```
<?xml version="1.0" ?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
 contributor license agreements.  See the NOTICE file distributed with
 this work for additional information regarding copyright ownership.
 The ASF licenses this file to You under the Apache License, Version 2.0
 (the "License"); you may not use this file except in compliance with
 the License.  You may obtain a copy of the License at

     http://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
-->

<config>

  <!-- Set this to 'false' if you want solr to continue working after it has
    encountered an severe configuration error.  In a production environment,
    you may want solr to keep working even if one handler is mis-configured.

    You may also set this to false using by setting the system property:
    -Dsolr.abortOnConfigurationError=false
  -->
  <abortOnConfigurationError>${solr.abortOnConfigurationError:true}
  </abortOnConfigurationError>

  <!-- Used to specify an alternate directory to hold all index data
       other than the default ./data under the Solr home.
       If replication is in use, this should match the replication configuration. -->
  <dataDir>c:/vufind/solr/authority</dataDir>

  <indexDefaults>
   <!-- Values here affect all index writers and act as a default unless overridden. -->
    <useCompoundFile>false</useCompoundFile>
    <mergeFactor>10</mergeFactor>
    <!--
      If both ramBufferSizeMB and maxBufferedDocs is set, then Lucene will flush based on
      whichever limit is hit first.

    -->
    <!--<maxBufferedDocs>1000</maxBufferedDocs>-->
    <!-- Tell Lucene when to flush documents to disk.
      Giving Lucene more memory for indexing means faster indexing at the cost of more RAM

      If both ramBufferSizeMB and maxBufferedDocs is set, then Lucene will flush based on
      whichever limit is hit first.

    -->
    <ramBufferSizeMB>32</ramBufferSizeMB>
    <maxMergeDocs>2147483647</maxMergeDocs>
    <writeLockTimeout>1000</writeLockTimeout>
    <commitLockTimeout>10000</commitLockTimeout>
```

```
    <!--
      Expert: Turn on Lucene's auto commit capability.

      TODO: Add recommendations on why you would want to do this.

      NOTE: Despite the name, this value does not have any relation to Solr's autoCommit
      functionality

    -->
    <!--<luceneAutoCommit>false</luceneAutoCommit>-->
    <!--
      Expert:
      The Merge Policy in Lucene controls how merging is handled by Lucene.  The default in
      2.3 is the LogByteSizeMergePolicy, previous
      versions used LogDocMergePolicy.

      LogByteSizeMergePolicy chooses segments to merge based on their size.  The Lucene 2.2
      default, LogDocMergePolicy chose when
      to merge based on number of documents

      Other implementations of MergePolicy must have a no-argument constructor
    -->
    <!--<mergePolicy>org.apache.lucene.index.LogByteSizeMergePolicy</mergePolicy>-->

    <!--
      Expert:
      The Merge Scheduler in Lucene controls how merges are performed.  The
      ConcurrentMergeScheduler (Lucene 2.3 default)
      can perform merges in the background using separate threads.  The SerialMergeScheduler
      (Lucene 2.2 default) does not.
    -->
    <!--<mergeScheduler>org.apache.lucene.index.ConcurrentMergeScheduler</mergeScheduler>-->

    <!--
      As long as Solr is the only process modifying your index, it is
      safe to use Lucene's in process locking mechanism.  But you may
      specify one of the other Lucene LockFactory implementations in
      the event that you have a custom situation.

      none = NoLockFactory (typically only used with read only indexes)
      single = SingleInstanceLockFactory (suggested)
      native = NativeFSLockFactory
      simple = SimpleFSLockFactory

      ('simple' is the default for backwards compatibility with Solr 1.2)
    -->
    <lockType>single</lockType>

    <!-- TODO: check if this are needed -->
    <maxFieldLength>10000</maxFieldLength>

  </indexDefaults>

  <mainIndex>
    <!-- options specific to the main on-disk lucene index -->
    <useCompoundFile>false</useCompoundFile>
    <ramBufferSizeMB>32</ramBufferSizeMB>
```

```
    <mergeFactor>10</mergeFactor>
    <maxMergeDocs>2147483647</maxMergeDocs>
    <maxFieldLength>10000</maxFieldLength>

    <!-- If true, unlock any held write or commit locks on startup.
