This documentation will explain the structure of an API request in kora. Array formats will be in JSON, though you can build them as PHP arrays and encode them later. We will also document helper functions within kora that exist to make this a smidge easier. The arrays are powerful and give a lot of control, but I want to make this easy and not overwhelming.

API URL: {YOUR_KORA_URL}/{API_FUNCTION_URI}

GET FUNCTIONS

These functions simply require a call to a URL. While the URLs themselves have some options (represented by the '{option}' blocks), no other request data needs to be provided.

KORA VERSION

Request Type: GET
Purpose: Gets the version number of the kora installation.
URI: api/version

PROJECT FORMS

Request Type: GET
Purpose: Gets a list of forms belonging to the given project, including their Form ID, name, and description.
URI: api/projects/{project_id}/forms

FORM FIELDS

Request Type: GET
Purpose: Gets a list of fields belonging to the given form, including their configurations.
URI: api/projects/{project_id}/forms/{form_id}/fields

FORM RECORD COUNT

Request Type: GET
Purpose: Gets the number of records belonging to a given form.
URI: api/projects/{project_id}/forms/{form_id}/recordCount

POST FUNCTIONS

These functions require extra post data to submit. Primarily, these functions involve the manipulation of records, but a couple for form/field related operations exist.

CREATE FORM

Request Type: POST
Purpose: Creates a form in a project using the JSON structure required of kora's import form feature.
URI: api/projects/{project_id}/forms/create
Parameters:
bearer_token: kora authentication token with CREATE permissions.
form: The JSON form data to create the form. TODO::CONNECT TO IMPORT DOC

EDIT FIELD OPTIONS

Request Type: PUT
Purpose: Edit field specific options of a field. NOTE: This can not edit general field configurations.
URI: api/projects/{project_id}/forms/{form_id}/fields
Parameters:
bearer_token: kora authentication token with EDIT permissions.
fields: The JSON structure of fields to be modified, and the information to modify with.

```
{
    “text field": {"Regex": "MY_NEW_REGEX_PATTERN"},
    “list field": {"Options": ["New List Item", "Another New List Item", ...]}
}
```

RECORD CREATE

Request Type: POST
Purpose: Create a new record in kora.
URI: api/create
Parameters:
bearer_token: kora authentication token with CREATE permissions.
form: The form id (or internal name) to create the record in.
fields: The JSON representing the record. NOTE: See IMPORT EXAMPLE below
zip_file (optional): Zip file containing the files belonging to the records. NOTE: Please upload this data into the $_FILES stream.

RECORD UPDATE

Request Type: PUT
Purpose: Edit a record in kora.
URI: api/edit
Parameters:
_method: NOTE: Due to Laravel’s understanding of HTTP request types, this value MUST be included and be set to "put".
bearer_token: kora authentication token with EDIT permissions.
form: The form id (or internal name) to edit the record in.

kid: The kora ID of the record to update.
fields: The JSON representing the record. NOTE: See IMPORT EXAMPLE below
zip_file (optional): Zip file containing the files belonging to the records. NOTE: Please upload this data into the $_FILES stream.

RECORD DELETE

Request Type: DELETE
Purpose: Delete record(s) in kora.
URI: api/delete
Parameters:
_method: NOTE: Due to Laravel's understanding of HTTP request types, this value MUST be included and be set to "delete".
bearer_token: kora authentication token with DELETE permissions.
form: The form id (or internal name) to create the record in.
kids: Array of records kora IDs to be deleted

```
["1-1-3", "1-1-7", ...]
```

IMPORT EXAMPLE

JSON structure for an imported record with examples for each field type.

```
[
	{
		"Text":"Your text value",
		"Rich Text":"Your <b>rich</b> text value",
		"Integer":"3",
		"Float":"3.21",
		"List":"List value",
		"Multi Select List":[
			"List value",
			"Another value",
			...
		],
		"Generated List":[
			"List value",
			"Another value",
			...
		],
		"Date":"2012-12-21",
		"DateTime":"2012-12-21 11:11:11",
		"Historical Date": {
			"prefix":"circa", NOTE: Circa tag is optional. Defaults to "".
			"year":"2012", NOTE: Always required.
			"month":"12", NOTE: Required when Day tag is specified.
			"day":"21", NOTE: Not required.
			"era":"CE" NOTE: Era tag is optional. Defaults to CE.
		},
		"Boolean": 1,
		…continued below
```

```
                "Geolocator":[
                        {
                                "geometry":{
                                        "location":{"lat":42.7314094,"lng":-84.476258}
                                },
                                "description":"Matrix",
                                "formatted_address":"288 Farm Ln, East Lansing, MI 48823"
                        },
                        ...
                ],
                "File Field":[
                        {
                                "name":"filename.ext",  NOTE: If the file in the zip is nested in a
folder, include the directories in front (i.e. someFolder/etc/filename.ext).
                                "caption":"Caption to explain file" NOTE: Only supported by
Gallery Fields.
                        }
                        ...
                ],
                "Associator":[
                        "1-23-4",
                        "5-67-8",
                        "9-10-11",
                        ...
                ],
                NOTE: reverseAssociations will connect the current records KID to an existing
record(s) in kora.
                "reverseAssociations": {
                        NOTE: For example, this will add the current KID to the associator field,
'Outside Associator', in record 1-3-77.
                        "Outside Associator": ["1-3-77","1-3-78",...],
                        ...
                }
        }
]
```

