Search

This page gives an overview of the **Search** microservice, including core features, t
Commerce platform.

Overview

Search is vital to delivering quality commerce experiences. It is fundamental for achieving business-drive
conversion rates and cart size. Indexed keyword-based search workflows are often the preferred method
comparison to searches performed by product category. Simply put, if products cannot be found, they ca

The Skava **Search** microservice ("Search") produces high-converting, contextual, type-ahead results for u
items defined in Catalog, within the context of a consumer-facing storefront. Based on the open source A
gives site administrators and third-party systems integrators, the ability to optimize the product search e
leveraged by using the best-in-class techniques that enable search workflows. Some examples are real-ti
synonym-based associations, faceted searching, and weighted results.

Search is coupled with the **Catalog [https://developer.skava.com/microservices/catalog/]** microservice.
performed against indexes, that have previously been harvested from source catalogs, via a queue-basec

Core Features

- Industry-standard search is already integrated with other services
- All setup work is completely out of the box
- Type-ahead, similar items, all the basic search functions you would expect
- Automatic facet normalization (for example, size)
- Search suggestions
- Contextual typeahead (for example, Shirts for Men)
- Boost and Bury search terms
- Synonym mapping
- Filtered search results
- Redirect to non-product search

SOLR Terminology

S.NO	TERM	DESCRIPTION
1	Collection	A complete logical index in a SolrCloud cluster. It is associated with a config set and is made shards. If the number of shards is more than one, it is a distributed index, but SolrCloud lets y collection name and not worry about the shards parameter that is normally required for Distr

S.NO	TERM	DESCRIPTION
2	Config Set	A set of config files necessary for a core to function properly. Each config set has a name. At of solrconfig.xml (SolrConfigXml) and schema.xml (SchemaXml), but depending on the conte may include other files. This is stored in Zookeeper. Config sets can be uploaded or updated command in the command-line utility or the bootsrap_confdir SOLR startup parameter.
3	Core	This is discussed in the General list (below) as SOLR Core. One difference with SolrCloud is t Zookeeper. With traditional SOLR, the core's config will be in the conf directory on the disk. Th a Lucene index along with all the SOLR configuration (SolrConfigXml, SchemaXml, etc.) requi SOLR application can contain 0 or more cores which are run largely in isolation but can comr if necessary via the CoreContainer. From a historical perspective: SOLR initially only supporte SolrCore class was a singleton for coordinating the low-level functionality at the "core" of SO added for creating and managing multiple Cores on the fly, the class was refactored to no lon the name stuck.
4	Leader	The shard replica that has won the leader election. Elections can happen at any time, but nor triggered by events like a SOLR instance going down. When documents are indexed, SolrClou leader of the shard, and the leader will distribute them to all the shard replicas.
5	Replica	One copy of a shard. Each replica exists within SOLR as a core. A collection named "test" cre and replicationFactor set to two will have exactly two replicas, so there will be two cores, eac (or SOLR instance). One will be named test_shard1_replica1 and the other will be named test them will be elected to be the leader.
6	Shard	A logical piece (or slice) of a collection. Each shard is made up of one or more replicas. An e determine which replica is the leader. This term is also in the General list below, but there it re SolrCloud concept of a shard is a logical division. A distributed index is partitioned into "shar corresponds to a SOLR core and contains a disjoint subset of the documents in the index.
7	Auto-warming	What SOLR does when it opens a new cache, and seeds it with key/value pairs based on the instance of the cache.
8	Constraint	A viable method of limiting a set of objects (*).
9	DisMax	Typically a reference to the DisMaxQParserPlugin but in older contexts may be referring to th
10	Facet	A distinct feature or aspect of a set of objects; "a way in which a resource can be classified"
11	Field Collapsing	A specific use case of Result Grouping where the groups are dictated by the value of a field.
12	Filter	Depending on context, may be: Another word for "Constraint" The "fq" param which constrair without influencing the scores. Specifically referring to the Lucene "Filter" class
13	NRT	Near Real Time: This refers to the general concept of wanting document updates to be "imm clients.
14	REquest Handler	A SOLR component that processes requests. For example, the DisMaxRequestHandler proce calling the DisMax Query Parser. Request Handlers can perform other functions, as well.

