

1. Home	4
1.1 Introduction	4
1.2 Quickstart	4
1.3 Development	7
1.3.1 API Framework	7
1.3.1.1 API - Examples	11
1.3.1.1.1 Actionscript - Cookies - Login, Get Person, Logout	12
1.3.1.1.2 Actionscript - Hashing a Password	13
1.3.1.1.3 Actionscript - HTTP Request	14
1.3.1.1.4 Actionscript - Login, Get a Source, Edit the Source, Test the new Source, Save the new Source, Logout	14
1.3.1.1.5 Actionscript - Login, Get Communities, Perform a Query, Logout	18
1.3.1.1.6 cURL - Cookies - Login, Get Person, Logout	21
1.3.1.1.7 cURL - HTTP Request	21
1.3.1.1.8 cURL - Login, Get Map Reduce Jobs, Get Map Reduce Job Results, Logout	21
1.3.1.1.9 cURL - Login, Get Public Communities, Request to Join Community, Logout	21
1.3.1.1.10 Java - Cookies - Login, Get Person, Logout	22
1.3.1.1.11 Java - Hashing a Password	24
1.3.1.1.12 Java - HTTP Request	24
1.3.1.1.13 Java - Login, Create a Community, Invite User, Logout	25
1.3.1.1.14 Java - Login, share a hadoop Jar file, schedule a map reduce job with jar, Logout	28
1.3.1.1.15 Web Browser - Cookies - Login, Get Person, Logout	32
1.3.1.1.16 Web Browser - Hashing a Password	32
1.3.1.1.17 Web Browser - HTTP Request	32
1.3.1.2 Auth - Deactivate	32
1.3.1.3 Auth - Forgot Password	33
1.3.1.4 Auth - Keep Alive	34
1.3.1.5 Auth - Login	35
1.3.1.6 Auth - Login - Admin	36
1.3.1.7 Auth - Logout	36
1.3.1.8 Chrome Source Extension	37
1.3.1.9 Community ID Regular Expression Usage	40
1.3.1.10 Config - Source	41
1.3.1.10.1 Source building - reference	42
1.3.1.10.2 Source gallery	43
1.3.1.10.3 Specifying a Data Source	111
1.3.1.10.4 Structured Analysis - Overview	118
1.3.1.10.5 Unstructured Analysis - Overview	137
1.3.1.11 Config - Source - Add (DEPRECATED)	143
1.3.1.12 Config - Source - Approve	144
1.3.1.13 Config - Source - Bad	145
1.3.1.14 Config - Source - Decline	148
1.3.1.15 Config - Source - Delete	149
1.3.1.16 Config - Source - Delete - Docs	149
1.3.1.17 Config - Source - Get	150
1.3.1.18 Config - Source - Good	151
1.3.1.18.1 Config - Source - Pending	152
1.3.1.19 Config - Source - Save	155
1.3.1.20 Config - Source - Test	157
1.3.1.21 Config - Source - User	158
1.3.1.22 Custom - Map Reduce	160
1.3.1.23 Custom - Map Reduce - Get Jobs	161
1.3.1.24 Custom - Map Reduce - Get Results	162
1.3.1.25 Custom - Map Reduce - Remove Job	163
1.3.1.26 Custom - Map Reduce - Schedule Job	164
1.3.1.27 Custom - Map Reduce - Update Job	166
1.3.1.28 Custom - Saved Query - Schedule Job	168
1.3.1.29 Custom - Saved Query - Update Job	169
1.3.1.30 GUI Utilities - Plugin Manager	171
1.3.1.30.1 Javascript Engine for Plugin Manager	175
1.3.1.31 GUI utilities - uploading files	179
1.3.1.32 Infinit.e.Manager	183
1.3.1.32.1 Infinit.e.Manager - Communities	184
1.3.1.32.2 Infinit.e.Manager - People	186
1.3.1.32.3 Infinit.e.Manager - Sources	188
1.3.1.33 Knowledge - Document - File - Get	192
1.3.1.34 Knowledge - Document - Get	193
1.3.1.35 Knowledge - Document - Query	202
1.3.1.35.1 Knowledge - Document - Query - Manual Aliases	208
1.3.1.35.2 Knowledge - Query - Input Options	211
1.3.1.35.3 Knowledge - Query - Output options	212
1.3.1.35.4 Knowledge - Query - Query Terms	232
1.3.1.35.5 Knowledge - Query - Scoring Parameters	237

1.3.1.36 Knowledge - Feature - Alias Suggest	240
1.3.1.37 Knowledge - Feature - Association Suggest	241
1.3.1.38 Knowledge - Feature - Entity Suggest	242
1.3.1.39 Knowledge - Feature - Geo Suggest - (NOT YET IMPLEMENTED)	247
1.3.1.40 Social - Community - Add	247
1.3.1.41 Social - Community - Get	248
1.3.1.42 Social - Community - Get All	251
1.3.1.43 Social - Community - Get Private	253
1.3.1.44 Social - Community - Get Public	255
1.3.1.45 Social - Community - Get System	257
1.3.1.46 Social - Community - Member - Invite	259
1.3.1.47 Social - Community - Member - Join	259
1.3.1.48 Social - Community - Member - Leave	260
1.3.1.49 Social - Community - Member - Update - Status	261
1.3.1.50 Social - Community - Member - Update - Type	261
1.3.1.51 Social - Community - Remove	262
1.3.1.52 Social - Community - Request Response	263
1.3.1.53 Social - Community - Update	264
1.3.1.54 Social - GUI - Modules - Delete	265
1.3.1.55 Social - GUI - Modules - Get	265
1.3.1.56 Social - GUI - Modules - Install	270
1.3.1.57 Social - GUI - Modules - Search	272
1.3.1.58 Social - GUI - Modules - User - Get	273
1.3.1.59 Social - GUI - Modules - User - Set	278
1.3.1.60 Social - GUI - UISetup - Get	278
1.3.1.61 Social - GUI - UISetup - Update	279
1.3.1.62 Social - Person - Delete	280
1.3.1.63 Social - Person - Get	281
1.3.1.64 Social - Person - List	282
1.3.1.65 Social - Person - Register	283
1.3.1.66 Social - Person - Update	285
1.3.1.67 Social - Person - Update - Email	286
1.3.1.68 Social - Person - Update - Password	287
1.3.1.69 Social - Share - Add - Binary	288
1.3.1.70 Social - Share - Add - Community	288
1.3.1.71 Social - Share - Add - JSON	289
1.3.1.72 Social - Share - Add - Reference	290
1.3.1.73 Social - Share - Endorse	291
1.3.1.74 Social - Share - Get	292
1.3.1.75 Social - Share - Remove	293
1.3.1.76 Social - Share - Remove - Community	294
1.3.1.77 Social - Share - Save - JSON	294
1.3.1.78 Social - Share - Search	295
1.3.1.79 Social - Share - Update - Binary	297
1.3.1.80 Social - Share - Update - JSON	298
1.3.1.81 Social - Share - Update - Reference	299
1.3.2 JSON object formats	300
1.3.2.1 Documents and their sub-objects (entities, associations, user metadata, aggregations)	300
1.3.2.1.1 Association JSON format	302
1.3.2.1.2 Document JSON format	309
1.3.2.1.3 Entity JSON format	316
1.3.2.1.4 Geo JSON format	321
1.3.2.1.5 Metadata JSON format	323
1.3.2.2 Social configuration objects	331
1.3.2.2.1 Community JSON formats	331
1.3.2.2.2 Person JSON format	336
1.3.2.2.3 Share JSON Format	337
1.3.2.2.4 User creation and updating JSON object formats	338
1.3.2.3 Source configuration objects	340
1.3.2.3.1 Authentication object	343
1.3.2.3.2 Database object	343
1.3.2.3.3 Enrichment engines	344
1.3.2.3.4 Feed object	345
1.3.2.3.5 File object	347
1.3.2.3.6 StructuredAnalysis object	350
1.3.2.3.7 UnstructuredAnalysis object	353
1.3.3 Widget Framework	355
1.3.3.1 Different views of the data provided to the widget	356
1.3.3.2 Infinite Widget and Plugin Use	357
1.3.3.3 Override the GUI setup from the URL	359
1.3.3.4 Special Widget Information	360
1.3.3.5 Widget Framework - uploading widgets	364

1.3.3.6 Writing your first Infnit.e Widget	367
1.4 Third Party Notices	376

Home

Getting Started

[Introduction](#) | [Quickstart](#)

Development

[API Framework](#) | [Widget Framework](#) | [JSON object formats](#)

Documentation

[Enterprise](#) | [Applications](#)

Support

[Support](#) | [Training](#) | [Consulting](#)

Community

[Web](#) | [Blog](#)

Meta

[AWS EULA](#) | [Privacy Policy](#) | [Third Party Notices](#)

Archived documentation

- [August 2013 release documentation](#) (covers [this release](#))
- [July 2013 release documentation](#) (covers [this release](#))
- [June 2013 release documentation](#) (covers [this release](#))
- (see attachments for others)

Introduction

The concept for Infinit.e was created around a need for an extremely open, extremely scalable, socially aware knowledge discovery and analytics platform built for today's web.