SEARCH POST FUNCTIONS

These functions require extra post data to submit. Primarily, these functions searching for sets of records within kora.

RECORD SEARCH

Request Type: POST
Purpose: Search for records within kora.
URI: api/search
Basic Parameters:

forms: The JSON structure used to define a search. It can contain multiple search sets. NOTE: See FORM SEARCH STRUCTURE below for more details.

format (optional | default: JSON | options: JSON,KORA_OLD,XML): The data format for the returned records.

Special Parameters:

merge (optional): When performing multiple form searches in the forms parameter, this function can map the names of similar fields in different search result sets to a single assigned name to represent those fields. NOTE: This will combine all search result sets into a single result set.

```
{
        “Custom Merged Name": [“field from set1", “field from set2", ...]
}
```

sort (optional): Performs a sort on all search results set, either by a custom field title designated in the merge parameter, or by common record metadata such as kora ID or timestamps. NOTE: This will combine all search result sets into a single result set.

```
[
        {“Custom Merged Name": "ASC"},
        {"created_at": "DESC"},
        ...
]
```

index (optional | requires: sort): Integer value that represents the index the search result set will start. i.e. You have a result set of 100 results. If you start at index 10, you will be given a result set that skips the first 10 records and give you the remaining 90 records.

count (optional | requires: sort): Integer value that represents the number of records returned. i.e. You have a result set of 100 results. A count of 20 would return the first 20 records in the result set. NOTE: Index is always applied before count.

FORM SEARCH STRUCTURE

Here we will define all the parameters that are required for setting the forms parameter in kora API record searches. The JSON structure can accept multiple form searches.

```
[
	{FORM_SEARCH_STUCTURE1},
	{FORM_SEARCH_STUCTURE2},
	...
]
```

This can be used to perform multiple form searches in a single API call. This is particular useful when applying the Special Parameters for your RECORD SEARCH call, as these parameters manipulate these form searches. For now, let's build a single form structure.

```
[
	{}
]
```

Basic Parameters - Enough to get back all the records:

bearer_token: kora authentication token with SEARCH permissions.

form: The form id (or internal name) to search within.

```
[
	{
		"bearer_token": "XXXXXXXXXXXXXXXXXXXXXX",
		"form": 37
	}
]
```

Metadata Parameters (optional) - Change up the structure of the returned records:

data (default: true): Determines if field data is returned with the records.

meta (default: true): Determines if record metadata is returned with the records.

size (default: false): Determines if the number of returned records is stored for this search result set. NOTE: This is recorded before the the count and index parameters take affect.

alt_names (default: false): Determines if field data within a record is indexed by its field name or alternative field name.

return_fields (default: "ALL" | requires: data): Array of field IDs that limit the returned field data.

assoc (default: false): Determines if field data for an associated field is returned as an array of records, or an array of kora IDs.

assoc_fields (default: "ALL" | requires: assoc): Array of field IDs that limit the returned field data for the returned associated records.

reverse_assoc (default: true): Determines if an array of kora IDs that associate to a returned record are returned with that record.

```
[
    {
        "bearer_token": "XXXXXXXXXXXXXXXXXXXXXX",
        "form": 37,
        "data": false,
        "meta": false,
        "size": true,
        "real_names": true,
        "return_fields": [“field 1", “field 2", ...],
        "assoc": true,
        "assoc_fields": [“assoc field 1", “assoc field 2", ...],
        "reverse_assoc": false
    }
]
```

Order Parameters (optional) - Manipulate the order and size of returned records:

sort: Performs a sort on a search result set, either by a field ID, or by common record metadata such as kora ID or timestamps. NOTE: Common metadata includes "kid", "legacy_kid", "created_at", and "updated_at".

