FusionBot Data-Extraction Setup / Documentation

Overview

FusionBot's data-extraction based indexing provides for more accurate indexing and results precision, as well as the ability to custom define specific "database" like fields for displaying, searching, and sorting within the search results.

Data-extraction based indexing works by partnering with our customer to generate a custom "extract file" of their items to be made searchable, rather than "parsing" content from HTML pages through the use of our crawler. The creation of an extract file provides for a more robust and accurate representation of your items, while at the same time reducing the load and bandwidth requirements on your hosting provider / web server.

We can work with our customer to either remotely, from FusionBot's side, extract the requested data directly from their database in the format / syntax required, provided the customer is able to grant FusionBot permission / remote access, or, your technical team can create a custom extract file / script and host this file / script on your server for FusionBot to retrieve on a scheduled basis. The basis of this document is to provide details for how to extract / construct the data from your database in a format required for FusionBot to read / build your index upon.

When utilizing the data-extraction method, the extract file allows you to define your own custom fields for searching and sorting your search results upon. In addition, the extract file can contain custom fields in order to re-display specific database type values as part of each matching result, such as an item's price, product number, brand, minimum order quantities, etc. The end result is the appearance of a more database-like search result, while benefiting from the advanced performance, features, and functionality that an index-based search solution provides.

Syntax

The extract file **ALWAYS** consists of exactly 4 lines per product (item) to be included in your index, consisting of the following:

Line 1: URL of Product Page
Line 2: Page Title
Line 3: Page Description
Line 4: Page Body / Content

If you were to have 4 items in your entire database, your file **MUST** contain exactly 16 lines, 4 for each item.

Tabs and new-lines must be removed from each entry prior to writing the record. Tabs and new-lines (chr(10)) are delimiters, that is, they define each line (or field) in your file, and therefore cannot be included **within** each line of your file.

Carriage returns must also be eliminated (chr(13)).

There is only one exception for inclusion of a tab anywhere in the export file, and this is when additional (optional) tags / indicators are assigned to a page on the URL line (line 1), example:

http://www.logika.net/ <TYP>PDF

The indicators are XML-like attributes that assign various optional settings (values) to a page, such as:

LEN - Length of Page (bytes)
LMD - Last Modified Date
TYP - File Type (pdf,rtf,ppt,doc,xls,txt)
CCH - Cache File Position
TNI - Thumbnail Image
TNW - Thumbnail Width
TNH - Thumbnail Height
TNU - Thumbnail Url/Href
TNT - Thumbnail Target
REL - Related/Region

The above indicators are FusionBot "reserved" indicators that cannot be used elsewhere on line 1. The reason this is important is that, in addition to the above indicators, you can custom define any of your own 3 letter xml-like indicators for passing in custom "database like" values to be part of each item's (page's) search result, such as the display of a price value for each item, or the display and searchability of an product id for an item. For example, line 1 of an extract file could appear as:

http://site.com/index.asp?id=5 <prc>24.95<pid>1557663327<brd>AMCO

For this above URL, you are assigning a price, a product id, and a brand to be associated with this particular item, so that visitors can subsequently search or sort based on these values, or so that you may re-display these values next to each item in your results. It is difficult, if not impossible, to achieve similar results when using a spider to simply crawl your pages.

It is also important to note that for **ANY** of the 4 individual lines that **ALL** HTML based tags **MUST** be removed from your text before including this text in your extract output. Therefore, if your database includes HTML characters / entities within your product title or description fields, you must write a function to strip these tags out before writing this data to lines 1,2,3 or 4 of your extract file.

Think of the file that you are building as essentially the same type of file that the FusionBot spider builds during its crawl of your site. Since you are eliminating the use of the FusionBot crawler using the extract approach, the extract file you create needs to be in the same format that would be created had the FusionBot spider indexed your site instead. During FusionBot's crawl process, your HTML pages are parsed and all text is extracted for creating your searchable index upon, and all HTML tags are discarded.

Once you have built your custom extract script, you can either schedule your script to run, which will then place a static extract file in a specific location on your web server, or, you can point FusionBot directly to the script itself, so that when our

"spider" initiates a connection to your server, FusionBot can call this script in order to dynamically generate the data-extraction based content on the fly. Either way, during the initial setup of your FusionBot account, we configure your account to look for your extraction file or script in the location you specify. Based on the existence of such a parameter in our system, FusionBot therefore knows to request an extraction of your content, rather than crawling your site using our spider-based indexer.

