CIOC API User Guide

Release Online Resources 3.7 / Client Tracker 3.2

Katherine Lambacher, KCL Software Solutions Inc.

September 03, 2015

Contents

CHAPTER 1

CIOC Online Resources API Introduction

1.1 Available API Types

CIOC has several APIs written specific members, an AIRS XML Export API, and one general use API for searching and record details that is available for both the Community Information and Volunteer modules.

The APIs available as of the last update to this documentation are:

- Standard Remote Search and Details API
- AIRS XML Export Feed [1]
- CSV, Excel, or Sharing Format Export Feed
- 211ontario.ca Export `Custom API`
- O211 Export `Custom API, DEPRECATED`
- CLBC Export & CLBC Update `Custom API`

To search and/or display records from an external website, the correct API to use is the Standard Remote Search and Details API. In addition to formal APIs, it is possible to create a CSV, Excel, or Sharing Format Export Feed, which may be useful in limited circumstances to automate regular download of an custom-formatted Excel/CSV file for use in an external system.

Note: This document **does not cover use of the 211ontario.ca, O211 Export, or CLBC Export / Update APIs**. These are purpose-built for specific external systems, and not generally applicable for use by others.

1.2 API Security

All APIs require user authentication. It is highly recommended that SSL and a compatible domain (i.e. *.cioc.ca) be used; support of non-SSL domains is deprecated and will not be supported in the future. Authentication uses HTTP Basic Authentication. It is recommended that API Accounts be be configured on a **per-project basis**, and used **exclusively for the API** for the given project. Non-API (regular user) accounts should generally not be given API access. Because of the nature of how API account information is accessed and stored, these accounts should never be provided with any update or setup privileges of any kind. If you have a developer that needs these extra privileges, they should have one account for themselves and a different account for production API use.

[1] Although the AIRS Export was purpose-built for a particular organization and has some modifications from the standard, it may still be suitable for others, particularly those sharing to iCarol.

Warning: Do not under any circumstances include account credentials in public scripts or webpages or distribute them in clear-text with applications

Each API requires a user account with permission to access that specific API. By default, no user has access to any of the APIs. All APIs except the Excel/CSV Feed are enabled in the User Type Configuration in the API Section. Because the CSV/Excel export tool has a front-end UI for users as well, any account that can access a particular Excel Output Profile through the regular UI can also access the feed; view the documentation for CSV, Excel, or Sharing Format Export Feed for more information on access control for Excel Output Profiles.

Note: The User Type and assigned View determine which records can be accessed through the API [2]. Records available in each API are the same as the account would have access to if used through the regular software UI, so logging into the UI with the API account is the best method to confirm that access restrictions are correct.

Most purpose-built APIs have a **fixed set of fields** that cannot be changed, while CIOC's Standard Remote Search and Details API has customizable data access restrictions that mirror what a User would see if logged in to the regular UI. Do not allow the API account to have access to Views that should not be displayed via the API (e.g. via View switching). Best practice often means configuring a View for the API on a per-project basis so that the security and data access can be properly controlled.

Warning: In general, an API View should not be configured to have switching access to other Views unless that is intended functionality based on how you are choosing to use the API.

1.3 API Statistics

"Record View" statistics are kept for the Standard Remote Search and Details API. There is an API flag in the Record View statistics table in the software to indicate when the Record View was the result of an API call, and the User ID of the account calling the API is also recorded. This information can be used to do reporting on the use of the API for a specific account/project. Note that API calls register as "logged in" uses for statistical purposes.

[2] The CLBC Export & Update APIs and the deprecated O211 Export API have fixed criteria and do not used View-based criteria. Usage of these APIs are not covered by this documentation.

CHAPTER 2

Standard Remote Search and Details API

Note: For general information on CIOC Online Resources APIs, including information on configuring user accounts to have API access, read the CIOC Online Resources API Introduction.

2.1 Introduction

This *Standard Remote Search and Details API* is designed to be a flexible, general-purpose solution to embedding search results and record details on another (non-CIOC) site. It is intended to be used in real-time, in that a request to the remote site results in a request to this API. All API endpoints are avialable in **JSON** (JavaScript Object Notation), a lightweight data interchange format that is commonly used to exchange data over the internet. The major endpoints are also available as **XML**.

