SALSIFY
CUSTOMER SUCCESS

JSON Import Format

Last Modified on 03/18/2017 8:01 pm EDT

Overview

The high level structure of the JSON document is:

Version 1: 2013-04

```
[
  { "header": { } },
  { "attributes": [] },
  { "attribute_values": [] },
  { "products": [] }
]
```

Version 2: 2

```
[
  { "header": { } },
  { "attributes": [] },
  { "attribute_values": [] },
  { "digital_assets": [] },
  { "products": [] }
]
```

The elements must appear in the JSON array in the specified order. The only mandatory element is the header that describes how the import document should be processed. Each of the elements in the JSON array will be described in detail throughout this document but first here are some core concepts:

- The primary types in the system are attributes, attribute values, products, relations, and digital assets.
- Digital assets and relations are nested within products in the version 1 format ("2013-04"); only relations are nested within products in the version 2 format.
- All of these entities can have attribute values assigned to them. Salsify has special handling of certain attributes and attribute values (e.g. a URL for a digital asset).
- Certain user defined attributes can have special roles in Salsify - in particular attributes for product id, product name, and relation type (e.g. a product's sku can be used to identify the product).

All the Salsify attribute names are lower snake case (no spaces), but custom attributes names do not have to follow this naming convention. All Salsify attributes that can appear in the same context as user defined attributes (e.g. the attribute values of an attribute) are prefixed with 'salsify:'.

Products

Products are just JSON objects whose members are the attributes of the product:

```
{
  "id": "3635065",
  "SKU": "3635065",
  "Model": "KDL-40NX700",
  "name": "Sony 40 in. Bravia TV",
  "Description": "This is an amazing tv. You should buy it.",
  "Brand": "Sony",
  "Aspect Ratio": "16:9",
  "Feature": ["Child Lock", "Picture in Picture", "Sleep Timer"],
  "Release Date": "10-19-2010"
}
```

Note that multi-valued product attributes are JSON arrays. Salsify has special handling of the following attribute roles:

- product_id (Required) - This uniquely identifies the product within the set of your products. This will be shown when looking at lists of products in Salsify. By default Salsify assumes this maps to an attribute called 'id'.
- product_name (Required) - A brief name for the product. This will be shown when looking at lists of products in Salsify. By default Salsify assumes this maps to an attribute called 'name'.

These default attribute role mapping can be overridden in the attributes section of the import document. The following example configures SKU to be used as the product id and *Display Name* to be used as the product name:

```
[
  {
    "attributes": [
      {
        "salsify:id": "SKU",
        "salsify:role": "product_id"
      },
      {
        "salsify:id": "Display Name",
        "salsify:role": "product_name"
      }
    ]
  },
  {
    "products": [
      {
        "SKU": "3635065",
        "Display Name": "Sony 40 in. Bravia TV"
      }
    ]
  }
]
```

Salsify will automatically create attributes based on the attribute values that appear in your products. However you can explicitly specify these attributes if you want to attach additional information to them. See the section on attributes for more information.

Categories and Enumerated Attribute Values

Product attributes can reference reusable attribute values. This is useful for defining hierarchies of attribute values and attaching additional metadata to attribute values. The following example shows how to define a *Product Category* hierarchy and associate a product with that hierarchy:

Attribute values support the following attributes:

- **salsify:attribute_id** (Required) - This identifies the attribute that this attribute value belongs to.
- **salsify:id** (Required) - This uniquely identifies the attribute value within the set of the owning attribute's values. This will be shown when looking at details of attribute values in Salsify.
- **salsify:parent_id** - For a hierarchical attributes, this will be the id of the attribute value's parent. This attribute should not be specified for attribute values that appear at the top of a hierarchy. A single root is not required at the top of the hierarchy.
- **salsify:name** - A brief name for the attribute value. This will be shown when looking at attribute values in Salsify. If not specified, this will default to an attribute value's id.

