**User Information Search Design Approach**

1. Table of Contents

[1. Table of Contents 2](#_Toc460322001)

[2. Definitions, Acronyms, and Abbreviations 4](#_Toc460322002)

[3. Introduction 5](#_Toc460322003)

[4. Purpose 5](#_Toc460322004)

[5. Problem Statement 5](#_Toc460322005)

[6. Possible Approaches 5](#_Toc460322006)

[6.1. About Apache LuceneTM and SolrTM 5](#_Toc460322007)

[7. Entities Search Requirements 6](#_Toc460322008)

[8. Entities Search Implementation 6](#_Toc460322009)

[8.1. SOLR configuration - solrconfig.xml 8](#_Toc460322010)

[9.1.1 UpdateHandler and Autocommit 8](#_Toc460322011)

[9.1.2 RequestHandler 9](#_Toc460322012)

[9.1.3 Search Component 9](#_Toc460322013)

[8.2. Data Import Handler configuration - db-dataconfig.xml and solrconfig.xml 10](#_Toc460322014)

[9.1.4 Full-Import configuration 11](#_Toc460322015)

[9.1.5 Delta-Import configuration 11](#_Toc460322016)

[8.3. Entities Search Fields configuration-schema.xml 13](#_Toc460322017)

[9.3.1 Phonetic Search 13](#_Toc460322018)

[9.4 Entities Search Rules configuration 15](#_Toc460322019)

[9.4.1 DPE Role Exclusion 15](#_Toc460322020)

[9.4.2 Entity Field Weighting 15](#_Toc460322021)

[9.4.2.1 Person Entity with IDN Search 17](#_Toc460322022)

[9.4.2.2 Person Entity without IDN Search 18](#_Toc460322023)

[9.4.2.3 Non-Person Entity Search 19](#_Toc460322024)

[9.4.3 Search Result Omission 20](#_Toc460322025)

[9. Solr Search Client 21](#_Toc460322026)

[10.1 Solr Search Query format 21](#_Toc460322027)

[10.2 Query String formation using SolrJ 22](#_Toc460322028)

[11 Single and Multi-tenant Support 23](#_Toc460322029)

[11.1 Multi-tenant support 23](#_Toc460322030)

[11.2 Single tenant (No Multi-tenant) support 23](#_Toc460322031)

[12 Examples 24](#_Toc460322032)

[13 References 25](#_Toc460322033)

[14 Approval 26](#_Toc460322034)

1. Definitions, Acronyms, and Abbreviations

| **Term/Acronym/Abbreviation** | **Description** |
| --- | --- |
| Entity / Entities | A user can be a person or non-person (for example: company or business). So, we use a generic term – Entity / Entities |
| SQL | Structured Query Language |
| XML | Extensible Mark-up Language |
| HTTP | HyperText Transport Protocol |
| REST | Representational state transfer |
| L10N | Localization |
| Solr | Apache LuceneTM project based SolrTM Web server |
| JDBC | Java Database Connectivity |
| CRUD | Create Update Delete |
| PRS | Product Requirement Specification |
| CDS | Component Design Specification |
| POC | Proof of Concept |
| DIH | Data Import Handler |

1. Introduction

User Information system provides Person, Non-person specific data to other systems. We call them “Entities”. There is a need to improve the way in which entities and their information is identified, retrieved and updated. This document introduces a faster and better alternative to search for Entities information from its database.

1. Purpose

The purpose of this document is to describe the possible approach to address the problem of searching and retrieving entities information faster and better than any of the existing services. The Entities Search application is being designed for providing high performance search operation while maintaining flexibility in search input and acceptable search result quality.

1. Problem Statement

The existing service to search for entity information does not work for “Wildcard”, “Phonetic” and N-Gram type of search queries. It only supports “Exact” and “Like” search. It is not flexible enough to search from multiple data sources like database, rich text documents etc. Also, on performance side, query performance is slow. It is not possible to index every column of the tables in the database for search purpose owing to performance reasons.

1. Possible Approaches

Two possible approaches are discussed in detail in the POC documents that are attached in the References section. POC has been carried out and as per recommendation from Discovery, SolrTM approach will be used. Following extract provides a brief introduction to the proposed approach.

* 1. About Apache LuceneTM and SolrTM

Full textual search has been considered since 1997 Google paper which is referred to in the References section. An inverted index approach along with page-ranking was proposed. This is also the basic idea behind the Apache LuceneTM project viz. to consider indexing and querying operations as separate activities and optimise each one of them independently.