All API endpoints except the *Volunteer Application* API (aka *"Yes, I'd Like to Volunteer!"*) accept `GET` requests and take parameters as a URL-encoded query string. The *Volunteer Application* endpoint takes a `POST` request with a `application/x-www-form-urlencoded` body. There is one common parameter that all endpoints accept: `format`, which may be set to `xml` to override the default return format of JSON and instead return an XML formatted result.

2.2 Search API

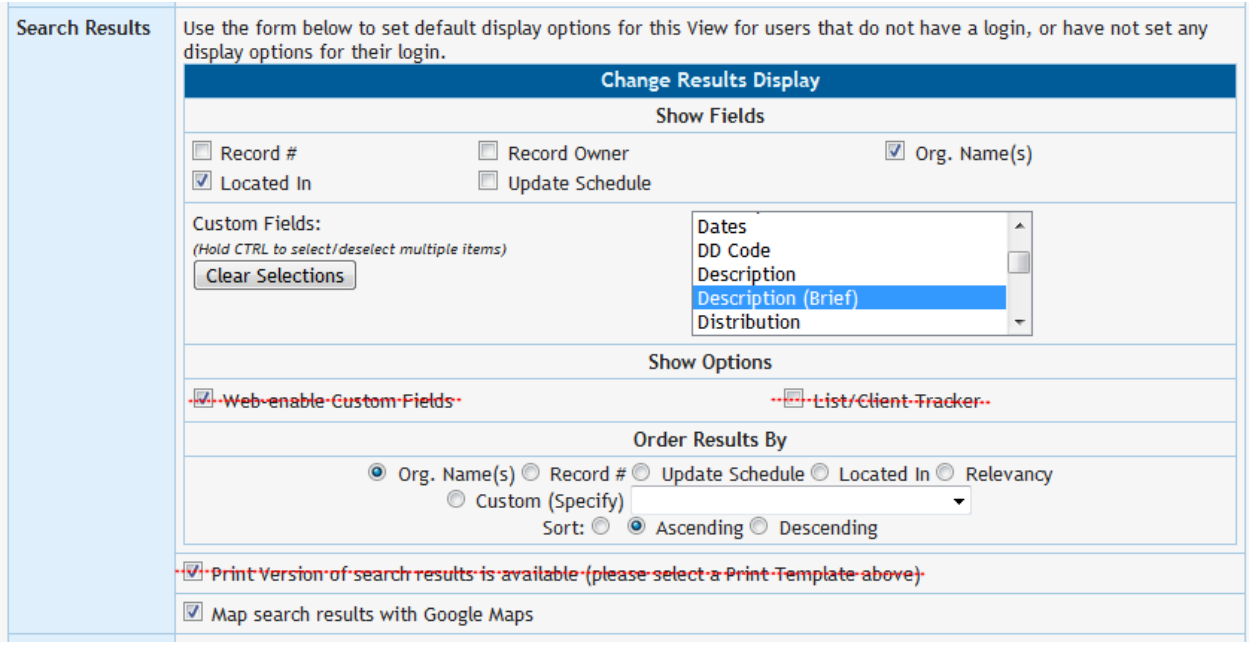
The Search API is useful for producing a list of records according to a specific criteria. The fields available are configurable (See *Configuring Fields*). The Search API is intended to be used "live" to search the database from an external system, though the implementer may choose to provide some periodic caching of search results to reduce transit for common requests, such as generating a list of records for a webpage that lists services in a particular category. If caching results, a reasonable refresh period should be determined.

2.2.1 Configuring Fields

The API results must include either Record Number *or* Name, and the link to the Details API for the given record. Many additional items can be added to the results through the setup for the View connected to the API Account.

Note: It is strongly recommended that the number of fields be limited to the record name plus 2 or 3 additional fields; because a very large number of records can be returned, the inclusion of many fields can create enormous request sizes.

The list of fields is configured through the **Display Options of the API User Account's View** in the **Search Results** section (see image below - Community Information settings shown). The *Show Options* and *Print Version* settings do not apply to the API. To enable the Latitude and Longitude fields in the API, turn on the *Map Search Results with Google Maps* option. Best practice is for each API project to have its own View, which allows control over what the API sees without impacting other uses.