         This defeats the locking mechanism that allows multiple
         processes to safely access a lucene index, and should be
         used with care. -->
    <unlockOnStartup>false</unlockOnStartup>
  </mainIndex>

  <!-- the default high-performance update handler -->
  <updateHandler class="solr.DirectUpdateHandler2">

    <!-- autocommit pending docs if certain criteria are met -->
    <autoCommit>
      <maxDocs>10000</maxDocs>
      <maxTime>20000</maxTime>
    </autoCommit>

    <!-- The RunExecutableListener executes an external command.
         exe - the name of the executable to run
         dir - dir to use as the current working directory. default="."
         wait - the calling thread waits until the executable returns. default="true"
         args - the arguments to pass to the program.  default=nothing
         env - environment variables to set.  default=nothing
      -->
    <!-- A postCommit event is fired after every commit or optimize command
    <listener event="postCommit" class="solr.RunExecutableListener">
      <str name="exe">snapshooter</str>
      <str name="dir">solr/bin</str>
      <bool name="wait">true</bool>
      <arr name="args"> <str>arg1</str> <str>arg2</str> </arr>
      <arr name="env"> <str>MYVAR=val1</str> </arr>
    </listener>
    -->
    <!-- A postOptimize event is fired only after every optimize command, useful
         in conjunction with index distribution to only distribute optimized indicies
    <listener event="postOptimize" class="solr.RunExecutableListener">
      <str name="exe">snapshooter</str>
      <str name="dir">solr/bin</str>
      <bool name="wait">true</bool>
    </listener>
    -->

  </updateHandler>


  <query>
    <!-- Maximum number of clauses in a boolean query... can affect
        range or prefix queries that expand to big boolean
        queries.  An exception is thrown if exceeded.  -->
    <maxBooleanClauses>1024</maxBooleanClauses>


    <!-- Cache used by SolrIndexSearcher for filters (DocSets),
         unordered sets of *all* documents that match a query.
```

```
       When a new searcher is opened, its caches may be prepopulated
       or "autowarmed" using data from caches in the old searcher.
       autowarmCount is the number of items to prepopulate.  For LRUCache,
       the autowarmed items will be the most recently accessed items.
     Parameters:
       class - the SolrCache implementation (currently only LRUCache)
       size - the maximum number of entries in the cache
       initialSize - the initial capacity (number of entries) of
         the cache.  (seel java.util.HashMap)
       autowarmCount - the number of entries to prepopulate from
         and old cache.
  <filterCache
    class="solr.LRUCache"
    size="512"
    initialSize="512"
    autowarmCount="256"/>
       -->
  <filterCache
    class="solr.LRUCache"
    size="300000"
    initialSize="300000"
    autowarmCount="50000"/>

 <!-- queryResultCache caches results of searches - ordered lists of
       document ids (DocList) based on a query, a sort, and the range
       of documents requested.
  <queryResultCache
    class="solr.LRUCache"
    size="512"
    initialSize="512"
    autowarmCount="256"/>
-->
  <queryResultCache
    class="solr.LRUCache"
    size="100000"
    initialSize="5000"
    autowarmCount="5000"/>

<!-- documentCache caches Lucene Document objects (the stored fields for each document).
     Since Lucene internal document ids are transient, this cache will not be autowarmed.
     -->
  <documentCache
    class="solr.LRUCache"
    size="50000"
    initialSize="50000"
    autowarmCount="10000"/>

  <!-- If true, stored fields that are not requested will be loaded lazily.
  -->
  <enableLazyFieldLoading>false</enableLazyFieldLoading>

  <!-- Example of a generic cache.  These caches may be accessed by name
       through SolrIndexSearcher.getCache(),cacheLookup(), and cacheInsert().
       The purpose is to enable easy caching of user/application level data.
