SellerDeck

ALL YOU NEED TO SUCCEED

SellerDeck 2013

Ajax API Information

Hugh Gibson, CTO SellerDeck

Rev 1.0 October 2012

Contents

1 Introduction

This document gives some details of the Ajax API implemented in SellerDeck 2013.

Note:

- Familiarity with the new features is assumed.
- Not all information is given. It's designed to give a flavour, not exhaustively document each call.
- The source code for the new APIs is available in the Perl scripts. Check how they are used in the JavaScript.
- The best tools to use to fully understand the code is a good set of browser developer tools, e.g. those accessed by pressing F12 in IE9 or Chrome.

2 Background

A fundamental principle of SellerDeck/Actinic Desktop has been to serve up statically generated pages. These are fast and well ranked by search engines. In recent releases some Web 2.0 techniques have been used to extend what has been shown on these static pages including display of stock levels after loading the page by querying the server from the browser - Ajax calls.

SellerDeck has always had a set of index files on the server for the search functionality but these were rarely used to their full power. The main index relates words, attributes and properties to product references. An additional index file relates product references to full information about the product.

With SellerDeck 2013 we have retained the static pages but have used Ajax to make the consumer's experience as smooth as possible for the new filtering pages. These calls exploit the indexes, which have been extended to include sorting information (e.g. price ascending, relevance) that can be applied in the server or web client.

3 Relevance

A fundamental extension to the index files is the concept of Relevance for words. This has been used to drive the auto-suggest functionality as well as sorting search results by relevance. Though this isn't an Ajax API, it is used to drive some of the APIs so some understanding of how it works is useful.

Relevance is calculated for every word encountered in the text which is indexed within the shop. This is controlled from Search and Filtering Settings, Options tab, Indexing Options group.

3.1 Configuration

Hidden in the setup file are some further options which control the weighting applied for various locations of a word. These are code 600 (nSearchWeightShortDescription) through 609 (nSearchWeightPageFragmentText). So, for example, a word appearing in a product short description has a higher weighting (150) than a word in a description (100).

If a word appears more than once in a field the weighting for that word is multiplied by the number of occurrences, up to a maximum of nSearchWeightMaxWordMultiplier, code 610 (default value 2).

These fields cannot be configured through the UI of SellerDeck as they are fairly esoteric. However, they may be useful and the default values chosen may need to be tweaked.

3.2 Word Relevance for Products

Each word encountered has an individual relevance stored for each product. This is up to a maximum of 4095 as it is encoded in two characters to save space. Therefore when searching for words to find products that match, each product associated with the word also has a relevance score for that word. When searching for multiple words, the relevance scores are added up for each product, giving an overall relevance for each product to the search terms entered.

Try searching for "men" in the sample store. The "Mens engraved bracelet" appears first followed by the "Gold Chain". The first product has "men" in the title so gets a higher relevance score than the second product, which has "men" in the description. The third product, "Wedding Rings", only has "men" as the title of an option so has a lower weighting again.

Another refinement of this is when doing partial matches. If a search is made for "egg" then that will match "egg" and "eggplant" if Match Whole Words Only (in Search And Filtering Settings, Options tab, Indexing Options) is not checked. However, the relevance score for "eggplant" is decreased by the ratio of the length of the match - so in this case it's multiplied by 3/8. If the hit was "eggs" then the relevance would be multiplied by 3/4. Therefore closer matches will have higher relevance.

3.3 Overall Word Relevance

For each word encountered, the overall relevance of that word is calculated. This is a summation of the relevance of the word for each product and is only limited to 2^32. It is stored with the word in the search index file.

When deciding which word to auto-suggest, the words that start with the text entered by the customer are sorted by their overall word relevance, and the first 10 shown.

Try typing "p" into the search box in the sample store. The words that appear start with "pearl" as that appears in the short description for lots of products. The last word suggested is "pair" which appears only once in a product description.

4 Auto-suggest

We have implemented an auto-suggest function for any search text box. This shows suggested words in a drop-down list below the text box, updating dynamically as the user types. The user is able to click in the list or select using up/down and Enter.

The Ajax call to support this is implemented in SearchScript.pl, short form ss.