Apache LuceneTM is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

SolrTM is the popular, blazing fast open source enterprise search platform from the Apache LuceneTM project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document (example, Word, PDF) handling, and geospatial search. SolrTM is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. SolrTM powers the search and navigation features of many of the world's largest internet sites.

SolrTM is written in Java and runs as a standalone full-text search server within a servlet container such as Jetty. SolrTM uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Solr's powerful external configuration allows it to be tailored to almost any type of application without Java coding, and it has extensive plugin architecture when more advanced customization is required.

For more detailed feature list, kindly refer to the links in the References section.

The fundamental premise of SolrTM is simple. Feed in it a lot of information, then later ask questions and find the piece of information as required. Feeding in information is called **indexing** or **updating**. Asking a question is called a **querying** or **searching**.

SolrTM performs following activities:

1. **Indexing**

SolrTM indexes the data present in the available configured data sources [**dataconfig.xml**]. As per [**schema.xml**] configuration, it uses filters for wildcard, lowercase, phonetic match etc. and creates indexes accordingly.  
This indexing activity can be scheduled to take care of updates in the database after CRUD operations. Replication, sharding of the indexes and search quality flexibility (faceting, relative fields’ boosters, spellcheck etc.) can be configured in [**solrconfig.xml**]

1. **Querying**

On receiving search input criteria, Solr searches the indexes and provides search results as per configuration.

1. Entities Search Requirements

This approach document consists of implementation approach for below requirements:

* **Wildcard search:** (For example search input “sin\*” should match “sinha”
* **Like search**: (For example search input “ashle” should match “ashlee” )
* **Lowercase search:** (For example search input “ashlee” should match “ASHLEE”)
* **Phonetic search:** (For example search input “cinna” should match “SINHA”)
* **Exact search:** (For example search input “Sachin” should match “Sachin” )

**Business Rules Implementation:**

* **DPE Role Exclusion:** Dummy Person Entity roles should be excluded from search results.
* **Entity Field Weighting:** (Details presented in section under Entities Field Weighting )
* **Search Result Omission:** (Details presented in section under Search Result Omission )
* **Multi-Tenancy Support:**

1. Entities Search Implementation

The requirements will be fulfilled by proper configuration of the Solr. Solr allows XML based configuration for features injection for both **indexing** and **querying** activities. It is a one-time activity to be performed before indexing. There are mainly 3 configuration files required by Solr viz.

1. **solrconfig.xml**

Solr configuration for search query handling, replication, sharding, cores, commit speed, external libraries etc. It also has a pointer to **db-data-config.xml.**

1. **db-data-config.xml**

Data Import Handler settings for importing data into Solr.

1. **schema.xml**

Schema Configuration for mapping source data field types to Solr field types, filtering etc.

Sample Solr configuration is attached as below with all the configuration options. The different sections from this configuration are referred in the subsequent document. Extract the contents of the below archive to find the configuration files.



Figure 1: Sample Solr Configuration files

1. **solrconfig.xml**

The sample structure of this file is as below-

|  |
| --- |
| <config>  <!—- one or more occurrences of any of below -->  <requestHandler>  <lst name="" ... >  <str ... />  </lst>  </requestHandler> </config> |

1. **db-dataconfig.xml**Sample structure:

|  |
| --- |
| <dataConfig>  <dataSource/>  <document>  <entity ... query=”” deltaImportQuery="" deltaQuery="" ... >  <field column=”” name=””/>  <field column=”” name=””/>  </entity>  </document> |

The sample changes in the configuration in **db-dataconfig.xml** with respect to Entities Search are as below:

1. **schema.xml**The sample structure of this file is as below:

|  |
| --- |
| <schema>  <!—- zero or more occurrences of any of below -->  <fieldType>  <analyser>  <tokenizer ... />  <filter ... />  <filter ... />  </analyser>  </fieldType>  <field name="" type="" ... />  <field name="" type="" ... />  <uniqueKey/>  <copyField source="" dest="" />  </schema> |

The sample changes in the configuration in this file **schema.xml** with respect to Entities Search are as below:

**In brief, the one-time configuration steps for Entities Search are as follows:**

1. The configuration for interacting with Solr that will be used during **Querying** activity is configured in **solrconfig.xml** configuration file. Below section ***9.1*** ***SOLR configuration*** explains detailed configurations required for entities search system.
2. The **Data Import Handler** is configured to map the database tables’ columns and columns types to the corresponding Solr *field names* that are used to index Solr documents.
3. The above *field names* are configured to the *field types* provided by Solr in the **Schema Configuration**.
4. The *filters* provided by Solr are configured for the *field types* focussing on meeting each of the Entities Search requirements.