       The regenerator argument should be specified as an implementation
       of solr.search.CacheRegenerator if autowarming is desired.  -->
  <!--
```

```
    <cache name="myUserCache"
      class="solr.LRUCache"
      size="4096"
      initialSize="1024"
      autowarmCount="1024"
      regenerator="org.mycompany.mypackage.MyRegenerator"
      />
    -->

   <!-- An optimization that attempts to use a filter to satisfy a search.
         If the requested sort does not include score, then the filterCache
         will be checked for a filter matching the query. If found, the filter
         will be used as the source of document ids, and then the sort will be
         applied to that.  -->
    <useFilterForSortedQuery>true</useFilterForSortedQuery>

   <!-- An optimization for use with the queryResultCache.  When a search
         is requested, a superset of the requested number of document ids
         are collected.  For example, if a search for a particular query
         requests matching documents 10 through 19, and queryWindowSize is 50,
         then documents 0 through 50 will be collected and cached.  Any further
         requests in that range can be satisfied via the cache.  -->
    <queryResultWindowSize>50</queryResultWindowSize>

    <!-- Maximum number of documents to cache for any entry in the
      queryResultCache. -->
    <queryResultMaxDocsCached>200</queryResultMaxDocsCached>

    <!-- This entry enables an int hash representation for filters (DocSets)
         when the number of items in the set is less than maxSize.  For smaller
         sets, this representation is more memory efficient, more efficient to
         iterate over, and faster to take intersections.  -->
    <HashDocSet maxSize="3000" loadFactor="0.75"/>

    <!-- a newSearcher event is fired whenever a new searcher is being prepared
         and there is a current searcher handling requests (aka registered). -->
    <!-- QuerySenderListener takes an array of NamedList and executes a
         local query request for each NamedList in sequence. -->
    <!--
    <listener event="newSearcher" class="solr.QuerySenderListener">
      <arr name="queries">
        <lst> <str name="q">solr</str> <str name="start">0</str> <str name="rows">10</str>
        </lst>
        <lst> <str name="q">rocks</str> <str name="start">0</str> <str name="rows">10</str>
        </lst>
      </arr>
    </listener>
    -->

    <!-- a firstSearcher event is fired whenever a new searcher is being
         prepared but there is no current registered searcher to handle
         requests or to gain autowarming data from. -->
    <!--
    <listener event="firstSearcher" class="solr.QuerySenderListener">
      <arr name="queries">
        <lst> <str name="q">fast_warm</str> <str name="start">0</str> <str
        name="rows">10</str> </lst>
```

```
      </arr>
    </listener>
    -->

    <!-- If a search request comes in and there is no current registered searcher,
      then immediately register the still warming searcher and use it.  If
      "false" then all requests will block until the first searcher is done
      warming. -->
    <useColdSearcher>false</useColdSearcher>

    <!-- Maximum number of searchers that may be warming in the background
      concurrently.  An error is returned if this limit is exceeded. Recommend
      1-2 for read-only slaves, higher for masters w/o cache warming. -->
    <maxWarmingSearchers>4</maxWarmingSearchers>

  </query>

  <!--
    Let the dispatch filter handler /select?qt=XXX
    handleSelect=true will use consistent error handling for /select and /update
    handleSelect=false will use solr1.1 style error formatting
  -->
  <requestDispatcher handleSelect="true" >
    <!--Make sure your system has some authentication before enabling remote streaming!  -->
    <requestParsers enableRemoteStreaming="false" multipartUploadLimitInKB="2048" />

    <!-- Set HTTP caching related parameters (for proxy caches and clients).

      To get the behaviour of Solr 1.2 (ie: no caching related headers)
      use the never304="true" option and do not specify a value for
      <cacheControl>
    -->
    <!-- <httpCaching never304="true"> -->
    <httpCaching lastModifiedFrom="openTime"
      etagSeed="Solr">
      <!-- lastModFrom="openTime" is the default, the Last-Modified value
        (and validation against If-Modified-Since requests) will all be
        relative to when the current Searcher was opened.
        You can change it to lastModFrom="dirLastMod" if you want the
        value to exactly corrispond to when the physical index was last
        modified.

