



Adobe® Marketing Cloud

Understanding and Designing Index Connector Feeds in Adobe Search&Promote

Contents

- Legal notice.....3**

- About index connector feeds.....4**
 - Anatomy of an index connector feed.....4

- Designing an index connector feed using fields with multiple values.....8**
 - Example of hierarchical category facets.....11
 - Example of hierarchical category facet where products are in two categories.....11
 - Example of facets with more than one value.....12
 - Example of facets with ranges.....12
 - Example of an XML and XSD index connector feed.....13
 - Transforming data at the time of indexing.....13

Legal notice

© 2013, Adobe Systems Incorporated. All rights reserved [Terms of Use](#) | [Privacy Center](#)

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Documentation version: 1.0

About index connector feeds

This paper describes the standard format of data feeds that Index Connector uses in Adobe Search&Promote. It also describes various ways to design your index connector feed.

The format of an index connector is similar to the following code example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<records version="1.0">
  <record>
    <id>...</id>
    <prod-name>...</prod-name>
    <price>...</price>
    ...
    <list>
      <item>
        <color>...</color>
        ...
      </item>
      ...
    </list>
  </record>
  ...
</records>
```

The root tag is always `<records>` and each product is nested within a `<record>` tag. Fields within each `<record>` tag are identified by custom tags that vary from customer to customer. Each record must include a unique identifier field that remains the same over time. A product SKU is often an appropriate choice for a unique identifier. Which tag contains the unique identifier is a configuration setting. Whichever tag is identified as the unique identifier is required in each record. If identifiers are not unique, search results and future index updates are unpredictable.

Let's now look at the anatomy of an example feed and breakdown its structure.

Anatomy of an index connector feed

The following is an example of an index connector feed. Each `<record>` node in the XML includes the following types of fields:

- id
- user-defined
- list
- dynamic facet

We will examine the feed more closely by studying the different field types that are used, including encoding concerns and the use of whitespace in the code.

Index connector feed example

Index Connector Feed parts	Description
<code><xml></code> tag	Required. The index connector feed starts with a standard <code><xml></code> tag.

Index Connector Feed parts	Description
	The encoding attribute is also required within the tag and the value is set to UTF-8 only. Any other encoding attribute value is rejected.
<records> tag	<p>Required. If this tag is missing, the entire feed is rejected.</p> <p>This is the root element, or outer-most tag in the document.</p> <p>While the <code>version</code> attribute lets Adobe Search&Promote detect different versions of the feed in the future, it is currently optional. If the <code>version</code> attribute is not present in the feed, it defaults to 1.0. If it is present, then its value must be 1.0.</p>
<record> tag	<p>Each <record> tag in the feed wraps around just one product.</p> <p>You can have zero or more <record> tags. In this index connector feed example, there is just one record.</p>
<list> tag	<p>This tag is the parent element for list attributes that are related to the record.</p> <p>You can have zero or more <list> tags within a single record.</p> <p>You cannot nest lists within lists.</p>
<item> tag	Each <item> tag in the feed wraps around a single item in the list. You can have zero or more <item> tags within a <list> tag.
Fields	<p>Records or products consist of one or more feed-specific field tags. There are no "predefined" field tags. You can also use whatever tag names are appropriate in each feed you create.</p> <p>The index connector identifies one field as the primary key. The primary key field must be present in every record. All other fields are optional. However, if fields are missing then your index is most likely incomplete.</p> <p>See Field types.</p>
List-valued fields	<p>You can use one of the following two schemas for denoting lists in your index connector feed:</p> <ul style="list-style-type: none"> Specify list-values fields by using a separator character between values. <p>The separator character must not appear within any value. The separator you use must match the separator character that is specified on the corresponding Adobe Search&Promote meta field definition.</p> <p>See http://microsite.omniture.com/t2/help/en_US/snp/index.html#Meta_tag_field_options.html</p> <pre data-bbox="467 1692 1029 1892"> <?xml version="1.0" encoding="UTF-8" ?> <records version="1.0"> <record> <sku-id>111 222</sku-id> <color>red blue</color> </record> </records> </pre>

Index Connector Feed parts	Description
	<p>The index connector feed example earlier uses this method.</p> <ul style="list-style-type: none"> Specify list-values fields by using "list" nodes to denote the individual items as in the following example: <pre data-bbox="467 407 1029 810"> <?xml version="1.0" encoding="UTF-8" ?> <records version="1.0"> <record> <list> <item> <sku-id>111</sku-id> <color>red</color> </item> <item> <sku-id>222</sku-id> <color>blue</color> </item> </list> </record> </records> </pre> <p> Note: Regardless of which schema you use, a limitation of the list mechanism is that a field can become long. You have a maximum of 1024 values per field for "list-type" fields.</p> <p>See Designing an index connector feed using fields with multiple values.</p>
<dynamic-facets> tag	<p>Dynamic facets are arbitrary name-value attributes that you can use as refinement facets. Currently, Adobe Search&Promote uses "facet slots" to handle dynamic facets. You can either do the facet slotting yourself, or you can have Adobe Consulting Services help you.</p> <p>If you choose to do the facet slotting yourself, then the facets come across like regular fields as in the following example:</p> <pre data-bbox="467 1268 886 1367"> <fn0>Metal</fn0> <fv0>Silver Gold</fv0> <fn1>Watch Face Type</fn1> <fv1>analog</fv1> </pre> <p>If you choose to have Adobe Consulting Services help you create the facet slots, then the name-value pairs must be within the <dynamic-facets> </dynamic-facets> nodes so that Adobe Search&Promote's translation step can identify which attributes to slot dynamically.</p> <p>Use dynamic-facets under the following scenarios:</p> <ul style="list-style-type: none"> You have attributes that change frequently at index time that you want to use as refinement facets. You have more than 70 category-specific attributes that you want to use as refinement facets.
Encoding	<p>You can write field content with or without a CDATA section. However, be aware that if you use fields without a CDATA section, be sure that they are properly encoded in XML.</p> <p>See http://en.wikipedia.org/wiki/CDATA#CDATA_sections_in_XML</p>

Index Connector Feed parts	Description
Whitespace	<p>Whitespace within fields is always normalized. That is, all runs of whitespace characters are treated as a single space. This rule is true whether the text is wrapped in a CDATA section or not. During the XML parsing step, sometimes not all whitespace is collapsed. However, during the indexing process, all whitespace is eventually collapsed.</p> <p>A consequence of this functionality is that you can wrap or indent text content any way you like—the result is the same.</p>

Designing an index connector feed using fields with multiple values

Adobe Search&Promote supports list fields which are simply fields with multiple values. At search time, use a field table if you want to query on records that have specific values in two or more lists. Be sure the lists line up correctly as explained in the following three examples.

In this first example, suppose that you have a product (abc) that comes in three colors and each color has its own SKU as in the following:

Product	Product SKU id	Color
abc	abcR	Red
abc	abcB	Blue
abc	abcG	Green

The resulting index connector feed uses fields with multiple values and looks like the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<records version="1.0">
  <record>
    <id>123456</id>
    <product>abc</product>
    <sku-id>abcR|abcB|abcG</sku-id>
    <color>red|blue|green</color>
  </record>
</records>
```