**To implement each of the above Entities Search requirements, configuration of different sections of the above configuration files is required.** **The changes in the configuration files will be explained in detail focussing upon how each of the Entities Search requirements can be met.**

* 1. SOLR configuration - solrconfig.xml

**solrconfig.xml** comprises of XML statements that set the configuration value of Solr instance. It is the first file that is read by Solr. If any external libraries are being used, it is mentioned in this file. Here is a brief description of some of the XML tags used in **solrconfig.xml**:

* + 1. UpdateHandler and Autocommit

The <updatehandler> section affects how updates are done internally. The <autoCommit> sub-element contains further configuration for controlling how pending updates will be automatically push to the index. <maxTime> is the number of milliseconds since the oldest uncommitted update. <openSearcher> - if false, the commit causes recent index changes to be flushed to stable storage, but does not cause a new searcher to be opened to make those changes visible.  
With respect to Entities Search, hard commit and soft commit is set as 15 seconds and 1 seconds respectively.

|  |
| --- |
| <autoCommit>   <maxTime>**${solr.autoCommit.maxTime:15000}**</maxTime>   <openSearcher>**false**</openSearcher> </autoCommit> |
| <!-- softAutoCommit is like autoCommit except it causes a 'soft' commit which only ensures that changes are visible but does not ensure that data is synced to disk. This is faster and more near-realtime friendly than a hard commit.--> |
| <autoSoftCommit>  <maxTime>**${solr.autoSoftCommit.maxTime:1000}**</maxTime> </autoSoftCommit> |

* + 1. RequestHandler

The <requestHandler> defines the logic executed for any request. Multiple instances of request handlers, each with different names and configuration will be declared. The **qt** url parameter or the path of the url is used to select the request handler by name. The default request handler is **SearchHandler**. This will be used for all FR1 requirements without any changes.

|  |
| --- |
| <requestHandler name="**/select**" class="**solr.SearchHandler**"> </requestHandler> |

* + 1. Search Component

Instances of **SearchComponent** define discrete units of logic that can be combined together and reused by Request Handlers (in particular, **SearchHandler**) that know about them. The default **SearchComponent** used by **SearchHandler** is query, facet, mlt (More Like This), highlight, stats, debug. For Entities Search requirement, **query** and **mlt** will be used.   
By default, the following components are available.

|  |
| --- |
| <searchComponent name="**query**" class="**solr.QueryComponent**" /> |
| <searchComponent name="**facet**" class="**solr.FacetComponent**" /> |
| <searchComponent name="**mlt**" class="**solr.MoreLikeThisComponent**" /> |
| <searchComponent name="**highlight**" class="**solr.HighlightComponent**" /> |
| <searchComponent name="**stats**" class="**solr.StatsComponent**" /> |
| <searchComponent name="**debug**" class="**solr.DebugComponent**" /> |

* 1. Data Import Handler configuration - db-dataconfig.xml and solrconfig.xml

As entities information is stored in entities core databases, Data Import Handler (DIH) is used to import data from database into Solr index. It exposes its API as HTTP request. The Data Import Handler is registered in **solrconfig.xml** that has a pointer to the **db-dataconfig.xml** file.

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **File name** | **Content snippet in the relevant section** |
| 1 | **solrconfig.xml** | <requestHandler name=**"/dataimport"** class=**"org.apache.solr.handler.dataimport.DataImportHandler"**> <lst name=**"defaults**"> <str name="**config**">**db-data-config.xml**</str> </lst> </requestHandler> |
| 2 | **db-dataconfig.xml structure** | <dataConfig>  <dataSource/>  <document>  <entity ... query=""   deltaImportQuery=""   deltaQuery="" ... >  <field column="" name=""/>  <field column="" name=""/>  </entity>  </document> </dataConfig> |

A sample **db-dataconfig.xml** is provided in the References section under “**Sample Solr configuration**”.

The <dataSource/> tag contains the data source as follows:

|  |
| --- |
| <dataSource name="**jdbc**" driver="**oracle.jdbc.driver.OracleDriver**" url="**jdbc:oracle:thin:@10.42.4.193:1521:DHDB**" user="**MT\_I18N**" password="**MT\_I18N**" /> |

It contains a single <document> tag representing a single Solr document. Within this document, there are multiple hierarchical <entity> tags depending upon the number of tables and their relationships. In Entities database, ENTITIES table is the parent table containing the primary key ENTITY\_NO. This key is used as the foreign key in ONE-TO-MANY relationships with other child tables. Accordingly, the relationship between tables in the database is reflected in the <entity> tags hierarchy maintained in the Solr indexed document.