It wasn't designed in a lab, it was designed and built based on our on experiences in the analytics community. We took a very pragmatic approach and decided that building from scratch was not the best idea; instead looking at the problem holistically by using tools and techniques that users, developers and IT administrators can run with.

What if we can make the most scalable, flexible, socially aware and easy to use analytics platform available? ...That is our goal.

Philosophy

- By simplifying the interfaces we can create a way for subject matter experts to extend and solve their analytics problems with simple-to-use tools
- Make the most scalable analytics system possible by using tools built for today's web such as distributed highly available nosql databases and search engines
- Enable all this to run both within a cloud architecture, and also on (potentially much smaller) tactical architectures

Quickstart

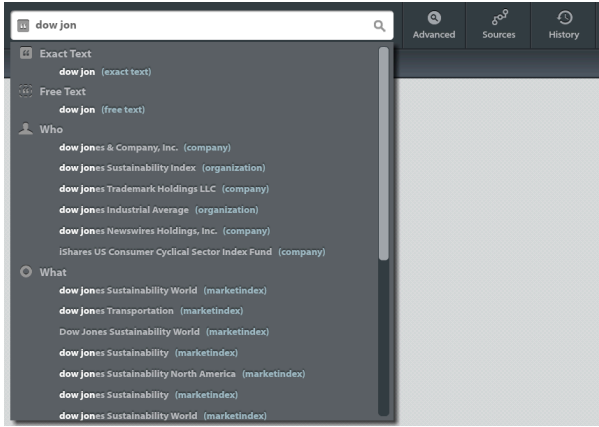
Access

In order to access the Infinit.e application you must first have an account. After you have your account established you can navigate to <http://infinit.e.ikanow.com> (or the address of your server in the case of custom installs/AWS PaaS/etc)

After you have successfully logged in you will be presented with a blank canvas. If you have accessed the software previously you will be presented with the any previously opened visualizations (widgets).

Searching using Infinite

Once you are inside the application you can use the Quick Search to start the Query Focused Dataset (QFD) process. This interface allows you to chain queries based on free text and exact field matches as well as boolean operations. For more advanced searches you can use the Advanced Search feature.



Once you begin typing text into the Simple Search you will start to see "search as you type" functionality. This is broken out by entity type (who, what, where) to better help you define and refine your query. As in the example above, the user has the option of performing an exact or free text query on the term "dow jones", or selecting one of the individual entities to query against (i.e. who: dow jones & company inc. (company)). This allows user to refine further refine queries based on the available entity data.

After you have made your selections you are ready to issue your simple query by clicking the search button or hitting enter. Global searches can also be performed using "*" operator. This will pull back the latest information in the environment.

For more advanced searches you can use the Advanced Search interface (tab next to the query bar). This will allow you to construct very complex search criteria, adding in addition search terms, associations, and geographic or date parameters. The video below illustrates an example how to use the Advanced Search interface.

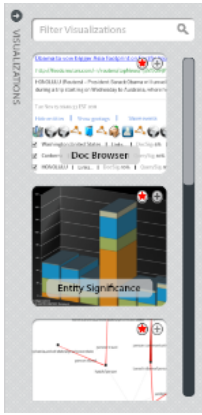
TODO ADD VIDEO HERE

Visualizations (Widgets)

Once you are inside the application and have issued your query from the Simple Search or Advanced Search you can begin to investigate and analyze your results by adding one or multiple Visualization Widgets to the canvas.



We have included a handful of "out-of-the-box" Visualizations to assist in the analytical process. Instructions are available that illustrate how to develop and integrate custom widgets.



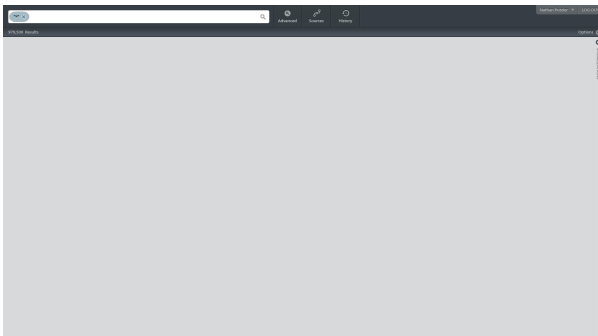
In order to add visualizations to the canvas you can double click on the icon, click the + at the top right of the icon, or "drag" the visualization on to the canvas. Included are several starter visualizations which are outlined below.

- **Map** displays a geospatial view of the QFD that the user submits. This maps out the geo locations that are extracted from the documents that are brought back from the query.
- **Entity Significance** creates a bar chart visualization ranking entities using metrics such as significance, coverage, and frequency. As you mouse over you can see the calculated values.
- **Doc Viewer** provides a result list view of the information that includes ranking and sorting as well as way to view the extracted entity values. From this interface you can also filter your queries.
- **Event Graph** provides a semantic graph visualization of entities returned by the query focused dataset and provides sub graph functionality by filtering the entity categories.
- **Timeline** provides a temporal view of document results.

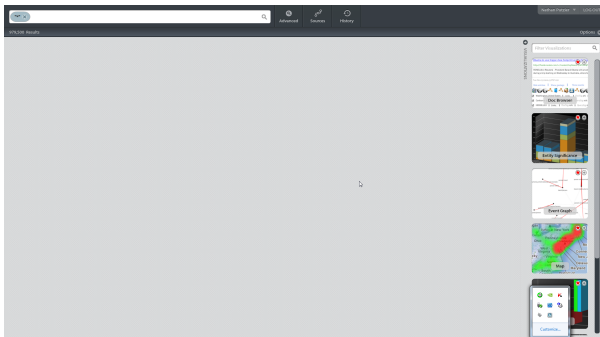
Here is short video illustrating how to use the visualizations.

Canvas

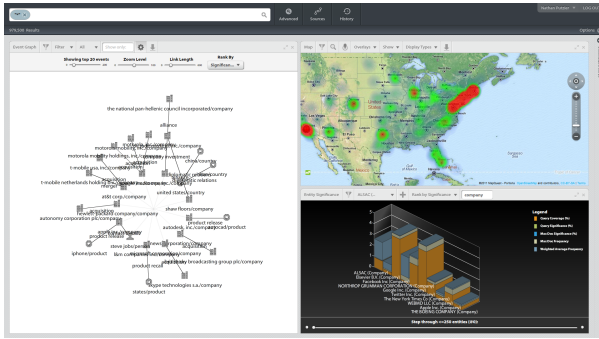
The Infinet.e user interface was designed to provide alot of the same rich features similiar to that of operating system within a browser. The canvas provides the palette to display the visualization interfaces and manipulate them by doing such things as tiling or cascading as well as resizing and moving.



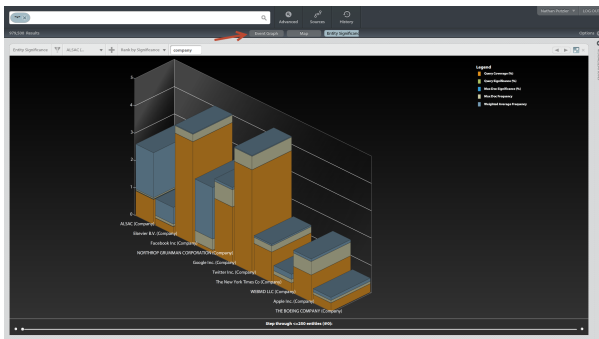
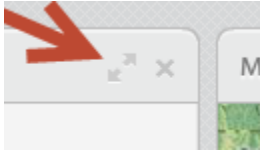
Expanding the visualization tray displays a list of available widgets with their thumbnail.



