variants are a sellable variation of a base product.

The *variants* property contains all of the product's variants (typically SKUs). Each variant is identified by a **variant_id**, which must be unique to the particular product.

Each variant has a nested *attributes* property that is similar to the base product *attributes* property. Every attribute that varies must be specified within the *attributes* property of each variant. Attributes that are identical across all variants should be specified as base product attributes instead. Refer the Attributes section for more information about attribute values.

An example *variants* property is given below:

Example variants property

```
"attributes": {
  "price": 70,
  "description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise.",
  "brand": "Acme"
},
//Base product attributes are inherited by all variants, so all variants share the “price”, “description”, and "brand" attributes here
"variants": {
  "43589801": {
  //Variant ID
    "attributes": {
      "size": "xlarge",
      //“size” is an attribute that varies
    }
  },
  "43589802": {
    "attributes": {
      "size": "large",
    }
  }
}
```

Patch operations

The PATCH operations available on Variants are given below:

- Add/replace variants of a product
- Remove variants of a product
- Add/replace a variant to a product
- Remove a variant to a product
- Add/replace variant attributes
- Remove variant attributes
- Add/replace a variant attribute
- Remove a variant attribute

Views

Views are used to associate different versions of a product or variant to certain viewers, typically for different stores or markets. For example, suppose you had two views, West and East. You could have a product that costs one price in the West view and a different price in the East view, or you could have a product that is only available in the West view.

Product views indicate that the product exists in a specific catalog view and contain the product's view-specific attributes. The views property contains all of a product's views. Each view is identified by a **view_id**.

If you are using Bloomreach Views functionality, the views property is required for every product and ALL of the views a product exists in must be supplied for every product.

> **View overrides**
>
> Attributes declared within a product view are combined with the base product attributes. If the same attribute is declared for the base product and within the product view, then the product view attribute overrides the base product attribute.
>
> For example, if the base product attributes include "price": 10.00 and the product view attributes include "price": 15.00, then the view will use "price": 15.00.

An example *views* property is given below:

Example views property

```
"views": {
  "viewID_1": {
  //View ID
    "attributes": {
    //View-specific attributes
      "price": 41.00 //Price in viewID_1
    }
  },
  "viewID_2": {
    "attributes": {
      "price": 60.00 //Price in viewID_2
    }
  }
}
```

Patch Operations

Not every customer uses variants and/or views. These sections were placed on the same page as the previous sections, but it may have been better to place these on their own subpages.

The patch operations available depend on the type of property they operate on, and the Product Manager requested "Patch operations" subsections for each property type.

The available patch operations and code samples for each are on the linked page.

The PATCH operations available for views of a product are given below:

- Add/replace views for a product
- Remove views for a product
- Add/replace a view for a product
- Remove a view for a product
- Add/replace view attributes for a product
- Remove view attributes for a product
- Add/replace a view attribute for a product
- Remove a view attribute for a product

Variants and Views

If you have both variants and views, any view-specific attributes or overrides should be specified within a *variants* property nested inside that *views* property. Variant attributes that are identical across all views should be specified in the top-level *variants* property instead.

> **Variant view overrides**
>
> Variant attributes declared within a product view are combined with the attributes specified in the top-level *variants* property. If the same attribute is declared for the base product and within the product view, then the variant view attribute overrides the attribute specified in the top-level *variants* property.
>
> For example, if the top-level *variants* property includes "price": 10.00 and the variant view includes "price": 15.00, then the variant will use "price": 15.00 in that view.

An example *views* property with variants is given below:

Example views property with variants

```
"variants": {
  "43589801": {
    "attributes": {
      //Variant attributes for all views it exists in
      "price": 10.00 //Price for all views without a variant view override
    }
  }
},
"views": {
  "viewID_1": {
  //View ID
    "attributes": {
      //Attributes shared by all variants in the view
      ...
    },
    "variants": {
    //Nested variants property within the view
      "43589801": {
        "attributes": {
          //Place view-specific variant attributes or overrides here
          "price": 15.00 //Variant view override, price of this variant will be 15.00 in this view
          ...
        }
      }
    }
  },
  "viewID_2": {}
  //View with no view-specific variant overrides must still be included if the product exists in that view
}
```

Variant views are a complex use case, so I tried to annotate the code sample with all the details.

Exclude a variant from a view

Variants are included by default in all views. To exclude a variant from a view, add this attribute within that variant's property: **"exclude_from_view": true**.

You can include the variant again by setting the value to false, or by removing the *exclude_from_view* attribute.

Example use of exclude_from_view

```
"views": {
  "viewID_1": {
    "attributes": {
      ...
    },
    "variants": {
      "43589801": {
        "exclude_from_view": true
        //exclude_from_view is nested within the variant ID, but outside "attributes"
        "attributes": {
          ...
        }
      }
    }
  },
  ...
}
```

Add/replace a complete product with views, variants, and variant views

The sample product record below has 3 variants and 3 views.

- In the "ae" view, one variant has been excluded
- In the "us" view, one variant has overrides, and the other two have no view-specific variant attributes
- In the "it" view, no overrides are provided

Product record with views, variants, and variant views


```
{
  "op": "add",
  "path": "/products/435898",
  "value": {
    "attributes": {
      "title": "Three-Pocket Straight Leg Jeans",
      "department": "apparel"
      "brand": "Acme",
      "keywords": ["straight leg", "mid-rise", "back pockets", "three pocket"],
    },
    "variants": {
      "43589801": {
        "attributes": {
          "size": "onesize",
        }
      },
      "43589802": {
        "attributes": {
          "description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise.",
        }
      },
      "43589803": {
        "attributes": {
          "image": "localhost/image.png",
        }
      }
    },
    "views": {
      "ae": {
        "attributes": {
          "currency": "AED",
        },
        "variants": {
          "43589801": {
            "attributes": {
              "availability": true,
              "price": 106.00,
              "color": "blue",
            }
          },
          "43589802": {
            "attributes": {
              "availability": true,
              "price": 53.00,
              "color": "red",
            }
```


The above product would have these attributes:

Attribute	Base product	"ae" view	"us" view	"it" view
product_id	435898	435898	435898	435898
title	Three-Pocket Straight Leg Jeans	Three-Pocket Straight Leg Jeans	Three-Pocket Straight Leg Jeans	Three-Pocket Straight Leg Jeans
department	apparel	apparel	apparel	apparel
brand	Acme	Acme	Acme	Acme
keywords	straight leg, mid-rise, back pockets, three pocket	straight leg, mid-rise, back pockets, three pocket	straight leg, mid-rise, back pockets, three pocket	straight leg, mid-rise, back pockets, three pocket
currency	N/A	AED	N/A	N/A
variant 43589801	"size": "onesize"	"size": "onesize" "availability": true "price": 106.00 "color": "blue"	"size": "one" "availability": true "price": 41.00	"size": "onesize"
variant 43589802	"description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise."	"description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise." "availability": true "price": 53.00 "color": "red"	"description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise."	"description": "For the most flattering fit, these gorgeous straight leg jeans have a great mid-rise."
variant 43589803	"image": "localhost/image.png"	excluded from this view	"image": "localhost/image.png"	"image": "localhost/image.png"

The table here represents the attribute values for the product, each of its variants, and in each view.

Patch operations

The PATCH operations for Variant View data are given below:

- Add/replace variants view data
- Remove variants view data
- Add/replace variant view data
- Remove variant view data
- Add/replace variant view attributes
- Remove variant view attributes
- Add/replace a variant view attribute
- Remove a variant view attribute