2.2.2 Access URL and Search Parameters

The main Search API is accessed through the following URLs (substituting your domain):

Community Information [Search] `https://yoursite.cioc.ca/rpc/orgsearch.asp`

Volunteer Opportunity [Search] `https://yoursite.cioc.ca/rpc/oppsearch.asp`

Volunteer Opportunity [Browse by Organization] `https://yoursite.cioc.ca/rpc/browseoppbyorg`

All Search API URLs accept a **format** parameter `format=[xml|json]`. If no format is specified, the default is JSON.

The Community Information and Volunteer **Search** API URLs support all search parameters available to the API account from the regular Search Results page from the UI (`results.asp`/`bresults.asp` in the UI), but at this time it does not include Saved Searches which are accessed through `sresults.asp` or some of the specialized Child Care search types accessed only through `cresults.asp`). The Volunteer *Browse by Organization* URL does not accept any additional search parameters, and always returns a list of organizations with active opportunities.

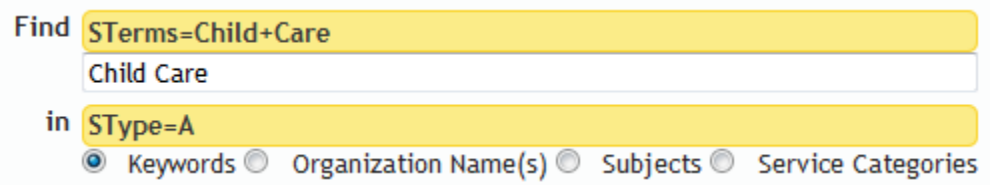
Which Parameters to Use?

There are hundreds of search parameters in the software, and which ones are available to the current user depend on various setup options. The best way to find the parameters available to the current user is by reviewing the Basic or Advanced Search forms.

Method 1: Execute a Search, Parse the URL Execute any search that posts to `results.asp` or `bresults.asp`, and view the available parameters in the URL of the results page. This includes Basic and Advanced Search, all Taxonomy Searches, Volunteer Step-by-Step Search, and any search linked from the Record Details page.

Method 2: Copy the Form You can directly copy HTML from the Basic or Advanced Search form and use it in another application.

Method 3: Turn on Search Parameter Key Display The Basic and Advanced Search forms have a special settings to allow you to see what the search parameters would be as you type criteria into the form:



Available search forms with the Search Parameter Key Display:

- `https://yoursite.cioc.ca/?SearchParameterKey=on`
- `https://yoursite.cioc.ca/advsrch.asp?SearchParameterKey=on`
- `https://yoursite.cioc.ca/volunteer/?SearchParameterKey=on`
- `https://yoursite.cioc.ca/volunteer/advsrch.asp?SearchParameterKey=on`

Note: It is intentionally difficult to produce a complete list of records with no criteria. The Search API is *not* intended to be used for complete database downloads: use the CSV, Excel, or Sharing Format Export Feed for that purpose in the Community Information module. In the Volunteer Opportunities module, a complete list of current opportunities can be retrieved using the criteria `DisplayStatus=C` to retrieve all currently active opportunities.

Note: Get Clarity with Codes: Many fields that appear with numeric IDs when searching in the regular CIOC UI can be converted to use human-readable Codes instead. Using Codes provides clarity, predictability, and portability and is highly recommended. A search with **ID** in it (`PBID`, `GHID`, `AIID`, and so on...) can be converted to a Code search by changing ID to Code (`PBCode`, `GHCode`, `AICode`, and so on...). You can add/edit the codes for each item and display a list of available codes via the Setup Area for the given item. General Headings can be searched by Code as explained here, but also by *Name* - just change the `GHType` parameter to `NM` and the `GHID` parameter to `GH`.

2.2.3 Download the Schemas

The following schemas indicate what to expect when retrieving output in XML. If using a JSON format, the structure and contents are the same and the XSD serves as a reasonable guide. See the *Sample JSON* for more examples.

- `Community Information: Search API Schema`
- `Volunteer Opportunity: Search API Schema`
- `Volunteer Opportunity: Browse by Organization API Schema`

There are 3 pieces of information provided by default in the search results for the Search APIs:

- `NUM`/`VNUM` - The Record ID
- `RECORD_DETAILS` - A link to the record on the CIOC Site
- `API_RECORD_DETAILS` - A link to the API URL for the record

Other fields provided in the results correspond to the Search Results settings in the account's View. A list of possible **Field name values** is available from the current CIOC Online Resources Field List.