The <field> tags map the database column name with the name that Solr will use to index it in **schema.xml**. The name="" is the name that will be referenced to in **schema.xml**.

Column name in database

Reference used in **schema.xml**

|  |
| --- |
| <field column="**ENTITY\_NO**" name="**entity\_no**"/> <field column="**SURNAME**" name="**surname**"/> |

The following operations are possible in DIH for Entities Search application:

* **Scheduler:** Scheduleris used for scheduling the **delta-imports** periodically.
  + 1. Full-Import configuration

Full-Import is used for full-import of existing data from database. Full-import command uses query=”” attribute**.** Following configuration is done in the relevant <entity query="" > </entity> sections in DIH file **db-dataconfig.xml**.  
  
**A sample db-dataconfig.xml snippet:**

|  |
| --- |
| <dataConfig>  <dataSource ... />  <document>  <entity name="**ENTITIES**" pk="**entity\_no**"  query="**SELECT ENTITY\_NO,FULL\_NAME,FIRST\_NAME,TENANT\_ID FROM ENTITIES**"> |
| <field column=”**ENTITY\_NO**” name=”**entity\_no**”/>  <field column=”**SURNAME**” name=”**surname**”/> |
| </entity>  </document>  </dataConfig> |

The SELECT query contains the fields that are to be indexed.

The Solr server is invoked with below URL for sending **Full-Import HTTP request:**  
[http://localhost:8080/solr-4.9.0/db-core/dataimport?command=](http://localhost:8080/solr-4.9.0/db-core/dataimport?command=full-import)**[full-import](http://localhost:8080/solr-4.9.0/db-core/dataimport?command=full-import)**

* + 1. Delta-Import configuration

The **DataImportHandler (DIH)** class exposes a variable called *last\_index\_time* which is a timestamp value denoting the last time full-import 'or' delta-import timestamp. This variable is used in the SQL queries in **db-data-config.xml** and its value will be replaced by the latest runtime value during indexing. Delta-Import is used for import of newly added/deleted/updated rows since the last index time. Delta-import command uses deltaImportQuery=”” and deltaQuery=”” attribute.

**A sample db-dataconfig.xml snippet:**

|  |
| --- |
| <dataConfig>  <dataSource ... />  <document> |
| <entity name="**entity**" pk="**ENTITY\_NO**"  deltaImportQuery="**SELECT ENTITY\_NO, FULL\_NAME,..., TENANT\_ID FROM ENTITIES WHERE ENTITY\_NO = ‘${dih.delta.entityno}’**"  deltaQuery="**SELECT ENTITY\_NO FROM ENTITIES WHERE DATE\_TIME\_LAST\_MODIFIED &gt; '${dih.last\_index\_time}'**" > |
| </entity>  </document>  </dataConfig> |

The Solr server is invoked with below URL for sending **Delta-Import HTTP request:**

[http://localhost:8080/solr-4.9.0/db-core/dataimport?command=**delta-import**](http://localhost:8080/solr-4.9.0/db-core/dataimport?command=delta-import)

**Scheduler configuration for Delta-Import:**

The schedule will be set to invoke the delta-import request to automatically update the index hourly or daily or weekly based upon the frequency of CRUD operations performed on the database tables. We propose to use **cron** or any other existing scheduler. A configuration file called **dataimport.properties** contains the HTTP URL for delta-import. The cron jobs will contain the **delta-import** command HTTP URLs. Some other fields like *last\_index\_time* are automatically written by Solr into this file.   
A sample **dataimport.properties** is attached in the References Section under “**Sample Solr Configuration**”.

* 1. Entities Search Fields configuration-schema.xml

Schema configuration describes how the indexes are to be created from the given input data.To build a searchable index, Solr takes in documents composed of data fields of specific field types. The **schema.xml** configuration file defines the *field types* and specific *fields* that the documents will contain, as well as how Solr should handle those fields when adding documents to the index or when querying for those fields. For some requirements, **db-data-config.xml** will also require certain configuration**.**