You can add up to four visualizations from the tray on the right side of the canvas. After you have added them your interface might look like something below.



From here you have few options. You can maximize widgets, and tab between open widgets using the navigation buttons at the top of the workspace.



The example of above illustrates the interface after you have issued a tile command to layout the available space in the browser.

Filtering

Filtering is offered throughout the environment that will help you to refine your query focused dataset. This can be accomplished through a mixture of ways. Most of the out of the box widgets provide a way for you to add filters to the query or directly filter by adding individual items to the filter set. The video below illustrates how filtering works with a few of our widgets.

Each widget has a toolbar header that provides functionality to filter to page through results among other types of functionality. From each you can mouse over the icons to be provided a tool tip that describes the action. We will discuss this filtering mechanism in more detail later on.

Here is short video illustrating how you perform filtering in two of the available widgets.

Development

API Framework

Getting started with the Infinit.e API

The Infinite API is available for non-commercial use by outside developers. Commercial use is possible by prior arrangement and is available as part of our enterprise appliance solutions. For commercial inquiries about the API please contact us at support@ikanow.com

Overview

The Infint.e API consists of a set of callable [RESTful](#) endpoints.

To perform an action using the Infinit.e API, you need to select the appropriate endpoint to service your request based on your need and provide the specified arguments. Documentation is available for each method specifying the required parameters for the request and the expected format of the response. If you require additional calls please contact us at support@ikanow.com. Earlier versions of the API are archived [here](#).

JSONP is supported via the global URL parameter "jsonp=<callback-name>".

Next let's get started interacting with the Infnit.e API.

Quick links:

- [Knowledge management: querying the document, entity and event databases](#)
- [Authentication methods](#)
- [Source management](#)
- [Community methods](#)
- [Share methods](#)
- [People](#)
- [Custom](#)
- [GUI utilities](#)

Interacting with the API

Most methods in the API require a cookie that is granted during authentication to use. The logical flow of using the API includes a [Login call](#), Other API calls, and finally a [Logout call](#). See the examples for details on how to do each. A cookie is only valid for 30 minutes (or whatever the system administrator has configured) from your last API call, to keep the cookie alive using the [Keep Alive API call](#) will renew the 30 minute limit. Some methods do not require a cookie and can just be called directly.

Example use

See [these pages](#) for examples of how to use the API.

REST methods

The Infnit.e API includes the following RESTful methods to access Infnit.e capabilities.

Knowledge management: querying the document, entity and event databases

The most important API call performs searches, and ranks and aggregates [documents](#) and sub-objects like [entities](#), [associations](#), and [meta data](#). The link below comprises a number of pages discussing the configuration and output.

- [knowledge/document/query](#): performs a query and returns documents and metadata about the results

It is also possible to get individual documents (just metadata or metadata and content):

- [knowledge/document/get](#): returns a single document given its "_id" field or its "sourceKey"/"url" pair.
- [knowledge/document/file/get](#): returns a single document its "sourceKey" and relative path (fileshares only).

We call any element of the [document data model](#) that exists independently to the document a "feature" (including entities, geotags, entity associations). There are a number of methods to retrieve these features:

- [knowledge/feature/entitySuggest](#): given free-form text, suggests matching entities.
- [knowledge/feature/assocSuggest](#): given a semi-structured text input (under "subject"/"verb"/"object"), suggests matching entity associations.
- [knowledge/feature/aliasSuggest](#): given an entity name, provides all known aliases of that entity.

Authentication methods

With a few minor exceptions, it is necessary to log-in before invoking any other methods. Some methods require system-wide admin privileges, indicated below with "**(admin)**". Other methods need to be approved by the community owner before they take effect, these are indicated below with "**(community-approval)**". Others can only be performed by the community owner, the content owner, or an admin, these are indicated below with "**(owner/moderator/admin)**". It is assumed that any call that takes a community ID requires membership of that community, apart from requesting to join public communities.

- [auth/login](#): returns a cookie to make subsequent api calls after authenticating a user
- [auth/login/admin \(admin\)](#): returns an admin cookie to make subsequent admin api calls after authenticating an administrator
- [auth/keepalive](#): updates a cookie to last for 30 more minutes
- [auth/logout](#): removes a users cookie not allowing them to make api calls anymore
- [auth/forgotpassword](#): starts process to receive password via email
- [auth/deactivate \(admin\)](#): turns off a users account so they can no longer log in

Logging out is optional - users can only be logged-in from one place and the older session is automatically terminated when the newer one begins.

If an API key is specified (via [REST](#) or the [GUI](#)) then any API command can be authenticated with the URL parameter "infinite_api_key" instead of logging in. API keys can be used in cookies also, in the format "infinitecookie=api:API_KEY;".

Source management

These methods can be used to get information about the sources that Infnit.e harvests to populate its database (see [this overview](#)). This might be needed in order to specify the inputs over which a query is run, or to check the harvest status of a source (either regularly or based on unexpected results).

- [config/source/get](#): returns a source object given its "_id" field.
- [config/source/good](#): returns a list of all active sources.
- [config/source/user](#): returns a list of all sources for the current user or, if the user is an administrator, a list of all sources for the communities that the user is a member of.

These methods are more administrative:

- [config/source/bad](#): lists unapproved sources (also sources that have been turned off by the Infnit.e harvester due to issues in harvesting the source)
- [config/source/pending](#): lists proposed sources that a community administrator has not yet approved.

The process of configuring source objects, which is basically simple but allows for sophisticated functionality in return for more complexity, is described [here](#).

A key activity, either for Open Source Analysis or for mining an organizations internal data-stores, is creating source configuration objects. The link below comprises a number of pages describing the different possible configurations.

- [config/source/save](#): (**community-approval**) for new sources, (**owner/moderator/admin**) for existing sources. Adds a new source to the harvest list
- [config/source/add](#): (**deprecated**) (**community-approval**)
- [config/source/test](#): tests a new source in the harvester to compare results and tweak source setup

After a source has been added, a request is sent to the community administrator, who has the option to accept or decline it. Only accepted sources are processed by the Infnit.e harvester.

- [config/source/approve](#): approves a source so it will start being harvested
- [config/source/decline](#): denies a source so it will not be harvested

Admins and owners can remove documents from sources (or the sources themselves, which automatically removes the documents as well):

- [config/source/delete](#): (**owner/moderator/admin**) removes a source from harvesting
- [config/source/delete/docs](#): (**owner/moderator/admin**) removes specific harvested documents

Community methods

The Community API calls are used to manage communities (add, update, remove, and get). The APIs can be broken out into the following basic areas of functionality:

Basic community administration: APIs used to add a new community, remove an existing community, or update an existing community.

- [social/community/add](#): adds a new community
- [social/community/remove](#) (**owner/moderator/admin**): removes a community
- [social/community/update](#) (**owner/moderator/admin**): changes information about a community

A user can retrieve their list of communities (together with other user metadata) with the "person/get" API call:

- [social/person/get](#)

A user can also retrieve a list of all other users they are in communities with:

- [social/person/list](#)

Community membership: APIs used to join (request membership in) a community, leave a community, invite a user to join a community, update a user's status within a community (active/pending), update user type (owner, moderator, user), and manage responses to invitation.

- [social/community/member/join](#): (**community-approval**) attempts to join a community
- [social/community/member/leave](#): removes yourself from a community
- [social/community/member/invite](#): (**owner/moderator/admin community-approval**) invites a user to a community
- [social/community/member/update/status](#): (**owner/moderator/admin**) updates a members status in a group (active,disabled,pending)
- [social/community/member/update/type](#): (**owner/moderator/admin**) updates a members type in a group (owner,moderator,content_publisher,member)
- [social/community/requestresponse](#): respond to community invite/join requests

Get: APIs used to retrieve data for one or more communities:

- [social/community/get](#): retrieves a single community based on the community._id value
- [social/community/getpublic](#): retrieves all publicly available communities
- [social/community/getprivate](#): retrieves all private communities (including personal communities)
- [social/community/getall](#): retrieves all communities

- `social/community/getsystem`: retrieves the system community

(Note that wherever community approval or ownership is required, a system admin can also perform that task. In addition, community "moderators" can also add, remove, and update users.)

