How to integrate your offer (products, ads,..) in the most convenient way into the Verbolia solution?

Product feed · https://www.verbolia.com/knowledge/data-feed-integration/

(An) up-to-date product/attribute/review feed(s) must be available in order to generate the most relevant, super-optimized landing pages.

Verbolia accepts **CSV, XML or JSON** formats via **FTP or URL** connection. Best is to make the data feed urls easily accessible, meaning **no authentication** required.

The product feed with all **general product information** and attributes feed with **extra product attribute details** can be integrated in one and the same feed, if structured well (see examples below). A reviews feed should be set-up separately, because of potential file size issues.

If you want/need to use different feeds (for product data, extra attributes, reviews), make sure there's a **unique identifier** (product id, ad id, etc.) available in every file to map the different feeds correctly with each other.

The data feed(s) must contain **all information** needed to rebuild the **design of the client's search result pages**. Meaning all product information, product attributes details, images, badges, dates,... the client shows on his search results pages (in titles, descriptions, buttons, filters,...) needs to be structurally available in the feed(s) to make sure the Verbolia landing pages have exactly the same look-and-feel and behaviour as the client's website.

In general: the more data the better, as it will also be helpful to reach more qualitative results in keyword research (if applicable) and for matching the most relevant products to the pages (via our internal search algorithms).

Some basic fields, using Verbolia's **default namings** are: Id, GroupId, Category, Brand, Title, Url, ImageUrl, SalePrice, OriginalPrice, QuantityInStock.

Additional fields are compulsory but could turn out to be mandatory when crosschecking the needs with the client's search result pages (mostly, using Verbolia's default namings: AdditionalImageUrl, Color, Size, Season, Badge, Rating,...).

The GroupId is needed when you want to only show 1 single (grouped) product result (image/link) on the Verbolia pages iso. all the different colors, sizes, etc.

Below you can read more detailed explanations and download several examples of clean data feeds, of different types and formats, which will give you more clarity on the needed information.

CSV

- Field separators can be comma , / TAB / semi-colon ; / pipe |
  Default separator** is **comma ,
- Attribute value delimiters can be comma , / semi-colon ; / pipe |
  Default attribute value delimiter** is **pipe |
- For AttributeList columns, delimiters can be colon : / equals = / comma , / semi-colon ; / pipe |
  Default attributetype:value delimiter** is **colon :
  Default attribute types delimiter** is **semi-colon ;

If you have multiple values for the same attribute type, you should duplicate the attributetype:value for all available attribute values.

Default example:

```
attributetype1:value;attributetype2:value;attributetype3:value
color:blue;size:small;size:medium;size:large;season:summer;season:spring
```

The chosen field separator must be different from the chosen delimiters however!!!

Double quotes “” can be used to enclose values which could create conflicts with the above defaults (eg. SalePrice value is comma-delimited too, full text values with commas,...).

CSV Example 1:

- 1 line equals 1 product on level of color/price.

- GroupId available for grouping (if needed). When activated in the Verbolia app, the results (image/link) on the Verbolia pages will be shown on this level.
- Different attribute columns (brand, category, sizes,...), values are pipe delimited.
- 1 column containing AttributeList per product (attribute types Color, ImageUrl, SalePrice,...), following the above default delimiters.

CSV Example 2:

- Same as 1a, only more attribute types are integrated into the AttributeList column (eg. size, season, badge,...).

CSV Example 3:

- Same as 1a, only 1 line equals 1 product on level of color/price and size.

XML

XML Example

- It's crucial to use **camelCase** for all element names.
- The element name for each product must be **<item>** inside of the root element **<channel>**

Example:
`<?xml version="1.0" encoding="UTF-8"?> <channel> <item> --> All data for a product will be inside of this element </item> <item> --> All data for a product will be inside of this element </item> ... </channel>`

- All attributes must be in an element **<attribute>** inside of the element **<attributes>**. The element **<attributes>** must be in the root of an element **<item>**.
- An element **<attribute>** can only contain **1 simple element** (like <value> or <type>). So **no list of attributes** is accepted! This means, if you have an attribute with two different values, you must create 2 elements <attribute> with the same type (so you can have duplicate types).

Example:

```
... <item> <attributes> <attribute> <type>color</type> <value>White</value> </attribute> <attribute> <type>colorFamily</type> <value>White</value> </attribute> <attribute> <type>colorFamily</type> <value>Green</value> </attribute> <attribute> <type>thumbnail</type> <value>/images/heh-9133_600x600.png</value> </attribute> </attributes> </item>
```

Json

Json Example

- It’s crucial to use **camelCase** for all element names.
- Format is an **array of objects**, where each object contains a product/ad.
  `[ { --> All data for a product will be inside of this element }, { --> All data for a product will be inside of this element }, … ]`
- Format is again an **array of objects** but each object can only contain 2 name-value pairs: type and value. This means, if you have an attribute with two different values, you must create 2 objects with the same type (so you can have duplicate types).
  `"attributes": [{ "type": "color", "value": "White" },{ "type": "colorFamily", "value": "Green" },{ "type": "colorFamily", "value": "White" },{ "type": "thumbnail", "value": "/images/heh-9133_600x600.png" },{ "type": "price", "value": "24.99" },{ ... } ]`