DEV URL:

All: <https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do>

Active: <https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?status=active>

Closed: <https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?status=closed>

QA URL:

All: <https://was-intra-qa.toronto.ca/LobbyistAdminWeb/disclosure.do>

Active: <https://was-intra-qa.toronto.ca/LobbyistAdminWeb/disclosure.do?status=active>

Closed: <https://was-intra-qa.toronto.ca/LobbyistAdminWeb/disclosure.do?status=closed>

# Lobby Registry API requirements

The following is a list of requirements needed to produce data as per the prototype.

We do not know yet if we can successfully use the aggregator or not. If you see Aggregator syntax, it represents the aggregator syntax.

## Data required for Filtering

### Subject Matters Listing

A list of subject matters is required to populate a predictive search filter subject matters

Aggregator Syntax:

<http://was8-inter-dev.toronto.ca/cc_sr_v1/data/clerks_lds_active?fields=SM_SubjectMatter&limit=999999&sort=SM_SubjectMatter>

Size: approx.: 115KB – then loop and distinct.

### Subject Matter Status

These will be hard coded to be "Active", "Closed", as they are hard coded in the application.

## Subject Matter Type

These will be hard coded to be "Consultant", "In-house", "Voluntary" as they are hard coded in the application.

### Lobbyist Type

These will be hard coded to be "Consultant", "In-House Lobbyist", "Sr. Officer" as they are hard coded in the application.

## Subject Matter JSON Data Feed

Currently the feed provides one parameter: Status=active or Status=closed

<https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?status=active>

The feed should be enhanced to include the following **optional** parameters that become part of an "AND"ed where clause if they are passed. Each parameter is used with a "contains" operation except for the hard coded parameters where "equals" is appropriate and special cases where we want to use "Starts With"

### Search=<string> search all fields looking for <string> value.

### SMStatus=<string> one of active|closed

### SMRegNo=<string>

Aggregator Syntax:

http://was8-inter-dev.toronto.ca/cc\_sr\_v1/data/clerks\_lds\_active?fields=SM\_SMNumber&q=SM\_SMNumber%20LIKE%20%27SM2\*%27&limit=4000

### SMType =<string> one of Consultant, In-House or Voluntary

### SubjectMatter=<string>

### LobbyistName=<string>

where

Registrant.LastName contains <string> or Registrant.FirstName contains <string>

Or

(Communications exists and Communication.LobbyistFirstName contains <string> or Communication.LobbyistLastName contains <string>

### BusinessOrg=<string>

selects where Firms exists and Firm.Name contains <string> or BENEFICIARY.tradeName contains <string>

### Client=<string>

-- selects where Beneficiaries exists and BENEFICIARY.Type = "Client"  
 and BENEFICIARY.Name contains <string> or BENEFICIARY.tradeName contains <string>

### BeneficiaryName=<string>

-- selects where Beneficiaries exists and BENEFICIARY.Name contains <string> or BENEFICIARY.tradeName contains <string>

### GovAgency=<string>

-- selects where Gmtfundings exists and Gmtfunding.GMTName contains <string> or and Gmtfunding.Program contains <string>

### POHName=<string>

-- selects where Communications exists and Communications. POH\_Name contains <string>

### POHPos=<string>

-- selects where Communications exists and Communications. POH\_Position contains <string>

### POHType=<string>

-- selects where Communications exists and Communications. POH\_Type contains <string>

### POHOffice=<string>

-- selects where Communications exists and Communications. POH\_Type contains <string>

{There are POHs and Lobbyist under committee meeting, are they included in the search for Lobbyists and search or POHs?] I assume so.. but there are not too many committee meetings

### GrassRoots=true

– selects where a 'Grassroots' relationship exists

### Committee=true

– selects where a 'meetings relationship exists

### Datefrom=YYYYMMDD

### DateTo=YYYYMMDD

### Start=<int>

-- starting row number to return (for paging)

### Limit=<int>

-- number of records to return (for paging

### Sort=

Example:

