OperationROI
A Tactical Approach To Internet Marketing

2536 N. Federal Highway
Ft. Lauderdale, FL 33305
1-888-277-5429
www.OperationROI.com

Feed Source File Specifications

This document describes the source file format and field requirements for imports into your OperationROI account. OperationROI can pick up your tab or comma delimited product source file (.txt or .csv) via FTP or HTTP from our -or- your servers. We can also import XML files generated according to OperationROI 's XML schema.

Using a Current Shopping Feed as Your Feed File

If you already have an existing feed file, such as Google Product Search, we can most likely use it as your source file. You will simply need to validate the source file prior to import to be sure the formatting is correct and the data quality is good.

> **Important Note:** Once your source file is uploaded and validated with OperationROI, ***the file name and format must remain unchanged***; you may add fields to your source file (up to 50 in total), but may not rename or remove existing fields.

Sample source files:

- Products.csv
- Products.xml

Comma/Tab delimited file format

- Feed files must be provided in a UTF-8 encoded format
- The first row must contain the field names separated by tabs or commas
- Each product listing must be on a separate line within the file
- You must make sure that no line-breaks are contained within the field content, only between product listings

Required/Suggested Fields

Below are the required and suggested fields for creating your OperationROI source file. These fields represent the most common fields found throughout the shopping site feeds. You may add more fields to your source file as needed.

Note: Field names do not have to be named the same as listed below

Field Name	Description	Required
Unique ID	The value that uniquely identifies the product in the merchant's system, such as product SKUs No duplicates	Yes
Name	The product name Between 15 – 70 total characters, usually Brand Adjective Adjective Noun, use appropriate keywords	Yes
Description	The product long description 750 total characters, include relevant keywords	Yes
Price	The product price Two decimal point format without $ sign	Yes
Merchant Category	The category the product belongs to in the merchant's internal system Do not name it non-category names like: On Sale, Misc., Accessories, New Arrivals - be specific such as: Women's Shoes > Sandals or Ergonomic Office Chairs	Yes
URL	The URL to the product details page on your site Must begin with http:// and contain same domain name	Yes
Image URL	The URL to a photo of the product Begin with http:// and end with .jpg, .tif, .png or .gif	Yes
Manufacturer	The manufacturer name of the product	Yes
Manufacturer Part Number	Product part number from the Manufacturer	Yes
Brand	The brand name of the product	Yes
Keywords	Brief, relevant keywords or search terms for the product separated by commas Up to 10 keywords	Suggested
Shipping Price	The shipping price of the product Generally shown is lowest price for ground shipping	Suggested
Quantity	The quantity of products in stock	Suggested
Weight	The shipping weight of the product	Suggested
Condition	The condition of the product New, Used, Refurbished, etc.	Suggested
UPC	The UPC of the product Must be 12-14 digits	Suggested
Sale Price	The marked-down sale price of the product	Suggested

OperationROI XML file format

The OperationROI XML file format is ideal for platform integrations, although it can be used for a single store import. It is also the preferred format for large product set imports for stores with tens of thousands of SKUs, since it has a built-in paging mechanism to make the import process more efficient.

There are three XML node sections to this file's schema, **Fields, Products and Paging;** each is discussed below.

Fields node

The **Fields** node section lets OperationROI know what product attributes to expect for your import. This node contains multiple child **Field** nodes (up to 50), each representing a product attribute to be imported from your system. Each Field node must contain a **name** attribute that contains the field name as a value.

For example:

```
<Fields>
        <Field name="UniqueID" />
        <Field name="Name"/>
        <Field name="Description" />
        <Field name="Price" />
        More Field nodes…
</Fields>
```

Products node

The **Products** node section provides your product listings to OperationROI. This node contains multiple child **Product** nodes, each representing a product in your system. Each **Product** node must contain child nodes named according to the **Fields** node section, **all** fields from the **Fields** node section must be present in each **Product** node and the content within each product field node must be contained in <![CDATA[ ]]> to avoid parsing errors caused by HTML tags within the content.

For example:

```
<Products>
        <Product>
                <UniqueID><![CDATA[6816916]]></UniqueID>
                <Name><![CDATA[Apple iPod Video 30GB White 5.5 GEN]]></Name>
                <Description><![CDATA[With 5.5 generation iPods all your music at your
                fingertips you may never want to stop listening…]]></Description>
                <Price><![CDATA[237.99]]></Price>
                More fields…
        </Product>
        More Product nodes…
</Products>
```

Paging node ***(optional section)***
The **Paging** node section allows OperationROI to relay paging calls to your system, providing an efficient mechanism for large product set imports and eliminating server timeouts on large XML files.

Important Note: Do **NOT** include the Paging section if you are not using a dynamically loading page that reads the query string and loads the products in batches as noted below

The **Paging** node contains three child nodes:
- **Start** – This is the starting index of the first product to be returned
- **Count** – The number of products to return
- **Total** – The total products to be returned for this import

You must populate each of these nodes as calls are made to your system.

For example:

```
<Paging>
        <Start>1</Start>
        <Count>100</Count>
        <Total>1000</Total>
</Paging>
```

OperationROI will make calls to your system via HTTP and will pass query string parameters that indicate which product page to return.

The HTTP call will be similar to the following:
http://www.MyStore.com/MyWebPage?start=1&count=100

To retrieve the second page of a 1000 products file the call will be similar to the following:
http://www.MyStore.com/MyWebPage?start=101&count=100