Search Results Configuration

Now that your extract file has been generated and your index has been built, you now need to be able to format your search results using the custom FusionBot template language / process. Details of how the FusionBot template process works can be found within our Template Documentation PDF.

In reviewing the Template Documentation you will see that FusionBot represents dynamic search results content, which gets swapped out in real-time, during a query, using what we call "template object tags". These object tags begin with either **$LGK_** or **$RES_**.

For example, to display the TITLE of a particular item in your search results, your template contains the tag **$RES_TITLE,** and, to display your item's description, the tag used is **$RES_DESCR.**

The above are reserved values defined by FusionBot for displaying these common fields in your search results, as extracted from your HTML pages when indexing, or in the above example, as extracted from Lines 2 & 3 for each item in your extract file.

The reason this is important is to therefore understand how to re-display the "custom" values you've assigned to each item after the tab in Line 1 of your extract file. For example, in the sample Line 1 above we specified:

http://site.com/index.asp?id=5 <**prc**>24.95<**pid**>1557663327<**brd**>AMCO

For an item with a URL on your site of:

http://site.com/index.asp?id=5

And therefore, within your FusionBot template, to display price, product id, and brand for each of your items, you would place the FusionBot object tags:

$RES_PRC, **$RES_PID**, and **$RES_BRD**

In the location you want these values to display for each item within your results.

Wherever in your template you wish to replace the value of a custom 3-letter tag assigned via Line 1 in your extract file, simply construct the object tag as **$RES_XXX**, where **XXX** is the name of the 3 letter indicator assigned.

Search Filters Configuration

Data-extract customers can also exploit the custom values assigned to each item via Line 1's XML-like indicators to enable visitors to "filter" their results by specific price ranges, product categories, brands, etc.

By doing so, visitors can further refine the scope of their results by "browsing" within categories. Filters enable you to display the total number of items that fit within a

specific range / category you assign, that users can then click-into to narrow their results based on their selected criteria.

Following is an example screen capture of the search filters feature:

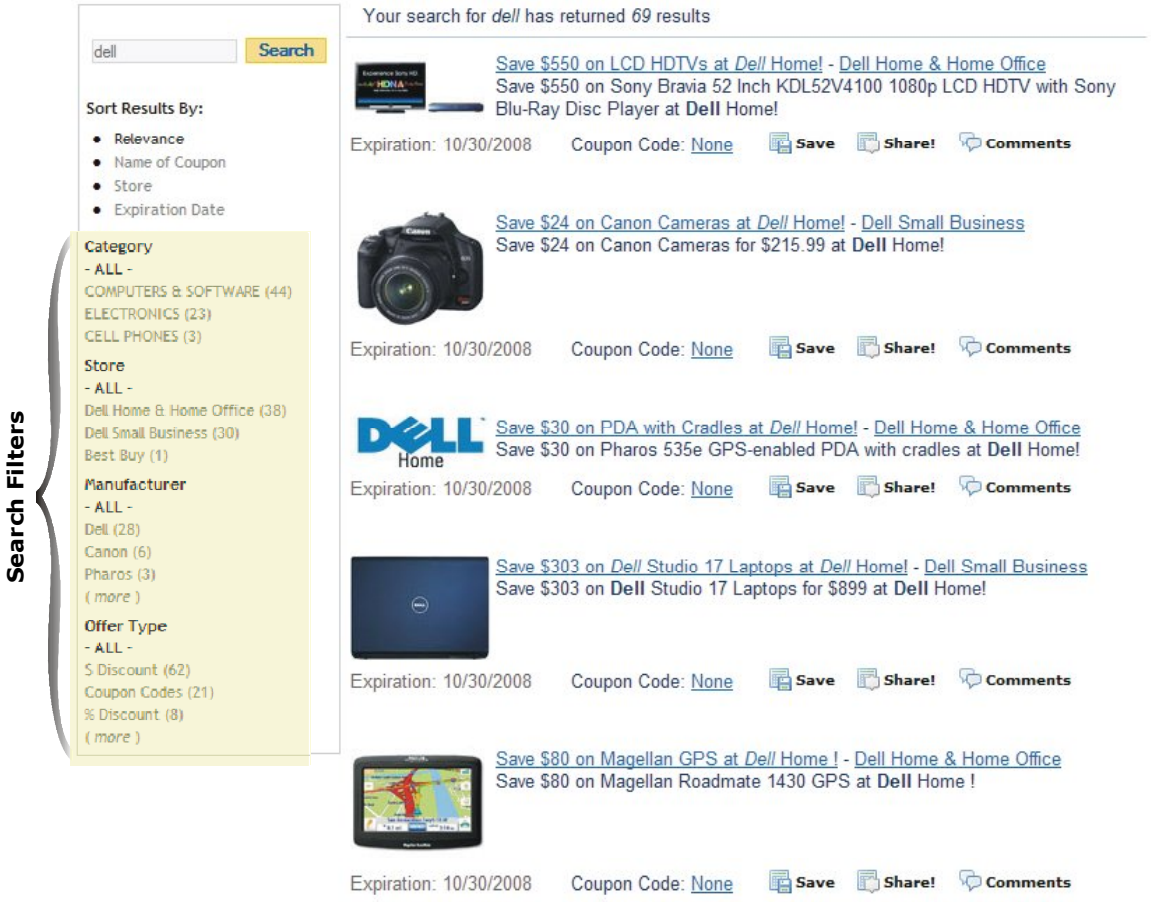

Figure 1

To enable search filters based on any of the fields assigned in Line 1 of your extract file, take note of the names of the 3 letter XML-like indicators assigned to each field, and enter the value for each indicator you wish to create a filter upon, separated by a comma, in your FusionBot account by selecting the 'search filters' link from the 'customization' tab.

Following is a sample entry in the search filter form within the FusionBot account corresponding to the output shown in figure 1 above:

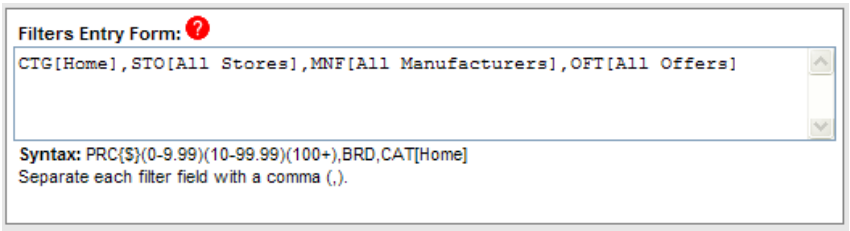

Figure 2

Each filter entry **MUST** be delimited by a comma, and **MAY** contain up to three types of attributes, in any order. The characters surrounding the values you are attempting to set define the type of attribute. The "surround characters" can be parenthesis, brackets, or braces, with the following meaning:

Surround Characters	Meaning
() Parentheses	**RANGE:** For numeric fields in your extract file, enables you to specify a range for which each item in your results should be assigned. The most common use is to display a price range filter, enabling visitors to view items only within the price range they select based on the ranges you define using this option. The last range option should contain a single value followed by a plus sign, in order to encompass all items assigned a value exceeding the highest value specified. Example: PRC(0-9.99)(10-99.99)(100+)
{} Braces	**Prefix:** Enables the assignment of a prefix to be displayed for any configured filter. The most common use is to assign a currency symbol to your filter's display range, since the numeric values in your extract file **MUST NOT** contain any alphabetic characters in order to be considered numeric, and thus provide for the ability to create a numeric range for display. Example: PRC{$}(0-99.99)(100-199.99)(200+) The above example creates a filter for display similar to the following: Price - ALL - $0 - $99.99 (2) $100 - $199.99 (3) $200 + (7) By using the braces prefix {$}, when the filter values / ranges print, FusionBot automatically inserts the value between the braces to the front of each range value.
[] Brackets	**Node Name:** When visitors select a particular filter in their results, a visual "cue" can be displayed specifying the "collection" of filters that have been selected. Visitors can then elect to "click out" of specific filters as desired. The display text used for the top-level node name is assigned between the brackets. The object tag **$LGK_FILTERTAG** controls where in your results page the visual cue displays. Example: CAT[Home],STO[All Stores],OFT[All Offers]

Table 1

Figure 3 below displays the result of "browsing" into the filters: Category, Store, and Offer Type, on the left-hand side, and the resultant display, via the $LGK_FILTERTAG, of the nodes selected. Please note that nodes selected will **ONLY** be displayed for a particular filter if a top-level node name has been assigned via the brackets attribute.