2.2.4 Error Handling

JSON search results will be an object with a string `error` property. If there is an error of any kind, the `error` property will be a string value explaining the error. If there is no error, the error property will be `null`.

XML search results will have a `<root>` element containing an `<error>` element and a `<recordset>` element. If there is an error of any kind, the `<error>` element will contain a string value explaining the error and the <recordset> element will be a self-closed empty element. If there is no error, the <error> element will be a self-closed empty element, and the `<recordset>` element will contain a series of `<record>` elements (or if no records match the query, the `<recordset>` element will also be a self-closed empty element).

Note: Sanitize your inputs. Although CIOC processes inputs that come via the API, it is *emphatically* recommended that external applications using the API take some responsibility for minimizing potential disruption to the production system that can be caused by bad actors. Perform a basic level of pre-processing of public requests before passing them to the API by only passing on parameters you are actually expecting to receive, and checking the type of those parameters when appropriate. This also allows an extra level of protection if you have unintentionally configured the security of your API to have access to more data than you intended.

2.2.5 Sample XML

Community Information Search Result

```
<root>
        <error/>
        <recordset>
                <record NUM="ABC00001">
                        <field name="RECORD_DETAILS">https://yoursite.cioc.ca/record/ABC00001</field>
                        <field name="API_RECORD_DETAILS">https://yoursite.cioc.ca/rpc/record/ABC00001
                        <field name="ORG_NAME">Organization Name</field>
                        <field name="LATITUDE">43.718193</field>
                        <field name="LONGITUDE">-79.722915</field>
                        <field name="LOCATED_IN_CM">Brampton (City of)</field>
                </record>
        </recordset>
</root>
```

Volunteer Opportunity Search Result

```
<?xml version="1.0" encoding="utf-8"?>
<root>
        <error/>
        <recordset>
                <record VNUM="V-DEF0001" OPID="123">
                        <field name="RECORD_DETAILS">https://yoursite.cioc.ca/volunteer/record/V-DEF
                        <field name="API_RECORD_DETAILS">http://tika-dev/rpc/opportunity/V-DEF0001?fo
                        <field name="POSITION_TITLE">Position Title</field>
                        <field name="ORG_NAME">Organization Name</field>
                        <field name="COMM_BALLS">&lt;img src=&quot;https://yoursite.cioc.ca/images/te
                        <field name="DUTIES">Description of Duties</field>
                        <field name="INTERESTS">Interest 1; Interest 2; Interest 3</field>
                </record>
        </recordset>
</root>
```

Volunteer Opportunity: Browse by Organization

```
<?xml version="1.0" encoding="utf-8"?>
<root>
        <item>
                <ORG_NAME>Organization Name</ORG_NAME>
                <NUM>ABC00001</NUM>
                <OP_COUNT>4</OP_COUNT>
                <OPP_SEARCH_LINK>https://yoursite.cioc.ca/rpc/oppsearch.asp?NUM=ABC00001&amp;format=x
                <API_RECORD_DETAILS>https://yoursite.cioc.ca/rpc/record/ABC00001?format=xml</API_RECO
        </item>
</root>
```

2.2.6 Sample JSON

Community Information Search Result

```
{
        "error":null,
        "recordset":[
                {
                "NUM":"ABC00001",
                "RECORD_DETAILS":"https://yoursite.cioc.ca/record/ABC00001",
                "API_RECORD_DETAILS":"https://yoursite.cioc.ca/record/ABC00001",
                "ORG_NAME":"Organization Name",
                "LATITUDE":43.718193,
                "LONGITUDE":-79.722915,
                "LOCATED_IN_CM":"Brampton (City of)"
           }
        ]
}
```

Volunteer Opportunity Search Result

```
{
        "error":null,
        "recordset":[
                {
                "OPID":123,
                "VNUM":"V-DEF0001",
                "RECORD_DETAILS":"https://yoursite.cioc.ca/record/DEF0001",
                "API_RECORD_DETAILS":"https://yoursite.cioc.ca/record/DEF0001",
                "POSITION_TITLE":"Position Title",
                "ORG_NAME":"Organization Name",
                "COMM_BALLS": "<img src=\"https://yoursite.cioc.ca/images/teal_ball.gif\" alt=\"Commu
                "DUTIES":"Description of Duties",
                "INTERESTS":"Interest 1; Interest 2; Interest 3"
           }
        ]
}
```