Attributes

Attributes will automatically be defined by Salsify when attribute values for that attribute are encountered in the import document. Attributes can also be explicitly defined to attach additional information to them. The follow example illustrates how to do this:

```
{
  "attributes": [
    {
      "salsify:id": "SKU",
      "salsify:role": "product_id"
    },
    {
      "salsify:id": "Release Date"
    },
    {
      "salsify:id": "Resolution",
      "salsify:name": "Screen Resolution",
      "Attribute Category": "TECH_SPEC"
    }
   ]
}
```

Notice that customers can associate arbitrary metdata with attributes via custom attribute values. Salsify has special handling of the following attribute attribute values:

- **salsify:id** (Required) - This identifies the attribute within the set of all attributes.
- **salsify:name** - A brief name for the attribute. This will be shown when looking at attributes in Salsify. If not specified, this will default to an attribute's id with underscores replaced with spaces and the initial letter of each word capitalized.
- **salsify:role** - The role this attribute plays in various contexts. See the section on roles for more information.
- **salsify:attribute_group** - The attribute group this attribute belongs to
- **salsify:position** - The sort order of the property. This must be a value between 0 and 2**31 (inclusive)
- **salsify:data_type** - Defaults to string. See the section on data types for more information.

Roles

Attribute roles provide a mapping for Salsify to use terminology they are familiar with e.g. products have SKUs not IDs. The following roles are supported:

- product_id
- product_name
- relation_type

See the section on products for more details on the product_id and product_name roles. See the section on relations for more detail on the relation_type role.

Data Types

The following data types are currently supported:

- **string** (default) - plain text, accepts alpha numeric and special characters, case sensitive.
- **link** - a url
- **html**
- **rich_text** - accepts alpha-numeric and special characters with formatting, case sensitive.
- **enumerated** - an enumerated set of attribute values. Plain text, alpha numeric characters, hierarchical, fixed set, case sensitive.
- **number** - Plain, numeric characters only
- **date** - yyyy-mm-dd format
- **boolean** - Accepts y, yes, n, no, true, false, not case sensitive. Exports to JSON format as true or false.
- **digital_asset**

The product_id and product_name properties can only have a data type of string. The relation_type property must have a data type of enumerated.

Relations

Products can have relations to other products associated with them (e.g. accessories, up-sells, cross-sells, etc.):

```
[
  {
    "attributes": [
      {
        "salsify:id": "relation_type",
        "salsify:name": "relation_type",
        "salsify:data_type": "enumerated",
        "salsify:role": "relation_type"
      }
    ]
  },
  {
    "attribute_values": [
      {
        "salsify:attribute_id": "relation_type",
        "salsify:id": "CABLES",
        "salsify:name": "Cables"
      },
      {
        "salsify:attribute_id": "relation_type",
        "salsify:id": "MOUNTS",
        "salsify:name": "Mounts"
      }
    ]
  },
  {
    "products": [
      {
        "SKU": "3635065",
        "name": "Sony 40 in. Bravia TV",
        "salsify:relations": [
          {
            "salsify:target_product_id": "2317408",
            "relation_type": "CABLES"
          },
          {
            "salsify:target_product_id": "1048546",
            "relation_type": "MOUNTS"
          }
        ]
      },
      {
        "SKU": "2317408",
        "name": "Monster 1000HDX 12ft HDMI Cable"
      },
      {
        "SKU": "1048546",
        "name": "Sanus Tilting Wall Mount for Flat-Panel TVs"
      }
    ]
  }
]
```

Relations require the following attribute values:

- salsify:target_product_id - The ID of the related product.
- A value for the relation_type attribute.

Digital Assets

Products can have digital assets associated with them via a *Digital Assets* attribute in version 1 of the JSON format:

```
{
  "SKU": "3635065",
  "name": "Sony 40in. Bravia TV",
  "salsify:digital_assets": [
    {
      "salsify:id": "3635065-FRONT-VIEW",
      "salsify:name": "Sony 40in Bravia TV Front View",
      "salsify:url": "http://example.com/3635065-FRONT-VIEW.png",
      "salsify:is_primary_image": true
    },
    {
      "salsify:id": "3635065-REAR-VIEW",
      "salsify:name": "Sony 40in Bravia TV Rear View",
      "salsify:url": "http://example.com/3635065-FRONT-VIEW.png"
    }
  ]
}
```