Share methods

The Share methods are used to allow the saving and sharing of queries, datasets, sources, documents, feeds, and other objects that can be encoded using standard JSON notation (or are arbitrary binary objects).

- `social/share/add/json`: saves a json object to the share database
- `social/share/update/json`: (**owner/moderator/admin**): replaces a current json object with a new one in the share database
- `social/share/remove`: (**owner/moderator/admin**): deletes an object from the share database
- `social/share/add/community`: allows a community to access a share
- `social/share/remove/community`: disallows a community to access a share
- `social/share/endorse`: (**moderator/admin**): adds or removes an `endorsed` tag for a share within a community
- `social/share/get`: returns a share by `_id`
- `social/share/search`: searches shares for ones you have access to
- `social/share/add/binary`: adds a binary file to the share db
- `social/share/update/binary`: (**owner/moderator/admin**): replaces a current binary object with a new one in the share database
- `social/share/add/ref`: creates a shared "pointer" to an existing Infinite artifact to the share database
- `social/share/update/ref`: (**owner/moderator/admin**): replaces a shared "pointer" to an existing Infinite artifact with a new one in the share database

People

The People methods are admin functions used by the 3rd party user management systems (eg Wordpress web-site for the Ikanow SaaS version) to register and update users. These calls must come from an authenticated IKANOW server or admins and will not work for other users.

- `social/person/register`: (**admin**): registers a new user in the system
- `social/person/update`: (**admin**): updates a user in the system
- `social/person/delete`: (**admin**): removes a user from the system
- `social/person/update/password`: updates a users password
- `social/person/update/email`: updates a users email address(es)

Custom

These custom methods are related to running and viewing custom jobs.

- `custom/mapreduce/schedulejob`: schedules a hadoop map reduce job to be ran in the future, possibly on a regular interval
- `custom/mapreduce/updatejob`: updates a scheduled hadoop job, can be used to rerun a job
- `custom/mapreduce/removejob`: deletes a job and its data
- `custom/mapreduce/getresults`: returns the results from a map reduce job
- `custom/mapreduce/getjobs`: returns a list of jobs the current user has access to
- `custom/savedquery/schedulejob`: schedules a query job to be ran in the future, possibly on a regular interval
- `custom/savedquery/updatejob`: updates a scheduled query job, can be used to rerun a job

GUI integration

The uisetup methods are GUI specific methods for searching, storing, and getting modules (widgets). Most API users will not need to use these methods

- `social/gui/uisetup/get`: returns a user's last saved uisetup including open widgets, last query, and advanced settings
- `social/gui/uisetup/update`: saves a users's uisetup so `/uisetup/get` will return it next time
- `social/gui/modules/get`: returns all widgets available to a user (for widget browser)
- `social/gui/modules/install`: Uploads or updates the metadata describing a widget (that allows it to be accessed from the widget browser)
- `social/gui/modules/delete`: Deletes all metadata describing a widget (ie the widget can no longer be accessed from the widget browser)
- `social/gui/modules/user/get`: returns a users last saved list of widgets (for widget browser)
- `social/gui/modules/user/set`: saves a users last set of widgets (for widget browser)
- `social/gui/modules/search`: searches all available widgets on title, author and description fields of widget

Extra Information

These pages may link to some useful extra information for using some of the api calls:

- [Overview of the query process.](#)
- [Overview of the source configuration process.](#)
- [Community ID regular expression usage:](#) Some functions allow sending regular expressions for the community ids, to make hand-crafted requests easier.

Graphical utilities

A number of common sets of API calls have simple web pages that allow them to be performed with directly touching the API:

- [Managing users](#) (part of the integrated management GUI)
- [Managing communities](#) (part of the integrated management GUI)
- [Uploading files to the shared storage](#)
- [Uploading widgets to the GUI framework](#)
- [Uploading Hadoop plugins and scheduling their execution](#)
- [Editing and monitoring sources](#) (part of the integrated management GUI)
- [Chrome Extension for fast source creation](#)

API Terms and Conditions

Now for the part we have to do. The Infinite API follows under the same terms and conditions as all other portions of our application. By using the Infnit.e APIs, you agree to these terms. If you disagree with any of these terms, IKANOW does not grant you a license to use the Infinite APIs. We reserve the right to update and change these terms from time to time without notice. We always have your most recent version of these terms available [here](#)

Your license to the Infinite APIs under these terms continues until it is terminated by either party. You may terminate the license by discontinuing use of all or any of the Infinite APIs. IKANOW reserves the right to terminate the license at any time for any reason. Your rights to use the Infnit.e APIs terminate automatically if (i) you violate any of these terms, (ii) IKANOW publicly posts a written notice of termination on infinite.ikanow.com, (iii) IKANOW sends a written notice of termination to you, or (iv) IKANOW disables access to the Infnit.e APIs to you.

API - Examples

This page shows examples of how to use the API in to perform simple tasks in Java, Actionsript and curl. Any language that can send HTTP request can consume the API, these are just some examples showing some common languages.

Note that there is a "beta" Java driver in the data model JAR ([github link](#)). It is beta in the sense that it is incomplete (ie some REST calls don't have Java functions) and that it is (currently) undocumented. All functions that are present are well-tested (they have been used in our own internal applications).

Example 0: Hashing a password

If we look at the [login page documentation](#), we see that the password needs to be SHA-256 encoded and a base-64 string.

Because logging in is a vital function, here is a Java, Actionsript, and Web example of getting your hashed password:
"If you are doing this correctly the password "12345" will hash to "WZRHGrSBEsr8wYFZ9sx0tPURuZgG2ImzyvWpwXPKz8U="

[Java - Hashing a Password](#)
[Actionsript - Hashing a Password](#)
[Web Browser - Hashing a Password](#)

Example 1: Sending an HTTP request and sending a request to Login

Because our API is REST based, the most important thing to do in any language is understand how to send HTTP Requests so you are able to call our system. Many languages may have multiple ways to do this so we will show an example of how you can send requests in Java, Actionsript, curl, and your browser.

[Java - HTTP Request](#)
[Actionsript - HTTP Request](#)
[cURL - HTTP Request](#)
[Web Browser - HTTP Request](#)

Once you have mastered how to send http requests, you can attempt to send requests to our API now. To login to the system we simply just need to send an http request to <http://infinite.ikanow.com/api/auth/login> with our username and password. On a successful login you will receive a cookie that will be active for 30 minutes from your last command. This cookie will need to be passed back anything you want to perform an api call that needs authentication. In the next section we show examples of how to send requests with cookies.

Example 2: Getting a cookie, and sending the cookie in a subsequent request (Login, Get Person, Logout).

To make requests to many of our services we require you to send a cookie along with the HTTP Request so we can verify who we are getting the information for. Below are examples of how to receive cookies from a Login call and then send the cookies to another call, then finally logging out.

[Java - Cookies - Login, Get Person, Logout](#)
[Actionsript - Cookies - Login, Get Person, Logout](#)
[cURL - Cookies - Login, Get Person, Logout](#)
[Web Browser - Cookies - Login, Get Person, Logout](#)

Beyond this point the examples are just showing added functionality. As a result only a single language will be used to demonstrate each example. Using the above principles in example 2 you can extend any language to do the tasks below.

Example 3: Login, Get Communities, Perform a Query, Logout

Performing a query adds a final bit of complexity in which we must send a POST request with a json object in the body.