* + 1. Phonetic Search

Consider that there is a column “**SURNAME**” in the database table of VARCHAR2 type on which a phonetic query is to be applied (“cinna” should match “SINHA” or “sinha” and vice versa). It is imported through data-import settings in **data-config.xml** described in the DIH section as “**surname**” and is referred to by the same name in **schema.xml:**

|  |  |
| --- | --- |
| **File name** | **Content snippet in the relevant section** |
| **db-dataconfig.xml structure** | <dataConfig>  <dataSource/>  <document>  <entity ... query="" deltaImportQuery="" deltaQuery="">  <field column="**SURNAME**"" name=""**surname**""/>  <field column="" name=""/>  </entity>  </document> </dataConfig> |
| **schema.xml** | <field name="**surname**" type="**phonetic**" indexed="**true**" stored="**true**" multiValued="**false**" required="**false**" />  <fieldType name="**phonetic**" class="**solr.TextField**" positionIncrementGap="**100**" stored="**false**" multiValued="**true**">  <analyzer>    <tokenizer class="**solr.StandardTokenizerFactory**" />    <filter class="**solr.PhoneticFilterFactory**" encoder="**DoubleMetaphone**" inject="**true**" />    <filter class="**solr.LowerCaseFilterFactory**" />  </analyzer>  </fieldType> |

**Explanation:**

1. <fieldType> - tag is used to configure the custom types in Solr. Here, say for configuring all fields to be enabled for phonetic search, a <fieldType name="**phonetic**" ... /> is defined.

1<tokenizer> - Tokenizers that can be used in Entities Search are:

**solr.StandardTokenizerFactory:** Fields containing alpha-numeric characters with special characters like full-stop (.), comma (,), hyphen (-), slashes (/ \) are tokenized properly using this tokenizer. It is used for EXACT, LIKE and PHONETIC search.

2<filter> - Below are the different filters which could be configured for Entities Search

**2.1 PhoneticFilterFactory –** For Fields with phonetic Search.

**2.2. ReversedWildcardFilterFactory –** Fields with wild card search.

**2.3. LowerCaseFilterFactory -** Fields withlowercase search.

3<copyfield> **-** It is possible to map one field with multiple field types using this element.

Below is example configuration for surname field as Phonetic search with lowercase.

|  |  |
| --- | --- |
| **File name** | **Content snippet in the relevant section** |
| **schema.xml** | <field name="**surname**" type="**phonetic**" indexed="**true**" stored="**true**" multiValued="**false**" required="**false**" />  <fieldType name="**phonetic**" class="**solr.TextField**" positionIncrementGap="**100**" stored="**false**" multiValued="**true**">  <analyzer>    <tokenizer class="**solr.StandardTokenizerFactory**" />    <filter class="**solr.PhoneticFilterFactory**" encoder="**DoubleMetaphone**" inject="**true**" />    <filter class="**solr.LowerCaseFilterFactory**" />  </analyzer>  </fieldType> |

* 1. Entities Search Rules configuration

Below are the different rules and the configurations for Entities Search application.

* + 1. DPE Role Exclusion

All entities with the DPE (Dummy Person Entity) role are to be excluded from search results.

While full import first time the entities would be selected using below query.

Those entities with ‘DPE’ as one of the roles will be excluded from indexing itself. This can be implemented by configuring **db-data-config.xml** as below.

|  |
| --- |
| <dataConfig>  <dataSource/>  <document>  <entity ... query="SELECT ENTITY\_NO ENTITY\_ROLE FROM ENTITY\_ROLE\_USAGES WHERE ENTITY\_NO='${ENTITIES.ENTITY\_NO}' **AND ENTITY\_ROLE !='DPE'**"  />  </document>  </dataSource>  </dataConfig> |

* + 1. Entity Field Weighting

For implementing Weighting

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | | |
| **Field** | **Search Type Capability\*\*** | **Mandatory/Optional** | **Value Type** |
| **Biographical Information** | | | |
| **Person Entity** | | | |
| Person/ Non Person Indicator | Exact | Mandatory | VARCHAR2(1) |
| Legal Reference Number | Exact, Like | Mandatory | VARCHAR2(120) |
| Legal Reference Number Type | Exact | Mandatory | VARCHAR2(3) |
| **Non-Person Entity** | | | |
| Person/ Non Person Indicator | Exact | Mandatory | VARCHAR2(1) |
| Legal Reference Number | Exact, Like ‘Removal of ‘/’’ | Mandatory | VARCHAR2(120) |
| Legal Reference Number Type | Exact | Mandatory | VARCHAR2(3) |

The following table shows the Input Request parameters based upon which the segregation of request handlers are set in **solrconfig.xml**:

|  |  |  |
| --- | --- | --- |
| **Input** | | |
| **Person with IDN** | **Person without IDN** | **Non-Person** |
| **Solr input query** | | |
| person\_non\_person:P  AND  no\_type:IDN OR no\_type:PS | person\_non\_person:P AND NOT no\_type:IDN OR no\_type:PS | person\_non\_person:N |
| **solrconfig.xml configuration** | | |
| <requestHandler name="**/searchPersonWithIDN**" | <requestHandler name="**/searchPersonWithoutIDN**" | <requestHandler name="**/searchNonPerson**" |