In version 2 of the JSON format, digital assets have their own top-level key, and can be associated with products by assigning the digital asset ID to a property value with a data type of digital asset:

```
  "products": [
    {
      "SKU": "3635065",
      "name": "Sony 40in. Bravia TV",
      "front_view_image": "3635065-FRONT-VIEW",
      "rear_view_image": "3635065-REAR-VIEW"
    }
  ]
},
{
  "digital_assets": [
    {
      "salsify:id": "3635065-FRONT-VIEW",
      "salsify:name": "Sony 40in Bravia TV Front View",
      "salsify:url": "http://example.com/3635065-FRONT-VIEW.png"
    },
    {
      "salsify:id": "3635065-REAR-VIEW",
      "salsify:name": "Sony 40in Bravia TV Rear View",
      "salsify:url": "http://example.com/3635065-REAR-VIEW.png"
    }
  ]
}
```

Digital assets can have arbitrary attribute values associated with them. Salsify has special handling of the following attributes:

- salsify:id - This uniquely identifies the digital asset within the set of all digital assets for all products. This is only required if you will be externally referencing this digital asset within Salsify (e.g. via a data update operation in the yet to be defined data update api).

- salsify:name - A brief name for the digital asset. If not specified, the name will be generated from the ID. If the ID also isn't specified, the name will be generated from the URL.
- salsify:url (Required) - A url that they digital asset can be fetched from.
- salsify:is_primary_image - Indicates that a thumbnail of this digital asset should be shown when displaying product images in Salsify. This should only be true for on digital asset for a given product.

Headers

The header section of the import document contains information on how the import document should be processed. The header contains the following fields:

- **version (required)** - This must be "2013-04" for version 1 or "2" for version 2.
- **scope (required)** - Scope allows you to restrict the scope of the types affected by the import. The valid values for this are `products`, `attribute_values`, `attributes`, `digital_assets`, `relations`, and `all`.
    - `Scope` also optionally allows imports to access entity values by specifying the entity with specific property or attribute ids, `dynamic` (the default), or `all`. (Example below)
- **mode (optional)** - Can be `upsert` or `truncate`. Upsert will add or update entities. Truncate removes any entity within the specified import scope that does not appear in the import file. The default is upsert.

The following example imports all types into the system:

Version 1: 2013-04

```
{
  "header": {
    "version": "2013-04",
    "scope": ["all"]
  }
}
```

Version 2: 2

```
{
  "header": {
    "version": "2",
    "scope": ["all"]
  }
}
```

The following example only imports products, attribute values, and attributes:

```
{
  "header": {
    "version": "2013-04",
    "scope": ["products", "attribute_values", "attributes"]
  }
}
```

Imports can be limited to operate only on certain product attributes by enumerating the attributes within the product scope:

```
{
  "header": {
    "version": "2013-04",
    "scope": [
      "attribute_values",
      "attributes",
      "digital_assets",
      { "products": ["sku", "name", "brand", "color"] }
    ]
  }
}
```

This import will only change the sku, name, brand, and color attributes on products. All other attributes will be unaffected by the import. The list of product attributes must include the attribute with the product id role.

Example of import configured to truncate any entity which is not specified in the import body.

```
{
  "header": {
    "version": "2013-04",
    "scope": ["all"],
    "mode": "truncate"
  }
}
```

Dynamic Import Scope

If an attribute is included in the scope but a value is not provided on a product, it will remove that value from the product if it exists. For example, the following import is scoped to the sku, brand, and description attributes, but each product only contains values for two of the three:

```
[
  {
    "header": {
      "version": "2013-04",
      "scope": [
        { "products": ["sku", "brand", "description"] }
      ]
    }
  },
  {
    "products": [
      {
        "sku": "2317408",
        "brand": "Sony"
      },
      {
        "sku": "1048546",
        "description": "Sanus Tilting Wall Mount for Flat-Panel TVs."
      }
    ]
  }
]
```

Because product 2317408 did not specify a description and product 1048546 did not specify a brand, those values will be removed from the respective products if they exist.

If you'd prefer to only update attributes where values are given, you can use the `dynamic` product import scope:

```
{
  "header": {
    "version": "2013-04",
    "scope": [
      { "products": "dynamic" }
    ]
  }
}
```

With the dynamic product import scope, the above import file will update the brand for 2317408 and the description for 1048546, but it will not remove the missing attributes.

The dynamic mode is only available for the products scope.

Multi-File imports

Attributes, attribute values, and products can optionally be defined in separate documents. In this case the main import document should contain the relative path to the auxiliary document e.g.

```
[
  { "header": { } },
  { "attributes": "attributes.json" },
  { "attribute_values": "attribute_values.json" },
  { "products": "products.json" }
]
```