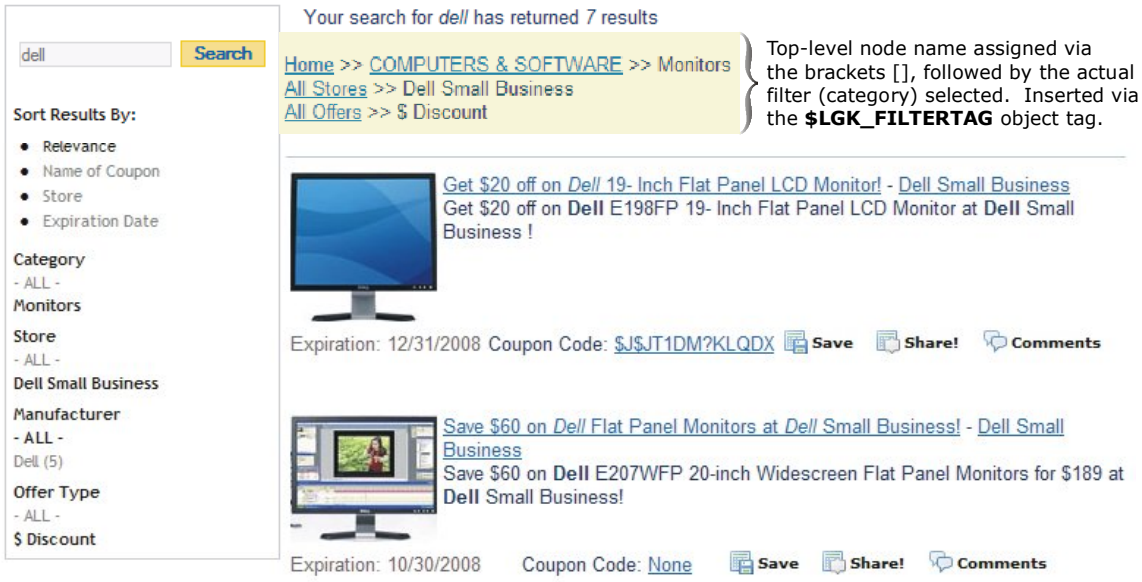

Figure 3

For alphabetic fields, such as Category (CTG) in the above example, all that **MUST** be entered is the 3-letter indicator. The output sort order for alphabetic listings will be first by number of matches then by name.

Once your entries are saved in your search filters form, you will then need to modify your search results template to incorporate the necessary template object tags for displaying your filter options.

Filter object tags **ALWAYS** begin with **$FLT_,** followed by the 3-letter indicator assigned to the particular filter. Thus, wherever you wish to display within your search results the filter data for store (STO), or manufacturer (MNF), for example, as shown in figures 1 & 3, you would insert into your template the custom object tags:

$FLT_STO
$FLT_MNF

The default maximum number of options (rows) to display per filter is 5. Which means if more than 5 options are present, a '***more***' link will be displayed. To override the default value of 5 for display of any filter, include the maximum number of rows to display between parentheses, in your results template, as follows:

$FLT_STO(4)
$FLT_MNF(8)

Following is the example template code used to display the output as shown in figures 1 & 3 above:

```
<style>
.flt { font: 11px trebuchet ms; text-decoration:none; margin:5px; color:black;}
.flt a:link { text-decoration:none;color:gray;}
.flt a:hover { text-decoration:underline;color:#0768A9;}
.flt a:visited { text-decoration:none;color:gray;}
.flt a:hover { text-decoration:underline;color:#0768A9;}
</style>
$FLT_CTG_IF
 <div class=flt>
   <div><b>Category</b></div>
   <font color="black">$FLT_CTG(3)</font>
 </div>
$FLT_CTG_END
$FLT_STO_IF
 <div class=flt>
   <div><b>Store</b></div>
   <font color="black">$FLT_STO(3)</font>
 </div>
$FLT_STO_END
$FLT_MNF_IF
 <div class=flt>
   <div><b>Manufacturer</b></div>
   <font color="black">$FLT_MNF(3)</font>
 </div>
$FLT_MNF_END
$FLT_OFT_IF
 <div class=flt>
   <div><b>Offer Type</b></div>
   <font color="black">$FLT_OFT(3)</font>
 </div>
$FLT_OFT_END
```

Figure 4

Sort Options Configuration

You can also provide for your users the ability to sort their search results by any XML indicator in your extract file. In this manner, in addition to the default option for sorting results by rank / relevance, your visitors can choose to re-order their results for display by any pertinent XML indicator you specify (see figure 5 below).