* + - 1. Person Entity with IDN Search

Request handler will be set in Solr configuration (**solrconfig.xml**). As we know from the mandatory request parameter person\_non\_person field that the request has been initiated for Person entity if its value is **P**. Further, from the mandatory no\_type field with value **IDN or PS**, we shall direct the Solr to search for “**Person with IDN**” according to the following configuration:

<requestHandler name="**/searchPersonWithIDN**" class="**solr.SearchHandler**">  
<!-- Query settings -->  
<str name="defType">edismax</str>  
<str name="qf">

firstname\_exact^300 firstname\_like^30 firstname\_phonetic^3   
nickname\_exact^100 nickname\_like^10 nickname\_phonetic^1  
surname\_exact^100 surname\_like^10 surname\_phonetic^1   
gender^300   
date\_of\_birth^300   
email\_addr\_exact^200 email\_addr\_like^20   
contact\_no^200

</str>  
<str name="mm">50%</str>  
</requestHandler>

The configuration contains the boost factors for individual fields relative to each other. Higher the boost factor, higher is the priority given to that field for matching purpose. For example firstname\_exact field has boost factor of 300 while nickname\_exact has the boost factor 100. This means that in case of “EXACT” match for firstname and nickname, the record with match on firstname will be ranked higher than that with the match on nickname. Also note that the EXACT/LIKE match ratio and LIKE/PHONETIC match ratio is maintained as 10 which is arbitrary and can be changed.

A sample FR4 operation for a “**Person with IDN**” query may result in the following query search string:   
firstname:Sachin AND surname:Sinha date\_of\_birth:24-03-1983T00:00:00Z AND gender:M AND title:MR. AND person\_non\_person: P AND other\_no:2001185721012 AND no\_type: IDN OR no\_type:PS AND email\_addr:sachins@zensar.in AND email\_type: Business

A sample HTTP request will contain the request handler for which the query is being performed: [http://localhost:8080/solr-4.9.0/#/db-core**/searchPersonWithIDN**?q=person\_non\_person%3AP+AND+surname%3Asinha+AND+firstname%3Asachin&rows=50&indent=on&qt=edismax](http://localhost:8080/solr-4.9.0/#/db-core/searchPersonWithIDN?q=person_non_person%3AP+AND+surname%3Asinha+AND+firstname%3Asachin&rows=50&indent=on&qt=edismax)

* + - 1. Person Entity without IDN Search

For Person without IDN will have the following configuration in **solrconfig.xml**:

<requestHandler name="**/searchPersonWithoutIDN**"class="solr.SearchHandler">  
<!-- Query settings -->  
<str name="defType">edismax</str>  
<str name="qf">

firstname\_exact^300 firstname\_like^30 firstname\_phonetic^3   
nickname\_exact^100 nickname\_like^10 nickname\_phonetic^1  
surname\_exact^200 surname\_like^20 surname\_phonetic^2   
gender^300   
date\_of\_birth^300   
email\_addr\_exact^200 email\_addr\_like^20   
contact\_no^200

</str>  
<str name="mm">50%</str>  
</requestHandler>

The change in the boost factor for surname\_exact is now 200.0 according the Entity Field Weighting table given in the PRS (FR4).   
An example FR4 operation for a **“Person without IDN”** query may result in the following query search string:  
firstname:Sachin AND surname:SINha date\_of\_birth:24-03-1983T00:00:00Z AND gender:M AND title:MR. AND person\_non\_person:P AND other\_no:2001185721012 AND (NOT no\_type:IDN) | (NOT no\_type:PS) AND email\_addr:sachins@zensar.in AND email\_type:Business

The change from the former in the query string pattern above is:   
AND (NOT no\_type:IDN) | (NOT no\_type:PS)

The HTTP request will contain the request handler for which the query is being performed: [http://localhost:8080/solr-4.9.0/#/db-core**/searchPersonWithoutIDN**?q=person\_non\_person%3A**P**+AND+surname%3Asinha+AND+firstname%3Asachin&rows=50&indent=on&qt=edismax](http://localhost:8080/solr-4.9.0/#/db-core/searchPersonWithoutIDN?q=person_non_person%3AP+AND+surname%3Asinha+AND+firstname%3Asachin&rows=50&indent=on&qt=edismax)

* + - 1. Non-Person Entity Search

We shall set the request handler with the following configuration in **solrconfig.xml**:

<requestHandler name="**/searchNonPerson**" class="**solr.SearchHandler**">  
<!-- Query settings -->  
<str name="defType">edismax</str>  
<str name="qf">

fullname\_exact^300 fullname\_like^30 fullname\_phonetic^3   
other\_no^300 other\_no\_like^30   
email\_addr\_exact^200 email\_addr\_like^20   
contact\_no^200  
</str>  
<str name="mm">50%</str>  
</requestHandler>

The boost factor for fullname\_exact field is the highest as specified in the PRS.