index: Integer value that represents the index the search result set will start. i.e. You have a result set of 100 results. If you start at index 10, you will be given a result set that skips the first 10 records and give you the remaining 90 records.

count: Integer value that represents the number of records returned. i.e. You have a result set of 100 results. A count of 20 would return the first 20 records in the result set. NOTE: Index is always applied before count.

```
[
    {
        "bearer_token": "XXXXXXXXXXXXXXXXXXXXXX",
        "form": 37,
        "sort": [
            {“field 1": "ASC"},
            {“field 2": "DESC"},
            ...
        ],
        "index": 3,
        "count": 7
    }
]
```

Filter Parameters (optional) - For a search result set, return the frequency of values in the field data for every field in that set of records.

filters (default: false): Determines if we want filters back for the returned records.

filter_count (default: 1 | requires: filters): Determines how frequent a field value needs to occur to be returned as a filter.

filter_fields (default: "ALL" | requires: filters): Array of field IDs that limit the fields to be processed for filters.

```
[
    {
        "bearer_token": "XXXXXXXXXXXXXXXXXXXXXX",
        "form": 37,
        "filters": true,
        "filter_count": 5,
        "filter_fields": [“field 1", “field 2", ...]
    }
]
```

Search Parameters - Now it's time to introduce the queries parameter. Queries can be used to refine the results of a single form search:

queries (optional): Array of queries to refine the search.

logic (requires: queries): Defines how the results from each search query should be combined, using AND/OR logic. Each number represents a query in the queries array, based on its index in the array. NOTE: If this variable is not used, all query results will be OR'd together by default.

```
[
    {
        "bearer_token": "XXXXXXXXXXXXXXXXXXXXXX",
        "form": 37,
        "queries": [
            {QUERY_STRUCTURE_0},
            {QUERY_STRUCTURE_1},
            {QUERY_STRUCTURE_2},
            {QUERY_STRUCTURE_3}
        ],
        "logic": {
            "and": [
                1,
                {
                    "or": [
                        0,
                        2
                    ]
                },
                3
            ]
        }
    }
]
```

Query Structure Parameters - The parameters required of a single search query.

search (options: advanced,kid,kid,legacy_kid): The type of search query being executed.

not (default: false): Determines if the query returns the set of records that do not meet this search.

adv_fields (search=advanced): The JSON structure for an advanced search. NOTE: See ADVANCED QUERY SEARCH STRUCTURE below for more details.

key_words (search=keyword): Array of search terms to evaluate in a keyword search.
key_fields (search=keyword): Array of field IDs that limit the fields to be processed in a keyword search.
key_method (search=keyword | default: OR | options: OR,AND): Defines how the results from each keyword term should be combined.
custom_wildcards (search=keyword | default: false): Determines if the terms in key_words will have wildcards provided by user, or automatically added.
kids (search=kid): Array of kora IDs to search for.
legacy_kids (search=legacy_kid): Array of legacy (pre v3) kora IDs to search for.

```
"queries": [
        {
                "search": "keyword",
                "key_words": ["golf football","tennis%"],
                "key_method": "AND"
                "key_fields": [“field 1”,"field 2"],
                "custom_wildcards": true,
                "not": true
        },
        {
                "search": "advanced",
                "adv_fields": {ADVANCED_SEARCH_QUERY_STRUCTURE}
        },
        {
                "search": "kid",
                "kids": ["1-1-0","1-1-1","1-1-2"],
                "not": true //default: false
        },
        {
                "search": "legacy_kid",
                "legacy_kids": ["1-1F-0","1-1F-1","1-1F-2"],
                "not": true
        }
]
```

Advanced Search Query Structure

The parameters required for an advanced search query. These are primarily field type to search parameter pairings.

Basic Parameters:

negative (default: false): Determines if the query returns the set of records that do not meet this search. NOTE: Same as the not parameter, but for a single field parameter of the advanced search, rather than the result of the whole result search.

empty (default: false): Determines if we want records where there is no data in this field.

```
"adv_fields": {
        “some field": {
                "some_search_parameter": "some_data",
                "negative": true
        }
        “some other field": {
                "empty": true
        }
}
```