The SKU for each color is repeated. At search time, using a field table means that you can query for SKUs that come in red instead of products that come in red.

In the second example, what if your SKUs have both color and size, but you only have certain sizes available for certain colors? For example, your fields with multiple values look like the following:

Product	Product SKU id	Size	Color	Clearance price
abc	abcSR	Small	Red	\$50
abc	abcSB	Small	Blue	\$50
abc	abcMB	Medium	Blue	NULL
abc	abcMR	Medium	Red	\$55
abc	abcLG	Large	Green	\$60
abc	abcLB	Large	Blue	\$60

In this case, the index connector feed is similar to the first example. You extend the list values so that all three attributes line up, as in the following:

```
<?xml version="1.0" encoding="UTF-8" ?>

<records version="1.0">
  <record>
    <id>123456</id>
    <product>abc</product>
    <sku-id>abcSR|abcSB|abcMB|abcMR|abcLG|abcLB</sku-id>
    <color>red|blue|blue|red|green|blue</color>
    <size>small|small|medium|medium|large|large</color>
    <clearance-price>$50|$50||$55|$60|$60<clearance-price>
  </record>
</records>
```

Notice that a `<clearance-price>` attribute for each of the SKUs was added to the feed. The clearance price also shows how you can handle a NULL attribute. In this case, the medium blue product (abcMB) does not have a clearance price, but it is still defined so that the attributes still line up.