A sample FR4 operation for a “**Person with IDN**” query may result in the following query search string:  
fullname:Zensar Technologies Ltd. AND person\_non\_person:N AND email\_addr:contact@zensar.in AND email\_type:Business

The HTTP request will contain the request handler for which the query is being performed: [http://localhost:8080/solr-4.9.0/#/db-core**/searchNonPerson**?q=person\_non\_person%3A**N**+AND+fullname%3Azensar%20technologies+AND+email\_addr%3Acontact@zensar.in&rows=50&indent=on&qt=edismax](http://localhost:8080/solr-4.9.0/#/db-core/searchNonPerson?q=person_non_person%3AN+AND+fullname%3Azensar%20technologies+AND+email_addr%3Acontact@zensar.in&rows=50&indent=on&qt=edismax)

* + 1. Search Result Omission

Within the request handler tag in **solrconfig.xml**, there is a tag for minimum number of matches called **mm** whose value is given as a percentage. It denotes the ratio of the number of fields in the records that “should” match with the number of fields present in the request parameters in order to show up in the search results. For example:

<str name="mm">60%</str>

The above configuration means that at least 60% of the fields given in the request should match.  
Apart from the above, all our search string request parameters will AND each other by default.

For example: fullname:”Zensar Technologies Ltd.” AND person\_non\_person:N  
If only fullname matches out of the two fields above, then those records will not be seen as the threshold **mm** is 60%

1. Solr Search Client

The Entities Search application will require a client service to send RESTful HTTP requests to the External Solr Search system. Its scope is described in the following activity diagram. The activity is the same for both FR1 and FR2 with respect to Solr Search Activity.



Figure 2: Solr Search Activity Diagram

The Web service request message will contain an EntityDetail object [Refer to CDS EntityDetail Model class diagram for further details]. A Solr search query string from this EntityDetail object after a series of transformations.

EntityDetail Object

Transformations: Validation for non-null fields, Appending to input list, and then converting this input List into string

Final Solr query string

| **Sample input list examples** | **Corresponding Search string** |
| --- | --- |
| [FirstName:Sa\*in] | “firstname:Sa\*in” |
| [FirstName:sachin, SurName:SINHA] | “firstname:sachin AND surname:SINHA” |
| [FirstName:sachin, SurName:SINHA, nickname:sach\*] | “firstname:Sachin AND surname:SINHA AND nickname:sach\*” |
| [FirstName:sachin, SurName:SINHA, nickname:sach\*, DateOfBirth : 01/10/1975] | “firstname:sachin AND surname:SINHA AND nickname:sach\* AND date\_of\_birth:01/10/1975T00:00:00Z” |

The Solr search query format is explained below:

* 1. Solr Search Query format

Solr exposes a RESTful web service. For example, to search for firstname “sachin” and surname “sinha”, the following query is required to be formed:

<http://localhost:8080/solr-4.9.0/#/db-core/select/?q=surname%3Asinha+AND+firstname%3Asachin&rows=50&indent=on>

Explanation:

|  |  |
| --- | --- |
| 10.41.50.19:7001 | ip\_address : port |
| solr-4.9.0 | Solr webapp context |
| db-core | Solr Core name (since Solr 4.0 onwards ) |
| q=surname%3Asinha+AND+firstname%3Asachin | query parameters for the fields **firstname** and **surname** as configured in the **schema.xml** and **db-dataconfig.xml** |

* 1. Query String formation using SolrJ

The query string is formed in the above format using **SolrJ** java library which simplifies this task. A sample SolrJ code snippet will explain the format:

HttpSolrServer solr = **new** HttpSolrServer("http://10.41.50.19:7002/solr-4.9.0/db-core/"); // running solr instance

SolrQuery query = **new** SolrQuery(); // create a query object

query.setQuery("firstname:sachin AND surname:sinha"); // using StringBuilder form this string from Input List

query.setFields("entity\_no"); // set the result fields

query.set("defType", "edismax"); // enable Extended Dismax query parser to take care of special characters