        etagSeed="..." is an option you can change to force the ETag
        header (and validation against If-None-Match requests) to be
        differnet even if the index has not changed (ie: when making
        significant changes to your config file)

        lastModifiedFrom and etagSeed are both ignored if you use the
        never304="true" option.
      -->
      <!-- If you include a <cacheControl> directive, it will be used to
        generate a Cache-Control header, as well as an Expires header
        if the value contains "max-age="

        By default, no Cache-Control header is generated.

        You can use the <cacheControl> option even if you have set
```

```
       never304="true"
     -->
     <!-- <cacheControl>max-age=30, public</cacheControl> -->
  </httpCaching>
</requestDispatcher>

<!-- requestHandler plugins... incoming queries will be dispatched to the
   correct handler based on the qt (query type) param matching the
   name of registered handlers.
   The "standard" request handler is the default and will be used if qt
   is not specified in the request.
-->
<requestHandler name="standard" class="solr.StandardRequestHandler" default="true">
  <!-- default values for query parameters may optionally be defined here
   <lst name="defaults">
     <int name="rows">10</int>
     <str name="fl">*</str>
     <str name="version">2.1</str>
   <lst>
  -->
  <lst name="defaults">
    <str name="echoParam">explicit</str>
  </lst>
</requestHandler>

<!-- DisMaxRequestHandler is an example of a request handler that
     supports optional parameters which are passed to
     its init() method.
 -->
<requestHandler name="dismax" class="solr.DisMaxRequestHandler" >
  <lst name="defaults">
   <float name="tie">0.01</float>
   <str name="qf">
      text^0.5 features^1.0 name^1.2 sku^1.5 id^10.0 manu^1.1 cat^1.4
   </str>
   <str name="pf">
      text^0.2 features^1.1 name^1.5 manu^1.4 manu_exact^1.9
   </str>
   <str name="bf">
      ord(poplarity)^0.5 recip(rord(price),1,1000,1000)^0.3
   </str>
   <str name="fl">
      id,name,price,score
   </str>
   <str name="mm">
      2&lt;-1 5&lt;-2 6&lt;90%
   </str>
   <int name="ps">100</int>
  </lst>
</requestHandler>
<!-- Note how you can register the same handler multiple times with
     different names (and different init parameters)
  -->
<requestHandler name="partitioned" class="solr.DisMaxRequestHandler" >
  <lst name="defaults">
   <str name="qf">text^0.5 features^1.0 name^1.2 sku^1.5 id^10.0</str>
   <str name="mm">2&lt;-1 5&lt;-2 6&lt;90%</str>
```

```
     <!-- This is an example of using Date Math to specify a constantly
          moving date range in a config...
       -->
     <str name="bq">incubationdate_dt:[* TO NOW/DAY-1MONTH]^2.2</str>
    </lst>
    <!-- In addition to defaults, "appends" params can be specified
         to identify values which should be appended to the list of
         multi-val params from the query (or the existing "defaults").

         In this example, the param "fq=instock:true" will be appended to
         any query time fq params the user may specify, as a mechanism for
         partitioning the index, independent of any user selected filtering
         that may also be desired (perhaps as a result of faceted searching).

         NOTE: there is *absolutely* nothing a client can do to prevent these
         "appends" values from being used, so don't use this mechanism
         unless you are sure you always want it.
      -->
    <lst name="appends">
      <str name="fq">inStock:true</str>
    </lst>
    <!-- "invariants" are a way of letting the Solr maintainer lock down
         the options available to Solr clients.  Any params values
         specified here are used regardless of what values may be specified
         in either the query, the "defaults", or the "appends" params.

         In this example, the facet.field and facet.query params are fixed,
         limiting the facets clients can use.  Faceting is not turned on by
         default - but if the client does specify facet=true in the request,
         these are the only facets they will be able to see counts for;
         regardless of what other facet.field or facet.query params they
         may specify.

         NOTE: there is *absolutely* nothing a client can do to prevent these
         "invariants" values from being used, so don't use this mechanism
         unless you are sure you always want it.