Using field lists with separators requires that the data be pivoted, which can be awkward. An alternative is to use the explicit list schema that lets you iterate over each item in your data set without pivoting. If you are an e-commerce customer and have product-SKU data, consider using this alternative schema for your SKU data. In this final example, the feed results in building the same index as the second feed example from above:

```
<?xml version="1.0" encoding="UTF-8" ?>

<records version="1.0">
  <record>
    <id>123456</id>
    <product>abc</product>
    <list>
      <item>
        <sku-id>abcSR</sku-id>
        <color>red</color>
        <size>small</size>
        <clearance-price>$50</clearance-price>
      </item>
      <item>
        <sku-id>abcSB</sku-id>
        <color>blue</color>
        <size>small</size>
        <clearance-price>$50</clearance-price>
      </item>
      <item>
        <sku-id>abcMB</sku-id>
        <color>blue</color>
        <size>medium</size>
        <clearance-price></clearance-price>
      </item>
      <item>
        <sku-id>abcMR</sku-id>
        <color>red</color>
        <size>medium</size>
        <clearance-price>$55</clearance-price>
      </item>
      <item>
        <sku-id>abcLG</sku-id>
        <color>large</color>
        <size>green</size>
        <clearance-price>$60</clearance-price>
      </item>
      <item>
        <sku-id>abcLB</sku-id>
        <color>blue</color>
        <size>blue</size>
      </item>
    </list>
  </record>
</records>
```

```

        <clearance-price>$60</clearance-price>
    </item>
</list>
</record>
</records>

```

Notice that the abcMB SKU does not have a clearance price and, therefore, has an empty node (<clearance-price></clearance-price>). This empty node is necessary to ensure that the SKU data lines up properly within the index.

Regardless of which schemas you decide to use, a limitation of the list mechanism is that a field can become long. You have a maximum of 1024 values per field for "list-type" fields.

See [Anatomy of an index connector feed](#).

Field types

During the indexing process, an index connector feed translates data from the XML into pre-defined Search&Promote metadata fields. After these fields are indexed there are various ways to use them in Search&Promote.

following are some example uses of metadata fields in Adobe Search&Promote:

Field use	Description	Example fields
Relevancy	Match content against the keyword that a user enters in the search area.	Brand, Keywords, Title
Search by default	Search this field on every query.	Brand, Title
Type	The type of data to expect.	Number, Rank, Text
Range	Return a range of results.	Dates, Price
Result	Include in the result set.	Brand, Thumbnail, Title
Sorting	Offer sorting options.	Price, Title
Rank	Use for result weighting algorithms of search results.	Conversion Rate, Dates, Page Visits
Faceting	Include as a refinement facet in the search results.	Brand, Color
List	Treat this field as a delimited list of elements.	Category, Color

You can use some fields for multiple purposes. As shown in the previous table, the Brand field functions as a facet, is searched against by default, returned in the result set, and matched against the keyword that the user enters.

It is recommended that you define uses of fields early in the development process. If Adobe Consulting Services are engaged on the implementation, the consultant requests and also recommends the purposes of each field. With that information they generate a Metadata Specification that defines how each field is used. This specification lets the consultant and the customer to align on the purposes of each field. It also serves as a helpful reference document for the future.

Example of hierarchical category facets

If your product catalog has each product in a category then build facets for the various categories.