Volunteer Opportunity: Browse by Organization


```
[
        {
                "ORG_NAME": "Organization Name",
                "NUM": "ABC00001",
                "OP_COUNT": 4,
                "OPP_SEARCH_LINK": "http://yoursite.cioc.ca/rpc/oppsearch.asp?NUM=ABC00001",
                "API_RECORD_DETAILS": "http://tika-dev/rpc/record/ABC00001"
        }
]
```


2.2.7 Additional Feeds

Using Supplementary JSON Feeds

To assist with performing searches, there are JSON-only feeds available that list available search criteria in some categories. All feeds are in the location: `https://yoursite.cioc.ca/jsonfeeds/`

Communities Feed

For use in the **CMID** Parameter: `community_generator.asp?term=partial+word_string`

Keyword Suggestions (Auto-complete)

For use in the **STerm** Parameter:

Community Information

- `cic_keyword_generator?term=partial+word_string&SearchType=[A|O|T|S]`
- A = Anywhere (Default), O = Org. Names, T= Taxonomy, S = Subjects

Volunteer Opportunities

- `vol_keyword_generator?term=partial+word_string&SearchType=[P|O]`
- P = Position Keywords, O = Org. Names

Areas of Interest Feed

For use in the **AIID** Parameter:

All Interest Categories: `interest_generator.asp?ShowAll=on`

Keyword Generation: `interest_generator.asp?term=partial+word+string`

By Interest Group ID(s): `interest_generator.asp?IGID=3,10,22`

General Heading HTML Option List Feed

There is a feed for generating **form HTML** for the General Headings of a selected publication. It is output as a list of HTML `option` elements. This feed includes **all** Headings, including non-public.

`heading_searchform.asp?PBID=123`

2.3 Record Details API

The Record Details API is for pulling the full details of a specific record. It is accessed through the following URLs

Community Information Record: `https://yoursite.cioc.ca/rpc/record/[NUM]`

Volunteer Opportunity Record: `https://yoursite.cioc.ca/rpc/opportunity/[VNUM]`

2.3.1 Configuring Fields

The fields available in the Record Details API are identical to what is available when the API account is viewing the Record Details page through the regular UI. To change the fields available or their re-organize them, change the **Field Groups** and **Details Fields** setup of the View used by the API Account. Best practice is for each API project to have its own View, which allows control over what the API sees without impacting other uses.

Note: The display order of fields *within* each Field Group is determined by the global Field Order setup. It is not possible to change the global Field Order without impacting every View in the database. Use Field Groups to cluster similar fields together.

2.3.2 Download the Schemas

The following schemas indicate what to expect when retrieving output in XML

- `Community Information Details API Schema`
- `Volunteer Opportunity Details API Schema`

If using a JSON format, the structure and contents are the same and the XSD serves as a reasonable guide.

Field Group name values are custom and determined by the View Setup. A list of possible **Field name values** is available from the current CIOC Online Resources Field List.

2.3.3 Error Handling

Any errors will be returned as HTTP errors. The only errors supported at this time are 404 (Not Found) and 401 (Permission Denied). 401 errors are provided with a textual reason if appropriate.

2.3.4 Sample JSON

Todo

Coming Soon

2.3.5 Sample XML

Todo

Coming Soon

2.4 Volunteer Application API

The *Volunteer Application* API (aka *"Yes, I'd Like to Volunteer!"*) takes a `POST` requests with a `application/x-www-form-urlencoded` body to the address `https://yoursite.cioc.ca/volunteer/volunteer2.asp`. Available parameters can be found on the form at `https://yoursite.cioc.ca/volunteer/volunteer.asp?VNUM=[ANY_VNUM]&SearchParameterKey=o` In addition to the parameters in that form, the API request needs to include the parameter `api=on` which indicates that the request is an API request and `VNUM=[vnum]` which indicates the opportunity to volunteer for.

Note: The API expects a response to CIOC's captcha. **If directly passing on submissions from the public without vetting (e.g. not requiring a user account of some kind or forcing a response to the CIOC captcha) then it is strongly recommended that you provide an alternative captcha**. If you provide an alternative captcha, the API consumer application will still have to respond to the CIOC captcha.