S.NO	TERM	DESCRIPTION
15	QTime	The elapsed time (in milliseconds) between the arrival of the request (when the SolrQueryRe and the completion of the request handler. It does not include time spent in the response wri the response to the client.
16	Query Parser	A SOLR component that parses the parameters and search terms submitted in a search quer
17	Searcher	In SOLR parlance, the term "Searcher" tends to refer to an instance of the SolrIndexSearcher responsible for executing all searches done against the index, and manages several caches. Searcher per SolrCore at any given time, and that searcher is used to execute all queries agai there may be additional Searchers open at a time during cache warming (in which and "old S requests while a "new Searcher" is being warmed up).
18	Slop	As in "phrase slop": the number of positions two tokens need to be moved in order to match
19	Static warming	What users can do using newSearcher and firstSearcher event listeners to force explicit warr when one of these events happens — frequently it involves seeding one or more caches with queries hard-coded in the `solrconfig.xml`.
20	SOLR Home Dir	Also referred to as the "SOLR Home Directory" or just "SOLR Home" this is the main directory configuration files, data, and plugins. Knowing which directory to use as the SOLR Home is th information that SOLR must either assume (the default is "./solr") or be configured using sor SOLR's normal configuration files. An example SOLR Home is included in SOLR releases and explaining the directory structure.

SOLR Scheme

```
<schema name="product" version="1.5">
    <fields>
        <!-- Mandatory field required for SOLR validation -->
        <field name="_version_" type="long" indexed="true" stored="true"/>

        <!-- search fields -->
        <field name="collectionid" type="long" indexed="true" stored="false"/>
        <field name="id" type="string" indexed="true" stored="true" required="true" multiValu
        <dynamicField name="index_number_*" type="float" indexed="true" stored="false" multiV
        <dynamicField name="index_string_*" type="text" indexed="true" stored="false" multiVa
        <field name="groupid" type="string" indexed="true" stored="false" multiValued="false'
        <field name="locale" type="string" indexed="true" stored="false" multiValued="true"/
        <dynamicField name="index_key_*" type="string" indexed="true" stored="false" multiVa
        <dynamicField name="suggestion_*" type="string" indexed="true" stored="true" multiVa
        <dynamicField name="key_*" type="string" indexed="true" stored="false" multiValued="t
```

```
            <dynamicField name="facet_*" type="string" indexed="true" stored="false" multiValued=
            <dynamicField name="range_facet_*" type="float" indexed="true" stored="false" multiVa
            <dynamicField name="sort_number_*" type="float" indexed="true" stored="false"/>
            <dynamicField name="sort_string_*" type="string" indexed="true" stored="false"/>
            <field name="starttime" type="timestamp" indexed="true" stored="true"/>
            <field name="historicalsale" type="long" indexed="true" stored="false"/>
            <field name="endtime" type="timestamp" indexed="true" stored="true"/>
            <field name="suggestion" type="text" indexed="true"  stored="true" required="false" r
            <field name="response" type="string" indexed="false" stored="true"/>
        </fields>
        <uniqueKey>id</uniqueKey>
        <types>
            <!-- common search fields types -->
            <fieldType name="long" class="solr.TrieLongField" precisionStep="0" positionIncrement
            <fieldType name="timestamp" class="solr.TrieDateField" precisionStep="0" positionIncr
            <fieldType name="float" class="solr.TrieFloatField" precisionStep="0" positionIncreme
            <fieldType name="string" class="solr.StrField" sortMissingLast="true" />
            <fieldType name="text" class="solr.TextField" positionIncrementGap="100">
                <analyzer type="index">
                <tokenizer class="solr.WhitespaceTokenizerFactory"/>
                <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNur
                        catenateWords="1" catenateNumbers="1" catenateAll="0" splitOnCaseChange="
                <filter class="solr.ASCIIFoldingFilterFactory"/>
                <filter class="solr.LowerCaseFilterFactory"/>
                <filter class="solr.ShingleFilterFactory" outputUnigramsIfNoShingles="true" maxSh
                        outputUnigrams="true"/>
            </analyzer>
            <analyzer type="query">
                <tokenizer class="solr.WhitespaceTokenizerFactory"/>
                <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNur
                        catenateWords="1" catenateNumbers="1" catenateAll="0" splitOnCaseChange="
                <filter class="solr.ASCIIFoldingFilterFactory"/>
                <filter class="solr.LowerCaseFilterFactory"/>
                <filter class="solr.ShingleFilterFactory" outputUnigramsIfNoShingles="true" maxSh
                        outputUnigrams="true"/>
            </analyzer>
            </fieldType>
        </types>
    </schema>
```

Revision History
2020-09-28 **|** AN – Updated the Core Features section.
2019-07-12 **|** AM – Added Core Features section for July 2019 release.
2019-01-23 **|** PLK – Page created and content uploaded.