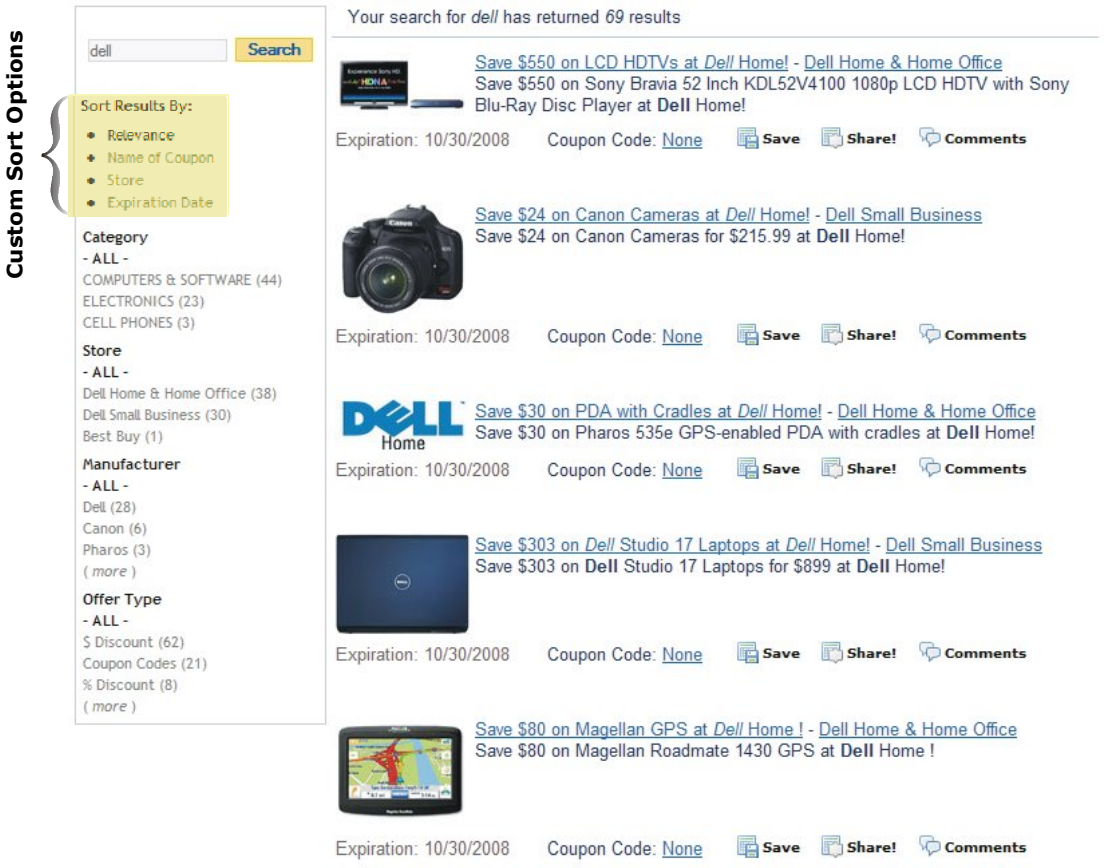

Figure 5

The search template HTML corresponding to the highlighted sort options in figure 5 above, would be constructed as follows:

```
<div class=flt><b>Sort Results By:</b></div>
<ul>
   $LGK_CMPQS_IF(nsrt,TTL_A0)
    <div class=flt>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=0"><font face="trebuchet ms">Relevance</font></a></li>
    <li><font face="trebuchet ms">Name of Coupon</font></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=STO_A0"><font face="trebuchet ms">Store</font></a></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=EXP_A0"><font face="trebuchet ms">Expiration Date</font></a></li>
    </div>
  $LGK_CMPQS_ELIF(nsrt,STO_A0)
    <div class=flt>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=0"><font face="trebuchet ms">Relevance</font></a></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=TTL_A0"><font face="trebuchet ms">Name of Coupon</font></a></li>
    <li><font face="trebuchet ms">Store</font></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=EXP_A0"><font face="trebuchet ms">Expiration Date</font></a></li>
    </div>
  $LGK_CMPQS_ELIF(nsrt,EXP_A0)
    <div class=flt>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=0"><font face="trebuchet ms">Relevance</font></a></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=TTL_A0"><font face="trebuchet ms">Name of Coupon</font></a></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=STO_A0"><font face="trebuchet ms">Store</font></a></li>
    <li><font face="trebuchet ms">Expiration Date</font></li>
    </div>
  $LGK_CMPQS_ELSE
    <div class=flt>
    <li><font face="trebuchet ms">Relevance</font></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=TTL_A0"><font face="trebuchet ms">Name of Coupon</font></a></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=STO_A0"><font face="trebuchet ms">Store</font></a></li>
    <li><a href="/b/q?$LGK_CNEXT(nsrt)&nsrt=EXP_A0"><font face="trebuchet ms">Expiration Date</font></a></li>
    </div>
  $LGK_CMPQS_ENDIF
</ul>
```