CHAPTER 3

AIRS XML Export Feed

Note: For general information on CIOC Online Resources APIs, including information on configuring user accounts to have API access, read the CIOC Online Resources API Introduction.

Todo

Coming Soon

CHAPTER 4

CSV, Excel, or Sharing Format Export Feed

Note: For general information on CIOC Online Resources APIs, read the CIOC Online Resources API Introduction.

Todo

Coming Soon

CHAPTER 5

Client Tracker Resource API

5.1 Overview

The Client Tracker allows for interaction with Resource Providers through a simple XML-based protocol. Provisions are made for registering a Resource Provider (aka Resource Type), fetching Resource Item details, determining that a user is currently handling a Request, and pushing Resource Items from a Resource Type into a user's current Request.

The Resource Type must expose URLs to the public Internet to provide Registration and Fetching. Resource Item Pushing requires the Resource Type to connect over the public internet to the Client Tracker host servers, but should still allow interaction with Resource Types that are only accessible via the end user's intranet.

5.1.1 Basic Conventions

All response messages can contain either a successful response or an error response. The error response can provide a message that will be displayed to the user indicating the source of the error.

5.2 Message Types

5.2.1 Registration

The registration message is designed to simplify the configuration of a Resource Type in the Client Tracker system. While all configuration items can be entered by hand, the registration message ensures accuracy when setting complex URL settings.

The Client Tracker server initiates a request for registration data by performing a HTTP GET request to the Resource Type's registration URL. The definition of the Registration response is defined in `registration.xsd`. The URL is stored with the other settings so that changes to the registration information can be fetched at the administrator's request.

5.2.2 Resource Item Fetching

The Client Tracker allows the option of having a user fetch Resource Item details from the Resource Type by ID. At the user's request the Client Tracker server performs an HTTP GET to the configured Fetch URL. The ID value entered by the user is substituted for any instance of `[ID]` in the Fetch URL. The definition of the Fetch response is defined in `resourceinfo.xsd`. The ID may be of any type that can be stored in 50 Unicode characters or less.

5.2.3 Resource Item Pushing

The Client Tracker also allows the Resource Type to push Resource Items into the user's current Request. There are 3 pieces of information that the Client Tracker provides to the Resource Type that must be provided with every request to the Client Tracker:

1. The name of the user for which the request is being made
2. The remote key
3. The Client Tracker's internal id for the Resource Type

These pieces of information are provided to the Resource Type for the current user through the Launch URL. When the user "Launches" a Resource Type, the Client Tracker opens a new browser window directed at the Launch URL where any instance of `[CTID]`, `[LOGIN]` and `[KEY]` are replaced with the appropriate values to be provided in messages from the Resource Type. These values could be stored in the user's session with the Resource Type.

The Resource Type can determine if the user is currently in a Request by making an HTTP POST request to `https://clienttracker.cioc.ca/ct/rpc/is_in_request` where the body of the POST is an isInRequest message as defined by `incallrequest.xsd`. The client tracker will respond with an isInRequest response message as defined in `incallresponse.xsd`. This response will include a list of Resource Item ids that have already been associated with the user's current Request, if applicable.

The Resource Type can push a Resource Item into the user's current Request by making an HTTP POST request to `https://clienttracker.cioc.ca/ct/rpc/add_resource` where the body of the POST is a pushResource message as defined in `pushresource.xsd`, with a response message as defined in `pushresourceresponse.xsd`.

CHAPTER 6

Client Tracker Public Request API

6.1 Introduction and Overview

The CIOC Client Tracker has support for accepting requests from the public. Client Tracker administrators may designate a set of Surveys as publicly available when used with a specific Classification of Inquiry. Individual survey question parts must also be flagged as public in order for that question/part of a survey to be available on public forms. Requests submitted by members of the public will come in similar to "feedback" in the Online Resources software. These requests go into a queue, where they must be accepted and processed by a staff member or - if they are not appropriate - the requests can be rejected. No public request is included in any other search or report until it is first accepted by a staff member.