In the index connector feed, you specify the category level and the category name. For example, suppose that you have the following two products in two categories:

Id	Product name	Tier 1 category	Tier 2 subcategory	Tier 3 subsubcategory
6358695	Glass Heart Necklace	Jewelry & Watches	World Jewelry	Handmade Jewelry
10002004	Soccer Madness 2013	Games	Video Games	Playstation


The feed can look like the following where t1 indicates tier 1 within the category hierarchy, t2 indicates tier 2, and so on:

```
<?xml version="1.0" encoding="UTF-8" ?>
<records version="1.0">
  <record>
    <id>6358695</id>
    <prod-name>Glass Heart Necklace</prod-name>
    <t1>Jewelry & Watches</t1>
    <t2>World Jewelry</t2>
    <t3>Handmade Jewelry</t3>
  </record>
  <record>
    <id>10002004</id>
    <prod-name>Soccer Madness 2013</prod-name>
    <t1>Games</t1>
    <t2>Video Games</t2>
    <t3>Playstation</t3>
  </record>
</records>
```

Example of hierarchical category facet where products are in two categories

The following example uses a combination of a list field and the hierarchical facet.

```
<?xml version="1.0" encoding="UTF-8" ?>
<records version="1.0">
  <record>
    <id>6358695</id>
    <prod-name>Glass Heart Necklace</prod-name>
    <t1>Jewelry & Watches|Jewelry</t1>
    <t2>World Jewelry|International Jewelry</t2>
    <t3>Handmade Jewelry|Custom Jewelry</t3>
  </record>
  <record>
    <id>10002004</id>
    <prod-name>Soccer Madness 2012</prod-name>
    <t1>Games|Electronics</t1>
    <t2>Video Games|Consoles</t2>
    <t3>Playstation|Playstation</t3>
  </record>
</records>
```

 **Note:** *Not all products must be in two categories.*

Example of facets with more than one value

Facets typically have one value such as a price, a size, or a brand name of a product. In some scenarios you could have a facet return more than one value, such as a category name and a taxonomy name or an icon that belongs to that category. You supply one field with the primary value. Then supply another field with the combination so that the search engine sorts the two field-lists the same way when faceting.

In the following example, the search engine is configured to build a facet based on `t1` with the associated field `tax1`. To ensure the two field-list are sorted the same way, the taxonomy field is prefixed with the same value as the primary data. Then, at search time, the search engine removes the prefix within the Guided Search layer.

```
"1.0">
  <record>
    <id>6358695</id>
    <prod-name>Glass Heart Necklace</prod-name>
    <t1>Jewelry & Watches</t1>
    <tax1>Jewelry & Watches:sto123</tax1>
  </record>
</records>
```

Example of facets with ranges

Adobe Search&Promote does not currently support the ability to dynamically create ranges of individual point data at search time. Instead, this functionality is done at index time. Assigning a range to a result is done by updating the index connector feed with the assigned range in a user-defined field. The value assigned to this field displays in the search results exactly how it is represented in the feed. You can see this arrangement in the `<price-range>` node in the following example:

```
<?xml version="1.0" encoding="UTF-8" ?>

<records version="1.0">
  <record>
    <id>6358695</id>
    <prod-name>Glass Heart Necklace</prod-name>
    <t1>Jewelry & Watches</t1>
    <price>1099.00</price>
    <price-range>$1000.00 - $1250.00</price-range>
  </record>
  <record>
    <id>6358695</id>
    <prod-name>Glass Heart Necklace</prod-name>
    <t1>Jewelry & Watches</t1>
    <price>1099.00</price>
    <price-range>$1000.00 - $1250.00</price-range>
  </record>
</records>
```

You can also use **Scripted Filtering** in Adobe Search&Promote. You simply create a script that assigns ranges at index time without the need for updating the feed. However, this approach is not recommended because it makes future modifications to the range values difficult. The range values would be defined in a scripting variable that could require Adobe Consulting Services or the implementation engineer to change.

Example of an XML and XSD index connector feed

- [XML-based index connector feed example](#)
- [XSD-based index connector feed example](#)

XML-based index connector feed example

XSD-based index connector feed example

Transforming data at the time of indexing

Indexing a feed is faster if Adobe Search&Promote does not have to transform the data. However, there are certain situations where it makes sense to have the software perform transformations of data at the time of indexing, such as the following:

- Binning of data points for a range facet.
- Prefixing one field with another for associated facets.
- Slotting for dynamic facets.

Consider working with Adobe Consulting Services to decide what data transformations you require.