      -->
    <lst name="invariants">
      <str name="facet.field">cat</str>
      <str name="facet.field">manu_exact</str>
      <str name="facet.query">price:[* TO 500]</str>
      <str name="facet.query">price:[500 TO *]</str>
    </lst>
  </requestHandler>

    <requestHandler name="/mlt" class="solr.MoreLikeThisHandler">
      <lst name="defaults">
        <str name="mlt.fl">manu,cat</str>
        <int name="mlt.mindf">1</int>
      </lst>
    </requestHandler>

    <!-- Search component for extracting terms -->
    <searchComponent name="term" class="org.apache.solr.handler.component.TermsComponent">
    </searchComponent>

    <!--
```

```
    Search components are registered to SolrCore and used by Search Handlers

    By default, the following components are avaliable:

    <searchComponent name="query"
    class="org.apache.solr.handler.component.QueryComponent" />
    <searchComponent name="facet"
    class="org.apache.solr.handler.component.FacetComponent" />
    <searchComponent name="mlt"
    class="org.apache.solr.handler.component.MoreLikeThisComponent" />
    <searchComponent name="highlight"
    class="org.apache.solr.handler.component.HighlightComponent" />
    <searchComponent name="debug"
    class="org.apache.solr.handler.component.DebugComponent" />

    If you register a searchComponent to one of the standard names, that will be used
    instead.

  -->

  <requestHandler name="/search" class="org.apache.solr.handler.component.SearchHandler">
    <lst name="defaults">
      <str name="echoParams">explicit</str>
    </lst>
    <!--
      By default, this will register the following components:

      <arr name="components">
      <str>query</str>
      <str>facet</str>
      <str>mlt</str>
      <str>highlight</str>
      <str>debug</str>
      </arr>

      To insert handlers before or after the 'standard' components, use:

      <arr name="first-components">
      <str>first</str>
      </arr>

      <arr name="last-components">
      <str>last</str>
      </arr>

    -->
  </requestHandler>

  <!-- Request handler to extract terms -->
  <requestHandler name="/term" class="org.apache.solr.handler.component.SearchHandler">
    <arr name="components">
      <str>term</str>
    </arr>
  </requestHandler>

  <!-- Update request handler.
```

```
  Note: Since solr1.1 requestHandlers requires a valid content type header if posted in
  the body. For example, curl now requires: -H 'Content-type:text/xml; charset=utf-8'
  The response format differs from solr1.1 formatting and returns a standard error code.

  To enable solr1.1 behavior, remove the /update handler or change its path

  "update.processor.class" is the class name for the UpdateRequestProcessor.  It is
  initalized
  only once.  This can not be changed for each request.
-->
<requestHandler name="/update" class="solr.XmlUpdateRequestHandler" >
  <!--
    <str
    name="update.processor.class">org.apache.solr.handler.UpdateRequestProcessor</str>
  -->
</requestHandler>

<!--
  Analysis request handler.  Since Solr 1.3.  Use to returnhow a document is analyzed.