**DineSafe** feed example:

<https://secure.toronto.ca/dinesafe/dataTableEstablishments.json?callback=jQuery17206593767133565219_1455641940981&status=1,2,3&sEcho=3&iColumns=4&sColumns=&iDisplayStart=30&iDisplayLength=10&mDataProp_0=0&mDataProp_1=1&mDataProp_2=2&mDataProp_3=3&iSortingCols=1&iSortCol_0=1&sSortDir_0=asc&bSortable_0=true&bSortable_1=true&bSortable_2=false&bSortable_3=true&_=1455642029930>

1) To get all grassroots SMs.

[https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?GrassRoots=true](https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?GrassRoots=true%20)

[the "=true" is redundant, the existence of the parameter should be enough]

2) Example with Numeric Paging

Get all active SMs where the client is "NorthWest Property Corporation".

When we are listing by SM, we will always use numeric paging (pages 1,2,3 etc)

<https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Status=Active&Client=NorthWest%20Property%20Corporation,start=1,limit=50>

We may need a second call to go to the next page

<https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Status=Active&Client=NorthWest%20Property%20Corporation,start=51,limit=50>

When numeric paging is involved, we need to know the overall number of records so we can show (page 1 of xxx, total records: yyy). The JSON expands with a totaling fields such as:

{"totalRecs" : 150,

"currentRecs" : 50, -- not sure we need this

[{"SM" : { …SMNumber:SMxxxxx....},

"SM" : { …SMNumber:SMyyyyy… }

+48 more SMs as part of 1st page.

}]

}

3) Example listing where alphanumeric paging is used.

The Beneficiary listing will show data organized by the 1st letter of the beneficiary Name

3a) Start with 26 parallel calls to determine existence of data to disable/enable letter paging options.

[https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=[A|B|C|D|E.....]\*,start=1,limit=1](https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=%5bA|B|C|D|E.....%5d*,start=1,limit=1)

ex:

https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=A\*&start=1&limit=1

[https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=B\*&start=1&limit=1](https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=B*&start=1&limit=1)

…

3b) Get a page of data, listing beneficiaries which start with an "A"

[https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=A\*,sort=Beneficiary.Name](https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=A*,sort=Beneficiary.Name)

"A\*" means instead of contains use "Starts With"

In this example, we get all the SMs, where the beneficiary name starts with a B.

If the user clicks on the "G" paging option, a 2nd call to see "beneficiaries" that start with an G is called.

[https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=G\*,sort=Beneficiary.Name](https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?Beneficiary=G*,sort=Beneficiary.Name)

## Structure of SM is as follows:

* Core SM Information
* A registrant segment
* One or more meetings (committee meetings)
* One of more communications
* One of more firms
* One or more beneficiaries
* One or more grassroots
* One or more Government Fundings

"SM" : {

"Type" : "In-house",

"EffectiveDate" : "2014-07-10",

"ProposedEndDate" : "2015-06-30",

"ProposedStartDate" : "2008-02-14",

"SMNumber" : "SM10651",

"Status" : "Active",

"Particulars" : "Property Taxes",

"InitialApprovalDate" : "2008-02-13",

"SubjectMatter" : "Budget"

"Gmtfundings": {

"Gmtfunding": [{

"GMTName": "Government of Canada",

"Program": "Department of the Environment"

},

{

"GMTName": "Government of Alberta",

"Program": "Alberta Energy"

},

{

"GMTName": "U.S Government",

"Program": "USDA"

}]

}

"Meetings" : {

"Meeting" : {

"Committee" : "Government Relations Committee",

"Date" : "2011-03-15",

"Lobbyists" : {

"Lobbyist" : [{

"Business" : "",

"Type" : "In-House Lobbyist",

"Suffix" : "",

"Number" : "10342H",

"FirstName" : "Von",

"Prefix" : "",

"LastName" : "Palmer",

"MiddleInitials" : ""