A typical call, on typing "p" into the quick search box in the demo shop, is:

```
http://localhost:8080/cgi-bin/SD2013/ss000001.pl?ACTION=MATCH&TEXT=p
```

This requests a MATCH with the text p.

The response is:

```
{"result":1, "matched":"p", "count":7,
"words":["pearl","perfect","perfectly","pendant","peridot","picture","pair"]}"]}
```

This gives the words that start with p, sorted by relevance.

Other parameters implemented but not used by the JavaScript are:

`MAXRESULTCOUNT` - change the maximum number of words returned. The default is 10. It can be increased to any number (within reason) so in theory you can extract all the words used in the online catalogue with 26 calls. That could be used for rudimentary error checking as typo words will be indexed like any other.

`INCLUDERELEVANCE` - include the raw relevance score for each word.

```
http://localhost:8080/cgi-
bin/SD2013/ss000001.pl?ACTION=MATCH&TEXT=p&INCLUDERELEVANCE=1
```

```
{"result":1, "matched":"p", "count":7,
"words":["pearl",850,"perfect",300,"perfectly",200,"pendant",150,"peridot",150,"picture",150,"
pair",100]}
```

This could be used to not show suggestions if they have a low relevance. If showing all the words in an online catalogue it could be used to show words sorted by relevance. Typos would congregate at the bottom if they didn't appear very often.

5 Searching

Searching uses the search-results page from v11 and earlier. The results are created by the Perl Scripts as before. The Perl uses templates embedded into the page to create the search results.

This isn't an Ajax request, but some of the parameters are common with the filtering Ajax requests so some explanation is helpful.

A typical query is:

```
http://localhost:8080/cgi-
bin/SD2013/ss000001.pl?page=search&SS=diamond+ring&PR=-1&TB=O&ACTION=Go%21
```

`page=search`
: Perform a search.

`SS=diamond+ring`
: Search for the words "diamond" and "ring"

`PR=-1`
: Don't restrict by price band.

`TB=O`
: Boolean operation on multiple words. "A" means to only find hits where All words are present (short for AND) and "O" means to find hits where Any word is present (short for OR). Note that with Relevance sorting being the default, OR searching becomes feasible as a default option so search results are always given.

6 Filtering

When opening a filtering page the browser calls the server and requests a list of all the products that match the filtering specification. The list is a compact set of product references including sorting information. Details of the products, as required, are loaded in a separate call. These are retained in the browser and are used when going through pages of filtering results or calculating counts for filter options. This ensures that calls to the server are kept to a minimum.

A restricted set of functionality for filtering has been implemented purely in the server so that consumers with JavaScript disabled in the browser can still use filtering. This is also used for when the page must display the filtered products initially.

Section pages that include filtering have results templates build into them that can be used by the Perl Scripts or the JavaScript.

6.1 Initial Perl script call

The Wedding Jewellery section has a URL of:

```
http://localhost:8080/cgi-
bin/SD2013/ss000001.pl?FILTERPAGE=WeddingJewellery.html&TB=A&FT=GF&SX_ID=0&
S_514_0=&Action=Search
```

This specifies a filter, and uses the results template built into the page to show the results.

The parameters are:

`FILTERPAGE=WeddingJewellery.html`

Specifies the section page to be used for filtering

`TB=A`

See searching above

`FT=GF`

Indicates that the Filter Type is a General Filter. In this case we create a set of search commands from the list of parameters.

`SX_ID=0`

Section ID. Only supply products from this section. 0 means all sections.

`S_514_0=`

Indicates the value for an option. Each set of options in the left has a code - 514 - and the trailing number indicates what sort of search. 0 = text, 1 = integer and 2 = date. In this case no value is specified for the option, indicating "Any".

`Action=Search`

Do a search rather than Match (for auto-suggest) or other actions.

6.2 Get Products that match Filter

Once the page has been loaded from the Perl Script, the JavaScript takes over and makes Ajax calls to show matching products or update filtering counts as necessary. Note that counts are only implemented in the JavaScript.