[Actionscript - Login, Get Communities, Perform a Query, Logout](#)

Example 4: Login, Create a Community, Invite User, Logout

[Java - Login, Create a Community, Invite User, Logout](#)

Note that the Java example above includes an example of how to use the Data Model library to deserialize from JSON strings into Infinite data store (or API) objects:

Using the ResponsePojo utility code

```
// Single object:
ResponsePojo response = ResponsePojo.fromApi(communityresult, ResponsePojo.class,
CommunityPojo.class, new CommunityPojoApiMap());
// (use null instead of "new CommunityPojoApiMap()" for objects derived from
BaseApiPojo)
CommunityPojo community = (CommunityPojo)response.getData();
// Multiple objects:
ResponsePojo response = ResponsePojo.listFromApi(communityresult, ResponsePojo.class,
CommunityPojo.listType(), new CommunityPojoApiMap());
List<CommunityPojo> communities = (List<CommunityPojo>)response.getData();
```

Example 5: Login, Get Public Communities, Request to Join Community, Logout

[cURL - Login, Get Public Communities, Request to Join Community, Logout](#)

Example 6: Login, Get a Source, Edit the Source, Test the new Source, Save the new Source, Logout

[Actionscript - Login, Get a Source, Edit the Source, Test the new Source, Save the new Source, Logout](#)

Example 7: Login, share a hadoop Jar file, schedule a map reduce job with jar, Logout

[Java - Login, share a hadoop Jar file, schedule a map reduce job with jar, Logout](#)

Note that unlike example 4, this Java example uses the simpler ResponsePojo "fromApi" call for when the "data" field from the API call is just a string:

Simpler ResponsePojo usage

```
ResponsePojo response = ResponsePojo.fromApi(schedulejobresult, ResponsePojo.class);
String responseData = (String) response.getData();
```

Example 8: Login, Get Map Reduce Jobs, Get Results of a map reduce job, Logout

[cURL - Login, Get Map Reduce Jobs, Get Results of a map reduce job, Logout](#)

[Actionscript - Cookies - Login, Get Person, Logout](#)

Actionscript - Login for cookie, send cookie to get/person, logout

```
protected function button_clickHandler( event:MouseEvent ):void
{
    //set up url
    var username:String = "sterling_archer@ikanow.com";
    var hashedpassword:String = "WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPkz8U%3D";
    //don't forget to URLEncode your arguments
    var address:String = "http://infinite.ikanow.com/api/auth/login/" + username + "/"
+ hashedpassword;

    //send login request
    var httpService:HTTPService = new HTTPService();
    httpService.addEventListener( ResultEvent.RESULT, httpResultHandler );
    httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler );
    httpService.url = address;
    httpService.send();
}

protected function httpResultHandler( event:ResultEvent ):void
{
    //convert result json string to an as3 object using the as3corelib library
    //available at: https://github.com/mikechambers/as3corelib
    var jsonObject:Object = JSON.decode( event.result.toString(), true );

    if ( jsonObject.response.success == true )
    {
        //successfully logged in, cookies are automatically stored in flash so no need
to handle them
        //send request for our person object (will print out in httpPersonResultHandler)
        var httpService:HTTPService = new HTTPService();
        httpService.addEventListener( ResultEvent.RESULT, httpPersonResultHandler );
        httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler );
        httpService.url = "http://infinite.ikanow.com/api/social/person/get";
        httpService.send();
    }
}

protected function httpPersonResultHandler( event:ResultEvent ):void
{
    //just print out person result json
    Alert.show( event.result.toString() );
    //logout
    var httpService:HTTPService = new HTTPService();
    httpService.url = "http://infinite.ikanow.com/api/auth/logout";
    httpService.send();
}

protected function httpFaultHandler( event:FaultEvent ):void
{
    Alert.show( "Http Request had an error: " + event.message );
}
```

Actionscript - Hashing a Password

```
//The Crypto class can be found in the as3crypto library
//the library can be found at: http://code.google.com/p/as3crypto/
var password:String = "12345";
var c:Crypto = new Crypto();
var cipher:IHash = Crypto.getHash( "sha256" );
var data:ByteArray = Hex.toArray( Hex.fromString( password ) );
var hashed:String = Base64.encodeByteArray( cipher.hash( data ) );
Alert.show( hashed );
```

Actionscript - HTTP Request

Actionscript - HTTP Request

```
protected function button_clickHandler( event:MouseEvent ):void
{
    //set up url
    var username:String = "sterling_archer@ikanow.com";
    var hashedpassword:String = "WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D";
    //don't forget to URLEncode your arguments
    var address:String = "http://infinite.ikanow.com/api/auth/login/" + username + "/"
+ hashedpassword;

    //send request
    var httpService:HTTPService = new HTTPService();
    httpService.addEventListener( ResultEvent.RESULT, httpResultHandler );
    httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler );
    httpService.url = address;
    httpService.send();
}

protected function httpFaultHandler( event:FaultEvent ):void
{
    Alert.show( "Http Request had an error: " + event.message );
}

protected function httpResultHandler( event:ResultEvent ):void
{
    //print out response json
    Alert.show( event.result.toString() );
}
```

Actionscript - Login, Get a Source, Edit the Source, Test the new Source, Save the new Source, Logout

Actionscript - Login, Get a Source, Edit the Source, Test the new Source, Save the new Source, Logout

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
```

```

xmlns:s="library://ns.adobe.com/flex/spark"
xmlns:mx="library://ns.adobe.com/flex/mx"
minWidth="955"
minHeight="600">

<fx:Script>
  <![CDATA[
    import com.adobe.serialization.json.JSON;
    import mx.controls.Alert;
    import mx.rpc.events.FaultEvent;
    import mx.rpc.events.ResultEvent;
    import mx.rpc.http.HTTPService;

    private var source:Object;

    protected function button1_clickHandler( event:MouseEvent ):void
    {
      //LOGIN
      //set up url
      var username:String = "sterling_archer@ikanow.com";
      var hashedpassword:String =
"WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D"; //don't forget to URLEncode your
arguments
      var address:String = "http://infinite.ikanow.com/api/auth/login/" +
username + "/" + hashedpassword;

      //send login request
      var httpService:HTTPService = new HTTPService();
      httpService.addEventListener( ResultEvent.RESULT,
httpLoginResultHandler );
      httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler );
      httpService.url = address;
      httpService.send();
    }

    protected function httpFaultHandler( event:FaultEvent ):void
    {
      Alert.show( "Http Request had an error: " + event.message );
    }

    protected function httpLoginResultHandler( event:ResultEvent ):void
    {
      //convert result json string to an as3 object using the as3corelib
library
      //available at: https://github.com/mikechambers/as3corelib
      var jsonObject:Object = JSON.decode( event.result.toString(), true );

      if ( jsonObject.response.success == true )
      {
        //this is a random microsoft source
        //you can get a source from config/source/good
        var sourceID:String = "4ef498756a068f1eeel7fac4";

        //successfully logged in, cookies are automatically stored in
flash so no need to handle them
        //send request for our person object
        var httpService:HTTPService = new HTTPService();
        httpService.addEventListener( ResultEvent.RESULT,
httpSourceResultHandler );

```

```

        httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler
    );

        httpService.url =
"http://infinite.ikanow.com/api/config/source/get/" + sourceID;
        httpService.send();
    }
}

protected function httpSourceResultHandler( event:ResultEvent ):void
{
    //convert result json string to an as3 object using the as3corelib
library
    //available at: https://github.com/mikechambers/as3corelib
    var jsonObject:Object = JSON.decode( event.result.toString(), true );

    if ( jsonObject.response.success == true )
    {
        //here we will edit the source to try something new
        source = jsonObject.data;
        source.url = "http://rss.cnn.com/rss/cnn_topstories.rss";
        source.title = "CNN top stories";
        source.description = "CNNs top news stories around the world";
        //lets removes some of the fields we dont need anymore
        delete source._id;
        delete source.created;
        delete source.modified;
        delete source.harvest;
        delete source.shah256Hash;
        delete source.ownerId;
        delete source.key;
        delete source.tags;
        delete source.harvestBadSource;