}, {

"Business" : "",

"Type" : "Committee Member",

"Suffix" : "",

"Number" : "15967H",

"FirstName" : "Calvin",

"Prefix" : "",

"LastName" : "Weinfeld",

"MiddleInitials" : ""

},

"POHS" : {

"POH" : {

"Name" : "Kristyn Wong-Tam",

"Type" : "Member of Council",

"Office" : "Ward 27 Toronto Centre-Rosedale",

"Title" : "Councillor"

}

},

"Desc" : ""

}

},

"Beneficiaries": {

"BENEFICIARY": [{

"FiscalEnd": "",

"FiscalStart": "",

"Name": "Whitecastle New Urban Fund 2",

"TradeName": "",

"Type": "Client",

"BusinessAddress": {

"PostalCode": "M4T 2S3",

"Country": "Canada",

"Province": "ON",

"AddressLine1": "22 St. Clair Avenue East",

"City": "Toronto",

"AddressLine2": "Suite1010"

}

},

{

"FiscalEnd": "",

"FiscalStart": "",

"Name": "Stephen Diamond",

"TradeName": "",

"Type": "Controlling interest holder",

"BusinessAddress": {

"PostalCode": "M4T 2S3",

"Country": "Canada",

"Province": "ON",

"AddressLine1": "22 St. Clair Ave. E",

"City": "Toronto",

"AddressLine2": "Suite 1010"

}

}]

},

"Communications" : {

"Communication" : [{

"PreviousPublicOfficeHoldLastDate" : "",

"LobbyistBusinessAddress" : {

"Phone" : "416-443-8100",

"PostalCode" : "M3B 3N1",

"Country" : "Canada",

"Province" : "Ontario",

"AddressLine1" : "1400 Don Mills Road",

"City" : "Toronto",

"AddressLine2" : ""

},

"LobbyistFirstName" : "Don",

"LobbyistMiddleInitials" : "",

"CommunicationMethod" : "",

"LobbyistType" : "Sr. Officer",

"LobbyistPrefix" : "Mr.",

"LobbyistBusiness" : "",

"LobbyistSuffix" : "",

"PreviousPublicOfficePositionProgramName" : "",

"POH\_Position" : "",

"POH\_Name" : "",

"POH\_Office" : "",

"PreviousPublicOfficeHoldPosition" : "",

"CommunicationDate" : "",

"PreviousPublicOfficeHolder" : "no",

"CommunicationGroupId" : "",

"LobbyistLastName" : "Richardson",

"LobbyistPositionTitle" : "Chief Executive Officer",

"POH\_Type" : "",

"LobbyistNumber" : "10341S-1"

}, {

"PreviousPublicOfficeHoldLastDate" : "",

"LobbyistBusinessAddress" : {

"Phone" : "416-443-8176",

"PostalCode" : "M3B 3N1",

"Country" : "Canada",

"Province" : "Ontario",

"AddressLine1" : "1400 Don Mills Road",

"City" : "Toronto",

"AddressLine2" : ""

},

"LobbyistFirstName" : "Cynthia",

"LobbyistMiddleInitials" : "",

"CommunicationMethod" : "",

"LobbyistType" : "In-House Lobbyist",

"LobbyistPrefix" : "",

"LobbyistBusiness" : "",

"LobbyistSuffix" : "",

"PreviousPublicOfficePositionProgramName" : ".",

"POH\_Position" : "",

"POH\_Name" : "",

"POH\_Office" : "",

"PreviousPublicOfficeHoldPosition" : ".",

"CommunicationDate" : "",

"PreviousPublicOfficeHolder" : "no",

"CommunicationGroupId" : "",

"LobbyistLastName" : "Lee",

"LobbyistPositionTitle" : "Manager, Political Outreach",

"POH\_Type" : "",

"LobbyistNumber" : "16867H"

},

]

},

"Grassroots" : {

"GRASSROOT" : {

"Community" : "public, members",

"Target" : "Mayor and City Councillors",

"EndDate" : "2009-04-03",

"StartDate" : "2009-03-30"