As product counts are enabled in the sample database, after opening the Wedding Jewellery section the JavaScript makes an Ajax call to:

```
http://localhost:8080/cgi-
bin/SD2013/ss000001.pl?PAGE=DynamicFilter&TB=A&FT=GF&=&SX_ID=0&SO=1_3_4_0&S
_514_0=&Action=Search&timestamp=1350282477666
```

The parameters are:

`PAGE=DynamicFilter`

Indicates that we are filtering. Absence of DPR parameter indicates we are just getting the list of products that match the filter.

`FT=GF`

See Perl filtering above.

`TB=A`

Combine search results for words using "ALL" instead of "ANY". Not used unless some search words are being specified which isn't implemented here.

`SX_ID=0 or SD=0`

See Perl filtering above.

`S_514_0=`

See Perl filtering above.

`&SO=1_3_4_0`

Not actually used with getting filtering results as an Ajax call, but is used elsewhere e.g. when the Perl Scripts have to sort the results. In other searches this indicates the Sort Orders to apply in sequence. The numbers refer to primary, secondary and tertiary sort orders and are listed in the database in the SearchSortOrders. The nSortID is fixed for each sort order.

`&Action=Search`

See filtering above.

This call returns a result like:


```
{"ProductSearchResults": { "ResultCount":32, "ResultSet":
["0:060025","0:030J39","0:030K38","0:0?0P3","0:030B35","0:030N20","0:080623","0:070M26","0:010
E45","0:030236","0:080I1!28","0:010Q44","0:030C34","0:030A29","0:020R1","0:010D41","0:000743",
"0:040?21","0:000442","0:030830","0:080924","0:050L22","0:080I28","0:000346","0:030F31","0:030
O37","0:030G33","0:06001!25","0:040H27","0:030132","0:030540","0:090S2"] } }
```


There are 32 results. Each result is a string like "`0:040H27`", termed a decorated reference. The 0: indicates that relevance is 0 for all products, because no text has been searched for. Note that in this case the second sort-order for Relevance kicks in, defaulting to Price Descending.

The next characters define sort orders. These are encoded in a custom base-64 encoding that preserves numeric sort order when sorted in a text form. All that is needed is to work out where the sort orders start (done through leading characters - no leading character implies two characters for the sort order, open curly brace { implies three characters, and closing curly brace } implies four characters) and how many there are, defined through pg_sSortOrdersPrependedToProdRefs defined globally in each page. For the best handle on this, see the JavaScript function doSort in actinicsearch.js.

In this case there are two sort orders defined, and they are two characters each. Therefore the product reference is what is left over, or 27.

When a filter option is clicked on, the filter specification is refined. For example, if Earring and Locket are checked, the call to the server includes:

`S_514_0=Earring&S_514_0=Locket`

This gives a different set of products.

6.3 Get Details for Products

Once we have a list of products that match the filter, we have to request details of those products in various circumstances.

If filter counts are enabled then all product details have to be requested so the counts can be calculated. If no counts are being shown then details of the products that are being displayed on the page have to be requested. In this case, if stepping through multiple result sets, a request would be made for products on the next page etc.

Once product details have been loaded into the browser they are remembered. Therefore changing sort orders or moving through pages of results will not cause any further Ajax requests.

A typical request, after opening the Wedding Jewellery page, is

```
http://localhost:8080/cgi-
bin/SD2013/ss000001.pl?PAGE=DynamicFilter&DPR=3_2_23_24_1!28_28_26_1!25_25_
22_21_27_32_36_40_30_29_35_34_31_33_39_38_20_37_1_41_45_44_46_42_43&timesta
mp=1350305180497
```

This indicates that details are required for the products listed in the DPR parameter. Typical results are below (some omitted, formatting added):

```
{"ProductDetails":
        { "ProductInfoCount": 32, "ProductInfoSet": [
        {"ProductInfo":
["3","1","Solitaire&#32;Diamond&#32;Ring","","Rings","s&#45;EngagementRing&#46;jpg","Solitaire
&#45;Diamond&#45;Ring&#45;3&#46;html&#35;SID&#61;1","GBP","15000","2"],
        "Properties": {"PropCount": 2, "PropertySet": ["625_0: Women&#33;","514_0: Ring&#33;"]
} },
        {"ProductInfo":
["2","1","Wedding&#32;Rings","","Rings","s&#45;WeddingRings&#46;jpg","Wedding&#45;Rings&#45;2&
#46;html&#35;SID&#61;1","GBP","13000","2"],
        "Properties": {"PropCount": 2, "PropertySet": ["625_0: Men&#33;","514_0: Ring&#33;"] }
}, ... }] } }
```

This list gives information about the products as:

- Product Reference
- Section ID
- Product Name
- Description
- Section name
- Image
- Anchor to find product
- Currency
- Price
- Decimals

It also then gives a list of Properties. The values relate encoded property to value. It would be possible to extract the name of the property from the filter options in the page, and provide a facility to do things like product comparison, listing all the values for the options.