        //now we will test our new source to see if the docs it brings
back are what we want
        var httpService:HTTPService = new HTTPService();
        httpService.addEventListener( ResultEvent.RESULT,
httpSourceTestHandler );
        httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler
    );

        httpService.method = "POST";
        httpService.url =
"http://infinite.ikanow.com/api/config/source/test?numReturn=1&returnFullText=false";
        //to send a post we just put the data in the send handler
        //we are posting the modified source we created and set the
content type
        httpService.contentType = "application/json";
        httpService.send( JSON.encode( source ) );
    }
}

protected function httpSourceSaveHandler( event:ResultEvent ):void
{
    //convert result json string to an as3 object using the as3corelib
library
    //available at: https://github.com/mikechambers/as3corelib
    var jsonObject:Object = JSON.decode( event.result.toString(), true );

    if ( jsonObject.response.success == true )

```



```

        {
            //dont forget to logout
            var httpService:HTTPService = new HTTPService();
            httpService.url = "http://infinite.ikanow.com/api/auth/logout";
            httpService.send();
        }
    }

    protected function httpSourceTestHandler( event:ResultEvent ):void
    {
        //convert result json string to an as3 object using the as3corelib
library
        //available at: https://github.com/mikechambers/as3corelib
        var jsonObject:Object = JSON.decode( event.result.toString(), true );

        if ( jsonObject.response.success == true )
        {
            //if we like how the source was harvest in jsonObject.data
            //we can save our source back to the server
            //you can get community ids from person/get
            var communityID:String = "4c927585d591d31d7b37097a";
            var httpService:HTTPService = new HTTPService();
            httpService.addEventListener( ResultEvent.RESULT,
httpSourceSaveHandler );
            httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler
);

            httpService.method = "POST";
            httpService.url =
"http://infinite.ikanow.com/api/config/source/save/" + communityID;
            //to send a post we just put the data in the send handler
            //we are posting the modified source we created and set the
content type

            httpService.contentType = "application/json";
            httpService.send( JSON.encode( source ) );
        }
    }
    ]]>
</fx:Script>

<fx:Declarations>
    <!-- Place non-visual elements (e.g., services, value objects) here -->
</fx:Declarations>
<fx:Script>
    <![CDATA[

        ]]>
</fx:Script>
<s:Button
    label="Push Me"
    click="button1_clickHandler(event)" />

```

```
</s:Application>
```

Actionscript - Login, Get Communities, Perform a Query, Logout

Actionscript - Login, Get Communities, Perform a Query, Logout

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx"
  minWidth="955"
  minHeight="600">

  <fx:Script>
    <![CDATA[
      import com.adobe.serialization.json.JSON;
      import mx.controls.Alert;
      import mx.rpc.events.FaultEvent;
      import mx.rpc.events.ResultEvent;
      import mx.rpc.http.HTTPService;

      protected function button1_clickHandler( event:MouseEvent ):void
      {
        //LOGIN
        //set up url
        var username:String = "sterling_archer@ikanow.com";
        var hashedpassword:String =
"WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D"; //don't forget to URLEncode your
arguments
        var address:String = "http://infinite.ikanow.com/api/auth/login/" +
username + "/" + hashedpassword;

        //send login request
        var httpService:HTTPService = new HTTPService();
        httpService.addEventListener( ResultEvent.RESULT,
httpLoginResultHandler );
        httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler );
        httpService.url = address;
        httpService.send();
      }

      protected function httpFaultHandler( event:FaultEvent ):void
      {
        Alert.show( "Http Request had an error: " + event.message );
      }

      protected function httpLoginResultHandler( event:ResultEvent ):void
      {
        //convert result json string to an as3 object using the as3corelib
library
        //available at: https://github.com/mikechambers/as3corelib
        var jsonObject:Object = JSON.decode( event.result.toString(), true );

        if ( jsonObject.response.success == true )
```

```

        {
            //successfully logged in, cookies are automatically stored in
            flash so no need to handle them
            //send request for our person object
            var httpService:HTTPService = new HTTPService();
            httpService.addEventListener( ResultEvent.RESULT,
            httpPersonResultHandler );
            httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler
            );

            httpService.url =
            "http://infinite.ikanow.com/api/social/person/get";
            httpService.send();
        }
    }

    protected function httpPersonResultHandler( event:ResultEvent ):void
    {
        //convert result json string to an as3 object using the as3corelib
        library
        //available at: https://github.com/mikechambers/as3corelib
        var jsonObject:Object = JSON.decode( event.result.toString(), true );

        if ( jsonObject.response.success == true )
        {
            //here we will grab the communities so we can send them in a query
            var communities:Array = jsonObject.data.communities;
            //now we need to turn the community ids into a comma delimited
            string
            //you do not need to send all communities, just the ones you want
            to query on

            var communityString:String = communities[ 0 ]._id;

            for ( var i:int = 1; i < communities.length; i++ )
            {
                communityString += "," + communities[ i ]._id;
            }

            //create a json object for the query, actionsript objects
            //can be converted into json so we can just create an anonymous
            object

            var queryData:Object = new Object();
            queryData.qt = new Array();
            var queryTerm:Object = new Object();
            queryTerm[ "ftext" ] = "sterling archer";
            queryData.qt.push( queryTerm );

            //send request for query
            var httpService:HTTPService = new HTTPService();
            httpService.addEventListener( ResultEvent.RESULT,
            httpQueryResultHandler );
            httpService.addEventListener( FaultEvent.FAULT, httpFaultHandler
            );

            httpService.url =
            "http://infinite.ikanow.com/api/knowledge/document/query/" + communityString;
            //to send a post we just put the data in the send handler
            httpService.send( JSON.encode( queryData ) );
        }
    }
}

```

library

```
protected function httpQueryResultHandler( event:ResultEvent ):void
{
    //convert result json string to an as3 object using the as3corelib
    //available at: https://github.com/mikechambers/as3corelib
    var jsonObject:Object = JSON.decode( event.result.toString(), true );

    if ( jsonObject.response.success == true )
    {
        //now you can do something with the results of query
        //say get every title?
        var titles:String = "";

        for each ( var doc:Object in jsonObject.data )
        {
            titles += doc.title.toString() + "\n";
        }
        Alert.show( titles );

        //dont forget to logout
        var httpService:HTTPService = new HTTPService();
        httpService.url = "http://infinite.ikanow.com/api/auth/logout";
        httpService.send();
    }
}
]]>
</fx:Script>

<fx:Declarations>
    <!-- Place non-visual elements (e.g., services, value objects) here -->
</fx:Declarations>
<fx:Script>
    <![CDATA[

        ]]>
</fx:Script>
<s:Button
```

```
        label="Push Me"
        click="button1_clickHandler(event)" />
</s:Application>
```

cURL - Cookies - Login, Get Person, Logout

cURL - Login for cookie, send cookie to get/person, logout

```
curl -c cookies.txt
http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ9
sx0tPURuZgG2lmzyvWpwXPKz8U%3D
curl -b cookies.txt http://infinite.ikanow.com/api/social/person/get
curl -b cookies.txt http://infinite.ikanow.com/api/auth/logout
```

cURL - HTTP Request

cURL - HTTP Request

```
curl
http://infinite.ikanow.com/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ9sx0t
PURuZgG2lmzyvWpwXPKz8U%3D
```

cURL - Login, Get Map Reduce Jobs, Get Map Reduce Job Results, Logout

cURL - Login, Get Map Reduce Jobs, Get Map Reduce Job Results, Logout

```
curl -c cookies.txt
'http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ
9sx0tPURuZgG2lmzyvWpwXPKz8U%3D' > response.txt
curl -b cookies.txt 'http://infinite.ikanow.com/api/custom/mapreduce/getjobs' >
response.txt
    //copy over the job id that we want to see results for
curl -b cookies.txt
'http://infinite.ikanow.com/api/custom/mapreduce/getresults/abcde12345' > response.txt
    //the results of the map reduce job will be in the data field of the response
curl -b cookies.txt 'http://infinite.ikanow.com/api/auth/logout' > response.txt
```

cURL - Login, Get Public Communities, Request to Join Community, Logout

cURL - Login, Get Public Communities, Request to Join Community, Logout

```
curl -c cookies.txt
'http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ
9sx0tPURuZgG2lmzyvWpwXPKz8U%3D' > response.txt
curl -b cookies.txt 'http://infinite.ikanow.com/api/social/community/getpublic' >
response.txt
//From the response of getpublic communities we can copy one of the ids to what
community we wish to join
curl -b cookies.txt
'http://infinite.ikanow.com/api/social/community/member/join/4c927585d591d31d7c37097b'
> response.txt
//response will say if you have been successfully added, or awaiting owner permission
curl -b cookies.txt 'http://infinite.ikanow.com/api/auth/logout' > response.txt

*note: All of these curl calls are pushing their output to a local file named
response.txt, you can remove this if you want to just print out the result in console
```