Field Specific Parameters - Explains the search parameters for each field type in an advanced search. NOTE: File field types do not support advanced search queries, only keyword search queries.

```
“Text Field": {
        "input": "Your search string"
        “partial”: false //do we want an exact or partial match
}
“Rich Text Field": {
        "input": "Your search string"
}
“Integer Field": {
        "left": -13 //left bound of search range, ignore for -infinity
        "right": 37 //right bound of search range, ignore for infinity
        "invert": false //determines if we look outside or inside the range
}
“Float Field": {
        "left": -13.3 //left bound of search range, ignore for -infinity
        "right": 37.1 //right bound of search range, ignore for infinity
        "invert": false //determines if we look outside or inside the range
}
“List Field": {
        "input": "List option"
}
“Multi Select List Field": {
        "input": ["List option", "another List Option", ...]
}
“Generated List Field": {
        "input": ["List option", "another List Option", ...]
}
```

```
 “Date Field": {
        "begin_month": 1
        "begin_day": 13
        "begin_year": 1337
        "end_month": 8
        "end_day": 19
        "end_year": 1990
}
“Date Time Field": {
        "begin_month": 1
        "begin_day": 13
        "begin_year": 1337
        "begin_hour": 1
        "begin_minute": 0
        "begin_second": 0
        "end_month": 8
        "end_day": 19
        "end_year": 1990
        "end_hour": 5
        "end_minute": 12
        "end_second": 52
}
“Historical Date Field": //
        "begin_month": 1
        "begin_day": 13
        "begin_year": 1337
        "begin_era": "CE" //options include CE, BCE, BP, and KYA BP
        "end_month": 8
        "end_day": 19
        "end_year": 1990
        "end_era": "CE"
}
“Boolean Field": {
        "input": true
}
“Geolocator Field": {
        "lat": 42.7314094 //latitude of search point
        "lng": -84.476258 //longitude of search point
        "range": 500 //distance in kilometers from the search point
}
“Associator Field": {
        "input": ["1-1-0","1-1-1","1-1-2",...]
        “any”: false //should the record contain all provided kora IDs or at least one
}
“Combo List Field": {
        “sub field name”: {
                //Use sub field structure above
        }
}
```

API Helper Functions

This section will describe a few helper functions that are provided in kora for generating the form JSON used in search. It ultimately creates it as a PHP array that you can JSON encode before sending out. They are provided in the class App/FieldHelpers/koraSearch.php. NOTE: This file also includes the functionality to perform a legacy KORA_Search. Additionally, if you want to use the KORA_Search portion of the file, please set the $kora3ApiURL variable in this file.

FORM SEARCH BUILDER

Takes queries and other information to build the full forms string value in an array.

string $**fid** - Form ID
string $**token** - Token to authenticate search
array $**flags** - Array of flags that customize the search further
Possible array values include “data”, ”meta”, ”size”, ”assoc”, ”reverse_assoc”, ”filters”, ”real_names”
NOTE: Please see Form Search Structure for information on filters
array $**fields** - For each record, the fields that should actually be returned
To return all fields, supply a blank array
array $**sort** - Defines what fields we are sorting by
NOTE: Please see Form Search Structure for information on sort arrays
array $**queries** - The collection of query arrays in the search
NOTE: Please see the QUERY BUILDERS below for information on array items
array $**qLogic** - Logic array for the search
NOTE: Please see Form Search Structure for information on logic arrays
int $**index** (default=null) - In final result set, what record should we start at
int $**count** (default=null) - Determines, starting from $index, how many records to return
int $**filterCount** (default=null) - Determines the minimum threshold for a filter to appear
array $**fitlerFlids** (default=null) - Determines which fields will return filter data
To return all fields, supply a blank array
array $**assocFlids** (default=null) - Determines which fields will return data for associated records
To return all fields, supply a blank array
RETURNS array - Array representation of the form search for the API

KEYWORD QUERY BUILDER

Builds the query string for a keyword search.

string $**keyString** - Keywords for the search
string $**method** - Defines if search is AND, OR, or EXACT
bool $**not** (default=false) - Get the negative results of the search
array $**flids** (default=empty array) - Specific fields to search in
bool $**customWildCards** (default=false) - Is the user providing wildcards
RETURNS array - The query array

KID QUERY BUILDER

Builds the query string for a KID search.

array $**kids** - KIDs we are searching for
bool $**not** (default=false) - Get the negative results of the search
bool $**legacy** (default=false) - Search for legacy kid instead
RETURNS array - The query array

ADVANCED QUERY BUILDER

Builds the query string for an advanced search.

array $**advData** - Array with search parameters for advanced search
NOTE: Please see Advanced Search Query Structure for information on this array
bool $**not** (default=false) - Get the negative results of the search
RETURNS array - The query array

QUERY LOGIC BUILDER

Builds simple array with two queries and a comparison operator.

array $**queryObj1** - Index of query object in your query array, or another logic array
string $**operator** - Comparison operator
array $**queryObj2** - Index of 2nd query object in your query array, or another logic array
RETURNS array - Logic array