7 Standard Page Requests

When opening a standard page there are a number of requests that are made. The example are for the Rings section in the standard Shop.

7.1 Stock level

A query is made to the server to get stock levels when the page is first opened:

```
http://localhost:8080/cgi-
bin/SD2013/st000001.pl?ACTION=GETSECTIONSTOCK&SID=1&timestamp=1350310154118
```

Parameters:

`ACTION=GETSECTIONSTOCK`

Get stock levels for all products in the section.

`SID=1`

For this section

Response:


```
{"11":10,"7":9,"2":100,"17":9,"1":100,"18":10,"16":9,"13":10,"6":10,"9":9,"12":10,"14":9,"15":
10,"8":9,"4":10,"10":9,"19":9,"5":10}
```


This relates product reference to stock level for all products in that page.

Other possibilities include ACTION=GETSTOCK with parameter REF to get stock for a particular product:

```
http://localhost:8080/cgi-bin/SD2013/st000001.pl?ACTION=GETSTOCK&REF=11
```

Response:

```
10
```

7.2 Price

When opening a page with dynamic price enabled, the price of each product based on the logged-in customer is obtained:

```
http://localhost:8080/cgi-
bin/SD2013/dx000001.pl?ACTION=GETALLPRICES&SID=1&v_3_3=on&v_3_1=&v_3_2=&tim
estamp=1350310154126
```

Key parameters are:

`ACTION=GETALLPRICES`

Get all the prices for the section

`SID=1`

For this section.

Response:

This lists the tax amounts and price for each product reference.


```
{"11":{"Tax1":0,"Total":0,"Tax2":0},"7":{"Tax1":0,"Total":0,"Tax2":0},"17":{"Tax1":0,"Total":0
,"Tax2":0},"2":{"Tax1":2600,"Total":13000,"Tax2":0},"1":{"Tax1":400,"Total":2000,"Tax2":0},"18
":{"Tax1":0,"Total":0,"Tax2":0},"16":{"Tax1":0,"Total":0,"Tax2":0},"13":{"Tax1":0,"Total":0,"T
ax2":0},"6":{"Tax1":0,"Total":0,"Tax2":0},"3":{"Tax1":0,"Total":0,"Tax2":0},"9":{"Tax1":0,"Tot
al":0,"Tax2":0},"12":{"Tax1":0,"Total":0,"Tax2":0},"15":{"Tax1":0,"Total":0,"Tax2":0},"14":{"T
```

ax1":0,"Total":0,"Tax2":0},"8":{"Tax1":0,"Total":0,"Tax2":0},"4":{"Tax1":0,"Total":0,"Tax2":0}
,"19":{"Tax1":0,"Total":0,"Tax2":0},"10":{"Tax1":0,"Total":0,"Tax2":0},"5":{"Tax1":0,"Total":0
,"Tax2":0}}

When a change is made, e.g. to select a material and size for the Solitaire Diamond ring, another request is made specifying those options:

```
http://localhost:8080/cgi-
bin/SD2013/dx000001.pl?ACTION=GETACCPRICE&PRODREF=3&SID=1&v_3_3=on&v_3_1=1&
v_3_2=2&Q_3=1&timestamp=1350311076988
```

Parameters:

`ACTION=GETACCPRICE`

Get Account Price including price breaks etc.

`PRODREF=3`

For product 3

`SID=1`

In section 1

`v_3_3=on`

The component is required - the encoding of these values is opaque

`v_3_1=1`

First drop-down has value 1

`v_3_2=2`

Second drop-down has value 2

`Q_3=1`

Quantity for this item is 1.

Response:

{"Tax1":3000,"Total":15000,"Tax2":0}

This gives the price for that option and quantity.