There are three methods for generating and processing a public request form:

- **Directly in Client Tracker** (best for those with no website or no technical expertise)
- **Partially-embedded in a remote site** (best for those with a website and minimal technical expertise)
- **Fully-embedded in a remote site** (best for those who want their visitors to always remain in their website during the request process)

All use of the API is provided through HTTP requests to URLs of the form `https://clienttracker.cioc.ca/public/[AGENCY_CODE]/[CLASSIFICATION_CODE]` where `[AGENCY_CODE]` is the 3 letter agency code of staff members that will handle the public request and `[CLASSIFICATION_CODE]` is a code from a Classification of Inquiry.

6.2 API Methods

6.2.1 Method 1: Using Forms Created Directly in the Client Tracker

With the method 1, the public user navigates with a browser to a URL as indicated above. No further programming is required; the software takes care of generating and processing the form. To use this method, it would be recommended to provide a simple link to the address `https://clienttracker.cioc.ca/public/[AGENCY_CODE]/[CLASSIFICATION_CODE]` with substitutions for your Agency Code and the Classification of Inquiry code, as described above.

6.2.2 Method 2 & 3: External Website Integration

Warning: Technical stuff ahead!

In methods 2 and 3, a remote server makes a HTTP `GET` request to a URL as indicated above including an ApiKey query string parameter with a valid Public API Key (see next section). The remote server will be provided with a JSON response that includes the HTML rendered form contents (without form tag) and other relevant data as described below in the section covering JSON GET Responses.

In method 2 (partially-embedded in a remote site), the HTML form is configured to POST to the same URL as the GET request and the optional redirect_to query string parameter can be provided indicate a URL to which visitors will be redirected after their request has been successfully submitted. If no redirect_to query string parameter is provided, then the visitor will be redirected to a generic thank you message that can be viewed at `https://clienttracker.cioc.ca/thanks`.

In method 3 (fully-embedded in a remote site), the same URL used to to handle a `GET` request accepts a `POST` request with an `application/x-www-form-urlencoded` encoded body and returns a JSON encoded response as described in below in JSON POST Responses. As with the remote GET request, a valid `ApiKey` parameter is required. The ApiKey parameter may be provided as part of the query string or in the POST body.

Public API Keys

A super user may create API keys for their Client Tracker membership in the "Public Request API Keys" section of the Client Tracker Administration area. Keys are Base 64 encode random strings that look something like *bJITwVg9GvEBgW7nDI0N3bZF6U5hUGpaHN2kqQjS6jTg*. A name may be associated with the Key to remind administrators of the intended use of the key, but this is not used in the validation process. Because keys are base 64 encoded they may contain special characters (like +) which require escaping before use in a query string or `application/x-www-form-urlencoded` encoded `POST` body.

Warning: The API key is not intended to be exposed on public pages.

JSON Get Response

When using method 2 or 3 (partially- or fully-embedded in a remote site) the current form is requested from the Client Tracker application with a get request to `https://clienttracker.cioc.ca/public/[AGENCY_CODE]/[CLASSIFICATION_CODE]?ApiKey=[API_KEY]`. The Client Tracker will respond with a JSON encoded response like the following:

```
{
        "alert": null,
        "form": "<!-- HTML Form Content -->",
        "errors: null
}
```

Because there were no errors, both alert and errors are null. form normally contains the contents of the form to display, without the `<form>` tag. The form tag is omitted so that you can decide to handle the POST back from the visitor's browser or have the Client Tracker site handle it for you.

JSON POST Response

When using mode 3 (fully-embedded in remote site) the visitor's form submission is is handled by the remote site and a POST request is made on the visitor's behalf to the Client Tracker with the request data. If there is no error, the following JSON encoded response will be returned:

```
{
        "success": true
}
```

In the event of a validation error the Client Tracker will return a JSON encoded response like the following:

```
{
        "alert": "General Error Message",
        "form": "<!-- HTML Form Content -->",
        "errors: {
                "FieldName": "Specific Error",
                "FieldName2": "Other Specific Error"
        }
}
```

As with the JSON Get Response the full html for the HTML form is returned as form. alert will always contain an error message that should be displayed to the visitor that explains the errors in a general way. If there were basic form value validation errors, errors will also contain the specific fields that had an error and details of the error. Note that these messages have already been rendered into the form HTML content but may be useful if you perform an AJAX POST to your site and want to return the error messages back to the page for visitor notification.