  Useful
  for debugging and as a token server for other types of applications
-->
<requestHandler name="/analysis" class="solr.AnalysisRequestHandler" >
  <!--
    <str
    name="update.processor.class">org.apache.solr.handler.UpdateRequestProcessor</str>
  -->
</requestHandler>

<!-- CSV update handler, loaded on demand -->
<requestHandler name="/update/csv" class="solr.CSVRequestHandler" startup="lazy" />


<!--
  Admin Handlers - This will register all the standard admin RequestHandlers.  Adding
  this single handler is equivolent to registering:

  <requestHandler name="/admin/luke"
  class="org.apache.solr.handler.admin.LukeRequestHandler" />
  <requestHandler name="/admin/system"
  class="org.apache.solr.handler.admin.SystemInfoHandler" />
  <requestHandler name="/admin/plugins"
  class="org.apache.solr.handler.admin.PluginInfoHandler" />
  <requestHandler name="/admin/threads"
  class="org.apache.solr.handler.admin.ThreadDumpHandler" />
  <requestHandler name="/admin/properties"
  class="org.apache.solr.handler.admin.PropertiesRequestHandler" />
  <requestHandler name="/admin/file"
  class="org.apache.solr.handler.admin.ShowFileRequestHandler" >

  If you wish to hide files under ${solr.home}/conf, explicitly register the
  ShowFileRequestHandler using:
  <requestHandler name="/admin/file"
  class="org.apache.solr.handler.admin.ShowFileRequestHandler" >
  <lst name="invariants">
  <str name="hidden">synonyms.txt</str>
```

```
      <str name="hidden">anotherfile.txt</str>
      </lst>
      </requestHandler>
    -->
    <requestHandler name="/admin/" class="org.apache.solr.handler.admin.AdminHandlers" />

    <!-- ping/healthcheck -->
    <requestHandler name="/admin/ping" class="PingRequestHandler">
      <lst name="defaults">
        <str name="qt">standard</str>
        <str name="q">solrpingquery</str>
        <str name="echoParams">all</str>
      </lst>
    </requestHandler>

    <!-- Echo the request contents back to the client -->
    <requestHandler name="/debug/dump" class="solr.DumpRequestHandler" >
      <lst name="defaults">
        <str name="echoParams">explicit</str> <!-- for all params (including the default
        etc) use: 'all' -->
        <str name="echoHandler">true</str>
      </lst>
    </requestHandler>

    <highlighting>
      <!-- Configure the standard fragmenter -->
      <!-- This could most likely be commented out in the "default" case -->
      <fragmenter name="gap" class="org.apache.solr.highlight.GapFragmenter" default="true">
        <lst name="defaults">
          <int name="hl.fragsize">100</int>
        </lst>
      </fragmenter>

      <!-- A regular-expression-based fragmenter (f.i., for sentence extraction) -->
      <fragmenter name="regex" class="org.apache.solr.highlight.RegexFragmenter">
        <lst name="defaults">
          <!-- slightly smaller fragsizes work better because of slop -->
          <int name="hl.fragsize">70</int>
          <!-- allow 50% slop on fragment sizes -->
          <float name="hl.regex.slop">0.5</float>
          <!-- a basic sentence pattern -->
          <str name="hl.regex.pattern">[-\w ,/\n\"']{20,200}</str>
        </lst>
      </fragmenter>

      <!-- Configure the standard formatter -->
      <formatter name="html" class="org.apache.solr.highlight.HtmlFormatter" default="true">
        <lst name="defaults">
          <str name="hl.simple.pre"><![CDATA[<em>]]></str>
          <str name="hl.simple.post"><![CDATA[</em>]]></str>
        </lst>
      </formatter>
    </highlighting>

  <!-- queryResponseWriter plugins... query responses will be written using the
    writer specified by the 'wt' request parameter matching the name of a registered
    writer.
```

```
    The "standard" writer is the default and will be used if 'wt' is not specified
    in the request. XMLResponseWriter will be used if nothing is specified here.
    The json, python, and ruby writers are also available by default.

    <queryResponseWriter name="standard" class="org.apache.solr.request.XMLResponseWriter"/>
    <queryResponseWriter name="json" class="org.apache.solr.request.JSONResponseWriter"/>
    <queryResponseWriter name="python" class="org.apache.solr.request.PythonResponseWriter"/>
    <queryResponseWriter name="ruby" class="org.apache.solr.request.RubyResponseWriter"/>

    <queryResponseWriter name="custom" class="com.example.MyResponseWriter"/>
  -->
<!--
    XSLT response writer (SOLR-49)
    Changes to XSLT transforms are taken into account every xsltCacheLifetimeSeconds at most.
   -->
  <queryResponseWriter
    name="xslt"
    class="org.apache.solr.request.XSLTResponseWriter"
   >
     <int name="xsltCacheLifetimeSeconds">5</int>
   </queryResponseWriter>

  <!-- config for the admin interface -->
  <admin>
    <defaultQuery>shakespeare</defaultQuery>

    <!-- configure a healthcheck file for servers behind a loadbalancer
    <healthcheck type="file">server-enabled</healthcheck>
    -->
  </admin>

</config>
```