Java - Cookies - Login, Get Person, Logout

Java - Login for cookie, send cookie to get/person call, logout

```
public static void main(String[] args) throws Exception
{
    //don't forget to URLEncode your arguments
    String address =
"http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ
9sx0tPURuZgG2lmzyvWpwXPKz8U%3D";
    String loginresult = sendRequest(address);
    //Our data objects can be used by importing infinit.e.data_model.jar and gson.jar
    ResponsePojo response = ResponsePojo.fromApi(loginresult, ResponsePojo.class);
    if ( response.getResponse().isSuccess() )
    {
        //send next request
        String personAddress = "http://infinite.ikanow.com/api/social/person/get";
        String personresult = sendRequest(personAddress);
        //We need to convert the result object into a response with a person object in
it
        response = ResponsePojo.fromApi(personresult, ResponsePojo.class,
PersonPojo.class, new PersonPojoApiMap());
        if ( response.getResponse().isSuccess() )
        {
            PersonPojo personResult = (PersonPojo)response.getData();
            System.out.println(personResult.getEmail());
        }
    }
    else
    {
        System.out.println("error logging in: " +
response.getResponse().getMessage());
    }
    //logout when we are done, this will deactivate our cookie
    sendRequest("http://infinite.ikanow.com/api/auth/logout");
}
```

```

}

private static String cookie = null;
public static String sendRequest(String urlAddress ) throws Exception
{
    URL url = new URL(urlAddress);
    URLConnection urlConnection = url.openConnection();
    if ( cookie != null ) //add cookie to request if we have one
        urlConnection.setRequestProperty("Cookie", cookie);
    ((HttpURLConnection)urlConnection).setRequestMethod("GET");

    //read back result
    BufferedReader inStream = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
    StringBuilder strBuilder = new StringBuilder();
    String buffer;
    while ( (buffer = inStream.readLine()) != null )
    {
        strBuilder.append(buffer);
    }
    inStream.close();

    //save cookie if cookie is null
    if ( cookie == null )
    {
        String headername;
        for ( int i = 1; (headername = urlConnection.getHeaderFieldKey(i)) != null;
i++ )
        {
            if ( headername.equals("Set-Cookie") )
            {
                cookie = urlConnection.getHeaderField(i);
                break;
            }
        }
    }
}

```

```
    }  
    return stringBuilder.toString();  
}
```

Java - Hashing a Password

Java - Hashing a Password

```
//In this example we use the apache commons codec library to convert our hashed  
password to base 64  
//the library can be found at: http://commons.apache.org/codec/download\_codec.cgi  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import org.apache.commons.codec.binary.Base64;  
public class hashpassword  
{  
    public static void main(String[] args) throws NoSuchAlgorithmException  
    {  
        String password = "12345";  
        MessageDigest digest = MessageDigest.getInstance("SHA-256");  
        byte[] hashbytes = digest.digest(password.getBytes());  
        String hashpassword = Base64.encodeBase64String(hashbytes);  
        System.out.println(hashpassword);  
    }  
}
```

Java - HTTP Request

Java - HTTP Request

```
//set up url
String username = "sterling_archer@ikanow.com";
String hashedpassword = "WZRHGrsBESr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D"; //don't
forget to URLEncode your arguments
String address = "http://infinite.ikanow.com/api/auth/login/" + username + "/" +
hashedpassword;

//send request
URL url = new URL(address);
URLConnection urlConnection = url.openConnection();
((HttpURLConnection)urlConnection).setRequestMethod("GET");

//read back result
BufferedReader inStream = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
StringBuilder strBuilder = new StringBuilder();
String buffer;
while ( (buffer = inStream.readLine()) != null )
{
    strBuilder.append(buffer);
}
inStream.close();

//print out response json
System.out.println(strBuilder.toString());
```

Java - Login, Create a Community, Invite User, Logout

Java - Login, Create a Community, Invite User, Logout

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;

import com.ikanow.infinite.data_model.api.ResponsePojo;
import com.ikanow.infinite.data_model.api.social.community.CommunityPojoApiMap;
import com.ikanow.infinite.data_model.store.social.community.CommunityPojo;

public class test
{
    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception
    {
        //login
```

```

        String address =
"http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ
9sx0tPURuZgG2lmzyvWpwXPKz8U%3D"; //don't forget to URLEncode your arguments
        String loginresult = sendRequest(address);
        ResponsePojo response = ResponsePojo.fromApi(loginresult, ResponsePojo.class);
        if ( response.getResponse().isSuccess() )
        {
            //create a community
            String communityName = "testcommunity4";
            String communityDesc = "some long desc";
            String communityTags = "testtag1,testtag2";
            String createCommunityAddress =
"http://infinite.ikanow.com/api/social/community/add/" +
URLEncoder.encode(communityName,"UTF-8") + "/" +
            URLEncoder.encode(communityDesc,"UTF-8") + "/" +
URLEncoder.encode(communityTags,"UTF-8");
            String communityresult = sendRequest(createCommunityAddress);
            //We need to convert the result object into a response with a person
object in it
            response = ResponsePojo.fromApi(communityresult, ResponsePojo.class,
CommunityPojo.class, new CommunityPojoApiMap());
            if ( response.getResponse().isSuccess() )
            {
                CommunityPojo communityResult = (CommunityPojo)response.getData();
                String personID = "abcde12345"; //some other users id that you want to
invite
                //now invite a user
                //to use ids, you need to import a mongo.jar available at
https://github.com/mongodb/mongo-java-driver/downloads
                String inviteUserAddress =
"http://infinite.ikanow.com/api/social/community/member/invite/" +
communityResult.getId().toString() + "/" + personID;
                String inviteresult = sendRequest(inviteUserAddress);
                response = ResponsePojo.fromApi(inviteresult, ResponsePojo.class);
                if ( response.getResponse().isSuccess() )
                {
                    //invited user successfully! now we are done
                }
            }
        }
        else
        {
            System.out.println("error logging in: " +
response.getResponse().getMessage());
        }
        //logout when we are done, this will deactivate our cookie
        sendRequest("http://infinite.ikanow.com/api/auth/logout");
    }

    private static String cookie = null;
    public static String sendRequest(String urlAddress ) throws Exception
    {
        URL url = new URL(urlAddress);
        URLConnection urlConnection = url.openConnection();
        if ( cookie != null )
            urlConnection.setRequestProperty("Cookie", cookie);
        ((HttpURLConnection)urlConnection).setRequestMethod("GET");

        //read back result

```

```
        BufferedReader inStream = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
        StringBuilder strBuilder = new StringBuilder();
        String buffer;
        while ( (buffer = inStream.readLine()) != null )
        {
            strBuilder.append(buffer);
        }
        inStream.close();

        //save cookie if cookie is null
        if ( cookie == null )
        {
            String headername;
            for ( int i = 1; (headername = urlConnection.getHeaderFieldKey(i)) !=
null; i++ )
            {
                if ( headername.equals("Set-Cookie") )
                {
                    cookie = urlConnection.getHeaderField(i);
                    break;
                }
            }
        }
        return strBuilder.toString();
    }
}
```