QueryResponse response = solr.query(query);

SolrDocumentList results = response.getResults();

**for** (**int** i = 0; i < results.size(); ++i) {

System.*out*.println(results.get(i));

}

}  
  
The result of the above code fragment is an HTTP query like this :

[http://localhost:8080/solr-4.9.0/db-core/select?q=firstname%3sachin+AND+surname%3sinha](http://localhost:8080/solr-4.9.0/db-core/select?q=firstname%253sachin+AND+surname%253sinha)  
  
A sample output for the above query in the local machine running a Solr instance gave the following result: **Entity numbers: 100000313, 1000000396, 1000000420, 1000000537, 1404783332, 1404783340, 1404783357**



Figure 3: Solr Query Result Example

1. Single and Multi-tenant Support
   1. Multi-tenant support

Kindly refer to the following diagram:

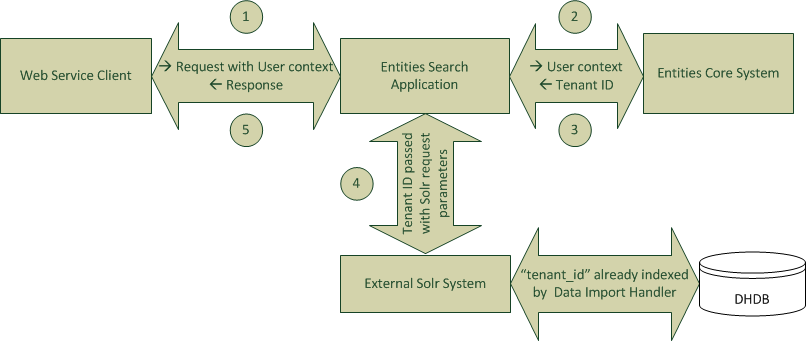


Figure 4: Multi-tenancy handling for FR1 and FR4

The above process is explained as below:

1. The Web service client will give a call to Entity Search application passing the requesting user context details along with the search input criteria.
2. The Entity Search application will pass on the user context details to Entities Core System by a web service call **getTenantIDFromUserContext** to retrieve the corresponding **tenant\_id** from the user context.
3. The **tenant\_id** is retrieved from Entities Core system call
4. The above **tenant\_id** is passed within the Solr Search Query to the External Solr System.
5. The search results (Entity numbers in case of FR4 and Entities information in case of FR1) are returned back as the response to the web service client.

Thus, effectively, the Search results will be limited to the caller tenant environment if the caller context details are passed within the web service client request.  
This approach also avoids hitting the databases for retrieving Entities information for FR1 as the same is retrieved from Solr itself in multi-tenant manner.

* 1. Single tenant (No Multi-tenant) support

If the caller context details are not passed within the web service client request, then, it will be assumed that the multi-tenant support is not required and the call to Entities Core System will be avoided. The Solr search query search will not filter the results based upon the tenant\_id as it is not provided in its input. Thus, effectively, the application will work as Single tenant (No multi-tenant support) manner without any filtering based upon tenant\_id.

1. Examples

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Description / Reference Section** | **Example Screenshot** |
| 1 | Lowercase Phonetic search |  |
| 2 | Wildcard search example |  |
| 3 | Exact string search example |  |
| 4 | Lowercase search example |  |
| 5 | DPE Role Exclusion example. The screenshots are taken before and after the required configuration respectively. |  |

1. References

This section specifies the complementary sources used.

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Reference** | **Location** |
| 1 | DHS\_Entities\_Search\_ComponentDesignSpecification.docx | CDS document location |
| 2 | Proof of Concept Documents: Comparison of Two approaches: DHS\_Design\_Approach\_Full\_Text\_Search\_Entities\_1.docx Presentation on SolrTM configuration: solr.odp Technical Specification of SolrTM: SOLR\_Standard\_1.0.docx |  |
| 3 | Apache LuceneTM | http://lucene.apache.org/core/ |
| 4 | SolrTM | http://lucene.apache.org/solr/ |
| 5 | Google paper | [The Anatomy of a Large-Scale Hypertextual Web Search Engine](http://infolab.stanford.edu/~backrub/google.html) |
| 6 | Scheduler | [a sample Scheduler](http://wiki.apache.org/solr/DataImportHandler#Scheduling) |
| 7 | Query Log Sheet |  |
| 8 | Sample Solr configuration |  |
| 9 | Entities Search Review Report |  |

1. Approval

| **Role** | **Name of the Person** | **Method of Approval** | **Date** |
| --- | --- | --- | --- |
|  |  |  |  |