CDISC Library maintains an index. Index is a collection of multiple document types. A document type is comprised of related metadata, or fields. Certain fields are pre-aggregated with a list of unique values to aid filtering or subsetting. The search API performs searches against this index. This strategy allows searches to be completed very efficiently, with most requests in less than 500 ms.

Note, CDISC Library uses Microsoft Cognitive Search as the search engine as of the 2020-11-11 release. Technical explanations such as query expressions, filters, and syntax alike can be found in the Technical References section below.

### Technical References
- Lucene query parser syntax for constructing search queries: https://lucene.apache.org/core/6_6_1/queryparser/org/apache/lucene/queryparser/classic/package-summary.html#N10013
- Lucene query operators for constructing search queries: https://lucene.apache.org/core/6_6_1/queryparser/org/apache/lucene/queryparser/simple/SimpleQueryParser.html
- Filter syntax to be used in search queries: https://docs.microsoft.com/en-us/azure/search/search-filters
- Hit highlighting to work with search results: https://docs.microsoft.com/en-us/azure/search/search-pagination-page-layout#hit-highlighting

The basic construct of a search API request is /
dr/search?q={searchExpression}&{scope}={scopeExpression}, where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>q={searchExpression}</td>
<td>Search expression to use in the query.</td>
<td>1</td>
</tr>
<tr>
<td>{scope}={scopeExpression}</td>
<td>The search scope and expression to filter against.</td>
<td>0,*</td>
</tr>
</tbody>
</table>

This section summarizes how the API would respond for each scenario listed:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Status code</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| Valid query with no matches | 200 Ok | The API returns a response, in JSON format as follows:

```json
{
    "hasMore": false,
    "hits": [],
    "totalHits": 0
}
```

| Valid query with at least one match | 200 Ok | The API returns a result set in JSON format. Each instance of matches will be in the hits element. The number of matches will appear in the totalHits element. Paging is supported. |
| Invalid query | 400 Bad Request | This includes, but not limited to, missing required parameter, malformed URL, and invalid paging. User should review the message and detail returned in the JSON-formatted response. |

A search expression `{searchExpression}` is a series of terms and operators. It may be one of the following:

<table>
<thead>
<tr>
<th>Expression type</th>
<th>Example</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single term</td>
<td>q=treatment</td>
<td>Return entries from index where the term treatment appears.</td>
</tr>
<tr>
<td>Multiple terms</td>
<td>q=date exposure treatment</td>
<td>Return entries from index where all the terms date exposure treatment appear in one document, regardless of word order.</td>
</tr>
</tbody>
</table>

Refer to the Technical References section for detailed explanations and examples.

In general, scopes are pre-aggregated field terms for filtering or subsetting searches. An endpoint is available to obtain a list of scopes:
Further, to obtain a list of pre-aggregated terms for each scope:

```
/mdr/search/scopes/{scope}
```

### Example

Below is a sample response from `/mdr/search/scopes/core`, where `core` is one of the scopes, referring to the Core column in dataset specification tables in CDASH, SDTM, and ADaM products:

```
{
    "hasMore": false,
    "total": 7,
    "values": [
        "Conditionally Required",
        "Expected",
        "Highly Recommended",
        "Optional",
        "Permissible",
        "Recommended/Conditional",
        "Required"
    ]
}
```

A scope expression `{scopeExpression}` is a series of terms and operators. It may be one of the following:

<table>
<thead>
<tr>
<th>Expression type</th>
<th>Example</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single term</td>
<td><code>q={searchExpression}&amp;core=Recommended</code></td>
<td>Limit the search to entries in the index where <code>core</code> contains the term <strong>Recommended</strong>.</td>
</tr>
<tr>
<td>Multiple terms</td>
<td><code>q={searchExpression}&amp;core=Conditionally Required</code></td>
<td>Limit the search to entries in the index where <code>core</code> contains the terms <strong>Conditionally Required</strong>.</td>
</tr>
</tbody>
</table>

Refer to the the Technical References section for detailed explanations and examples.

Highlighting is disabled by default.

It is possible to have highlights added to indicate what part of the index matched the search. The highlights appear as pairs in the JSON-formatted response returned. Client applications can consume and render additional visualization.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>highlight</code></td>
<td>A list of fields to highlight where matches are found. To enable highlighting, set the value of highlight to a comma-separated list of field names that should have highlighting applied. To highlight all fields, use <code>*</code> as the field name. The API will insert an element to each match instance, name <code>highlight</code>.</td>
<td>1</td>
</tr>
</tbody>
</table>

Refer to the the Technical References section for detailed explanations and examples.

Additional query parameters are available for paging. Paging is useful to limit and control the result set (or, hits) returned. By default, the API returns the first 100 hits. The total number of hits is at the end of a response.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>start</code></td>
<td>The initial result index to return, starting at 0. The default for <code>{startIndex}</code> is 0.</td>
<td>0..1</td>
</tr>
<tr>
<td><code>pageSize</code></td>
<td>The number of results to return, starting at <code>{startIndex}</code>. The default for <code>{pageSize}</code> is 100.</td>
<td>0..1</td>
</tr>
</tbody>
</table>

Paging works for both searches and scope listings, as follows:

<table>
<thead>
<tr>
<th>Application</th>
<th>API construct</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searches</td>
<td>/mdr/search?q={searchExpression}&amp;{scope}={scopeExpression}&amp;start={startIndex}&amp;pageSize={pageSize}</td>
<td>The total number of hits appears at the end of the API response, regardless of page index</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Scope listings</td>
<td>/mdr/search/scopes/{scope}&amp;start={startIndex}&amp;pageSize={pageSize}</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

Listing SDTMIG v3.3 variables whose core is Required, returns the 51st to 100th matches, out of 344 hits in total:

```
q=&start=50&pageSize=50&product=SDTMIG%20v3.3&core=Required
```