}

},

"Firms" : {

"Firm" : {

"FiscalEnd" : "2015-06-30",

"FiscalStart" : "2014-07-01",

"Name" : "Toronto Real Estate Board",

"Description" : "Provide services to members (REALTORS(R)) "TradeName" : "",

"BusinessType" : "Business/ Industry/ Trade Association",

"Type" : "In-house",

"BusinessAddress" : {

"PostalCode" : "M3B 3N1",

"Country" : "Canada",

"Province" : "Ontario",

"AddressLine1" : "1400 Don Mills Road",

"City" : "Toronto",

"AddressLine2" : ""

}

}

},

"Registrant" : {

"PreviousPublicOfficeHoldLastDate" : "",

"PositionTitle" : "Chief Executive Officer",

"Type" : "In-house",

"Suffix" : "",

"EffectiveDate" : "2014-09-11",

"Prefix" : "Mr.",

"LastName" : "Di Michele",

"PreviousPublicOfficePositionProgramName" : "",

"RegistrationNUmberWithSoNum" : "10341S-2",

"Status" : "Active",

"RegistrationNUmber" : "10341S",

"PreviousPublicOfficeHoldPosition" : "",

"FirstName" : "John",

"PreviousPublicOfficeHolder" : "no",

"MiddleInitials" : "",

"BusinessAddress" : {

"Phone" : "416-443-8100",

"PostalCode" : "M3B 3N1",

"Country" : "Canada",

"Province" : "Ontario",

"AddressLine1" : "1400 Don Mills Road",

"City" : "Toronto",

"AddressLine2" : ""

}

},

}

}

Note: Below is the syntax of the aggregator tool. Instead of hard coding specific parameters, we could follow that syntax, then the java code would have to decode the parameters and dynamically apply the select and sort

Alternative Syntax (following Aggregator concept)

[https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?q[Status=Active,Client=NorthWest%20Property%20Corporation](https://was8-intra-dev.toronto.ca/LobbyistAdminWeb/disclosure.do?q%5bStatus=Active,Client=NorthWest%20Property%20Corporation),dateFrom='20100101'dateto='20160101'],sort=SM.SMNumber

# City Aggregator overview

<http://was8-inter-dev.toronto.ca/cc_sr_v1/data/clerks_lds_active_regs/?load=http://www1.toronto.ca/City%2520Of%2520Toronto/Sandbox/Rob/disclosure_active.json&skipRoot=true>

Loaded http://www1.toronto.ca/City%20Of%20Toronto/Sandbox/Rob/disclosure\_active.json?skipRoot=true, with settings:

skipRoot=true

The indexed fields are:

* DateOnly SM\_InitialApprovalDate
* String SM\_SubjectMatter
* String SM\_Status
* DateOnly SM\_EffectiveDate
* String SM\_Type
* DateOnly SM\_ProposedEndDate
* String SM\_SMNumber
* String SM\_Particulars
* DateOnly SM\_ProposedStartDate

**Instructions on how to query**

# Data query overview

The Data query library aggregates JSON data-sets and make them queryable online through a REST API.

## Data query API specification

The data query API has the following operations: 1. List data-sets 2. Retrieve full data-set 3. Retrieve item from set, by item # 4. Retrieve item from set, by item ID(s) 5. Search data-set

#### List data-sets operation

/data

Returns a RSS listing of all data-sets, with links to retrieve the full data-set

#### Retrieve full data-set operation

/data/[data-set]/

Returns the entire data of the data-set

#### Retrieve item from set, by item # operation

/data/[data-set]/[item #]

Returns the item data, index starting with item # 0

#### Retrieve item from set, by item ID(s) operation

/data/[data-set]/[item ID field1]=[item ID value1]&[item ID field-n]=[item ID value-n]

Returns the item data

#### Search data-set operation

/data/[data-set]/?q=[query]

Returns the data of the matching items

## Parameters

The following parameters are supported:

* q=[query] - Search query to find items
* fields=[fields] - Restricts which fields are returned. Comma separated list of fieldnames.
* start=[start-number] - Determines the starting point in the result to return. 1 is the first one, and default.
* limit=[limit-number] - Sets the limit to how many items are returned. 30 by default
* out=[JSON/RSS/ATOM/@URL] - Format to return data in. Either JSON, RSS, ATOM or @URL. JSON is default return format.
* sort=[fields] - Sort result using numerical, date and date-time fields. Comma separated list of fieldnames. For reverse order sorting, append the field name with DESC, like in SQL statements

Some examples:

1. /data/ce/?q=eventDate=20151025&start=1&limit=3&sort=eventHour
2. /data/doors\_open/?q=location\_postal LIKE 'M4E\*'

## Data query syntax (q parameter)

The search query parameter uses a SQL-like language to find items in the dataset.

Some exampels are:

1. first\_name='Foo' AND age <=30 AND city='Boston'
2. age IN (31, 30, 50)
3. city IN ('Cincinatti', 'New York', 'Boyds')
4. age BETWEEN 35 and 50 AND first\_name LIKE 'Br\*'
5. first\_name='John' AND (age<=45 AND city NOT IN ('New York', 'Boston', 'Atlanta'))
6. last\_name != 'Smith'
7. publish\_date = 20150101
8. event\_timestamp BETWEEN 20150101140000 and 20150101190000
9. city NOT IN ('York', 'Halton')

Instead of LIKE, the character ~ can also be used.

As illustrated, the parameters can be nested, include LIKE statements, include date ranges and time ranges.

Search fields are case-senstive

### Query limitations

The searches are executed by the Lucene search engine, so it is not as precise as a database.

1. LIKE statement wild-card characters CAN NOT be used in the front (e.g. name LIKE '\*son')
2. LIKE statements search for a word in the field, not just the first word. This is both a feature and a restriction

# Data aggregation overview

To initialize the data aggregation:

/data/[data-set]/?load=[data-set-source-url[?loading-parameters]]

The data-aggregation is deployed separately in production from the data-access, in an intranet server environment. The data-aggregation is configured with URLs that are polled on a timed basis, using the standard Resource Manager settings.

The supporting loading parameters are: 1. interval - data loading frequency, in seconds. 2. skipRoot - parameter to skip wrapper objects. Either true (to skip one level), or a number to skip specific number of levels 3. internal - indicating if this resource will be publicly available or not

Please note that when loading multiple parameters, you need to separate each with %26 when entered directly in a browser URL bar, for instance:

/data/doors\_open/?load=<http://wx.toronto.ca/inter/culture/doorsopen.nsf/OpenDataDOTBuilding.JSON?OpenPage?skipRoot=true%26interval=120>

## JSON source parsing

If the JSON file consist of a JSON-array, the individual entries in the array become separate data-set items, that are individually indexed and searchable.

The root attributes of each JSON object are indexed, which means that they can later be searched, sorted by (numeric and dates only) and individually retrieved, using the fields parameter. Other internal complex objects are loaded and provided, but not indexed.

### Dates and Timestamps

Date fields are automatically converted into a numeric-only date-format (yyyyMMdd) and timestamp fields are also converted to a numeric timestamp format (yyyyMMddHHmmss). This simplifies and standardizes indexing, searching and sorting of dates and date-time-stamps.

* The following date formats are recognized and parsed:
  + yyyy-MM-dd
  + dd/MM/yy
  + MMM d, yyyy
  + MMMMMMMM d, yyyy
  + dd MMMMMM yyyy
  + yy-MM-dd
* The following timestamp formats are recognized and parsed:
  + yyyy-MM-dd'T'HH:mm:ss
  + yyyy-MM-dd'T'HH:mm:ss.SSS
  + yyyy-MM-dd'T'HH:mm:ss.SSSZ

yyyy-MM-dd'T'HH:mm:ss.SSSXXX