Figure 6

Essentially, what is created is an **IF/ENDIF** block corresponding to the number of sort options to display, with the **ELSE** block being set to the default sort option (typically rank / relevance).

Each **IF** block starts with the evaluator object tag:

$LGK_CMPQS_IF(nsrt,XXX_(A,N)_(0,1))

This instructs FusionBot to analyze the querystring URL looking to see which sort option has been selected in order to determine which **IF** block of HTML should be printed.

CMPQS, in the object tag above stands for "Compare Query String". The value '**nsrt**', $LGK_CMPQS_IF(***nsrt***,TTL_A0), tells FusionBot which variable to look for in the querystring, followed by **TTL_A0**, $LGK_CMPQS_IF(nsrt,**TTL_A0**), which tells FusionBot what do if the variable instructed to look for (nsrt) is set equal to this value (TTL_A0). When the condition is met, that is, when the sort option TTL_A0 has been clicked, the corresponding HTML in the applicable **IF** block will print, and the results will be sorted accordingly.

Thus, in the above example, the querystring sent to FusionBot would look similar to:

http://ssXXX.fusionbot.com/b/q?sn=123456&nsrt=TTL_A0

In analyzing the querystring, FusionBot sees that the value of 'nsrt' is set to 'TTL_A0', and thus the HTML **IF** block where this condition has been met is printed. For the block where the condition is met, this means that the results have been sorted by the 3-letter indicator passed in, and thus, that particular indicator would be the only sort option **NOT** surround by an HREF link, signifying that this is the active / selected sort option (i.e. it is not clickable).

In the syntax example: **$LGK_CMPQS_IF(nsrt,XXX_(A,N)_(0,1))**

Which translated to: **$LGK_CMPQS_IF(nsrt,TTL_A0)**, in our first instance, the option *A* or *N*, as specified via **$LGK_CMPQS_IF(nsrt,XXX_(A,N)_(0,1))**, tells FusionBot whether to sort the results either **A**lphabetically, or **N**umerically, and the option **$LGK_CMPQS_IF(nsrt,XXX_(A,N)_(0,1))**, tells FusionBot whether to sort the results in ascending (0), or descending (1), order.

Thus, putting it all together:

$LGK_CMPQS_IF(nsrt,TTL_A0), tells FusionBot to analyze the querystring, look for the variable named 'nsrt', and when equal to TTL_A0, print that which is in the corresponding HTML block.

Similarly, within each HTML **IF** block, you will see syntax such as:

<a href**="/b/q?$LGK_CNEXT(nsrt)&nsrt=STO_A0**">Store

This constructs the clickable link for each sort option, again, instructing FusionBot how to sort the results when the applicable link is clicked. In the above example, when the link labeled 'Store' is clicked, the results will be sorted by the XML indicator 'STO', alphabetically and in ascending order.

The value **$LGK_CNEXT(nsrt)** is a reserved object tag that dynamically populates the link with any additional / necessary querystring parameters (minus the currently selected sort option) for the query to function properly and **MUST** be included. The '**/b/q?**' string points to the FusionBot 'cgi-bin' and 'query' program accordingly.

For sorting by any custom XML tag, the value of the 'nsrt' parameter **MUST** be the 3-letter indicator, followed by an underscore, followed by the sort order options. FusionBot is able to discern sorting by a custom XML tag, versus sorting by default rank, when the value of 'nsrt' is set equal to 0 (sort by rank), versus set equal to a 3-letter indicator, followed by an underscore, followed by the sort options (A/N_0/1).

Conclusion

If you have any additional questions concerning any of the information contained within this document, don't hesitate to send an inquiry to our support staff, as we are always more than happy to help.