```
}
```

Java - Login, share a hadoop Jar file, schedule a map reduce job with jar, Logout

Java - Login, share a hadoop Jar file, schedule a map reduce job with jar, Logout

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;

import com.ikanow.infinite.data_model.api.ResponsePojo;

public class test
{
    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception
    {
        //login
        String address =
"http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ
9sx0tPURuZgG2lmzyvWpwXPKz8U%3D"; //don't forget to URLEncode your arguments
        String loginresult = sendRequest(address, null);
        ResponsePojo response = ResponsePojo.fromApi(loginresult, ResponsePojo.class);
        if ( response.getResponse().isSuccess() )
        {
            //get map reduce file bytes, send in POST to add binary call
            String mapreduceFile = "C:/Users/Burch/Desktop/mapreducefile.jar";
            byte[] fileBytes = getBytesFromFile(new File(mapreduceFile));
            String shareTitle = "Archer Sum Map Reduce";
            String shareDesc = "Sums up the sources by community, by Sterling Archer";
            String createCommunityAddress =
"http://infinite.ikanow.com/api/social/share/add/binary/" +
                URLEncoder.encode(shareTitle,"UTF-8") + "/" +
                URLEncoder.encode(shareDesc,"UTF-8");
            String communityresult = sendRequest(createCommunityAddress, fileBytes);
            //The response returns only the id of the share
            response = ResponsePojo.fromApi(communityresult, ResponsePojo.class);
            if ( response.getResponse().isSuccess() )
            {
                String shareID = (String)response.getData();
            }
        }
    }
}
```

```

        String jobTitle = "Test Job";
        String jobDesc = "testing a mr job";
        String communityIds = "4c927585d591d31d7b37097a"; //can get
communities from social/person/get
        String jarURL = "$infinite/share/get/" + shareID; //the web url of the
share we just added $infinite will be converted to the local api serverside
        String timeToRun = "0"; //if we want this job to run in the future we
will put the ms time to run at
        String frequencyToRun = "NONE"; //see api page for available enums, we
only want it to run once so we put NONE
        String mapperClass =
"com.ikanow.infinite.e.core.mapreduce.examplejars.Test%24TokenizerMapper"; //package
name and class of mapper
        String reducerClass =
"com.ikanow.infinite.e.core.mapreduce.examplejars.Test%24IntSumReducer"; //package name
and class of reducer
        String combinerClass =
"com.ikanow.infinite.e.core.mapreduce.examplejars.Test%24IntSumReducer"; //this can be
the same as reducer usually
        String query = "null"; //we want everything so we can put null,
otherwise you can put json mongodb query
        String inputcollection = "DOC_METADATA"; //see API for what is
available

        String outputKey = "org.apache.hadoop.io.Text";
        String outputValue = "org.apache.hadoop.io.IntWritable";
        //now we want to schedule a map reduce job with this jar
        String scheduleJobAddress =
"http://infinite.ikanow.com/api/custom/mapreduce/schedulejob/" +
        URLEncoder.encode(jobTitle,"UTF-8") + "/" +
URLEncoder.encode(jobDesc,"UTF-8") + "/" +
        URLEncoder.encode(communityIds,"UTF-8") + "/" +
URLEncoder.encode(jarURL,"UTF-8") + "/" +
        URLEncoder.encode(timeToRun,"UTF-8") + "/" +
URLEncoder.encode(frequencyToRun,"UTF-8") + "/" +
        URLEncoder.encode(mapperClass,"UTF-8") + "/" +
URLEncoder.encode(reducerClass,"UTF-8") + "/" +
        URLEncoder.encode(combinerClass,"UTF-8") + "/" +
URLEncoder.encode(query,"UTF-8") + "/" +
        URLEncoder.encode(inputcollection,"UTF-8") + "/" +
URLEncoder.encode(outputKey,"UTF-8") + "/" +
        URLEncoder.encode(outputValue,"UTF-8");

        //to use ids, you need to import a mongo.jar available at
https://github.com/mongodb/mongo-java-driver/downloads

        String schedulejobresult = sendRequest(scheduleJobAddress, null);
        response = ResponsePojo.fromApi(schedulejobresult,
ResponsePojo.class);
        if ( response.getResponse().isSuccess())
        {
            //scheduled job successfully, get results ID, will be available
once job has finished running
            String mapreduceResultID = (String) response.getData();
            System.out.println(mapreduceResultID);
        }
    }
}
else

```

```

        {
            System.out.println("error logging in: " +
response.getResponse().getMessage());
        }
        //logout when we are done, this will deactivate our cookie
        sendRequest("http://infinite.ikanow.com/api/auth/logout", null);
    }

    private static String cookie = null;
    public static String sendRequest(String urlAddress, byte[] postData ) throws
Exception
    {
        URL url = new URL(urlAddress);
        URLConnection urlConnection = url.openConnection();
        if ( cookie != null )
            urlConnection.setRequestProperty("Cookie", cookie);
        if ( postData != null )
        {
            urlConnection.setDoOutput(true);
            urlConnection.setRequestProperty("Accept-Charset", "UTF-8");
            urlConnection.setRequestProperty("Content-Type",
"application/java-archive" + ";charset=" + "UTF-8");
            ((URLConnection)urlConnection).setRequestMethod("POST");
            OutputStream output = urlConnection.getOutputStream();
            output.write(postData);
            output.close();
        }
        else
        {
            ((URLConnection)urlConnection).setRequestMethod("GET");
        }

        //read back result
        BufferedReader inStream = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
        StringBuilder strBuilder = new StringBuilder();
        String buffer;
        while ( (buffer = inStream.readLine()) != null )
        {
            strBuilder.append(buffer);
        }
        inStream.close();

        //save cookie if cookie is null
        if ( cookie == null )
        {
            String headername;
            for ( int i = 1; (headername = urlConnection.getHeaderFieldKey(i)) !=
null; i++ )
            {
                if ( headername.equals("Set-Cookie") )
                {
                    cookie = urlConnection.getHeaderField(i);
                    break;
                }
            }
        }
        return strBuilder.toString();
    }

```

```
}

/**
 * from: http://www.exampledepot.com/egs/java.io/file2bytearray.html
 *
 * @param file
 * @return
 * @throws IOException
 */
public static byte[] getBytesFromFile(File file) throws IOException
{
    InputStream is = new FileInputStream(file);

    // Get the size of the file
    long length = file.length();

    // You cannot create an array using a long type.
    // It needs to be an int type.
    // Before converting to an int type, check
    // to ensure that file is not larger than Integer.MAX_VALUE.
    if (length > Integer.MAX_VALUE) {
        // File is too large
    }

    // Create the byte array to hold the data
    byte[] bytes = new byte[(int)length];

    // Read in the bytes
    int offset = 0;
    int numRead = 0;
    while (offset < bytes.length
        && (numRead=is.read(bytes, offset, bytes.length-offset)) >= 0) {
        offset += numRead;
    }

    // Ensure all the bytes have been read in
    if (offset < bytes.length) {
        throw new IOException("Could not completely read file "+file.getName());
    }

    // Close the input stream and return bytes
    is.close();
    return bytes;
}
```

```
}
```

Web Browser - Cookies - Login, Get Person, Logout

Web Browser- Login for cookie, send cookie to get/person, logout

1. In the url bar of your favorite web browser just type the url you want to navigate to and push go:
`http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D`
2. In the url bar of your favorite web browser just type the url you want to navigate to and push go: `http://infinite.ikanow.com/api/social/person/get`
3. In the url bar of your favorite web browser just type the url you want to navigate to and push go: `http://infinite.ikanow.com/api/auth/logout`

Web Browser - Hashing a Password

1. Point your favorite browser to: `http://insidepro.com/hashe.php?lang=eng`
2. Type your password in the password text input at the top of the page.
3. Click 'Generate' next to the password text input at the top of the page.
4. Scroll down to the **SHA-256** section, the Base64 encoded password will be in the 2nd textbox in the **SHA-256** section (the textbox directly above the [1] footnote).

SHA-224(HMAC)	[1] fae7e6777fe7919acef30037f7f6ee232fcbeb3d87983f468dd008de [2] +ufmd3/nkZrO8wA39/buIy/L6z2HmD9GjdAI3g== [1] [2]
SHA-256	5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173ca WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U= [1] 1445217533e7d4d0cffd9109c4edb60d62a02c0f0de9537be44f5e00d34 [2] FEUhdTPn1NDP/ZEJxO22DWKgLA8N6VN75E9eANNI608= [1] [2]
SHA-256(Django)	sha256\$\$5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf

Web Browser - HTTP Request

Web Browser - HTTP Request

1. In the url bar of your favorite web browser just type the url you want to navigate to and push go:
`http://infinite.ikanow.com/api/auth/login/sterling_archer@ikanow.com/WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D`

Auth - Deactivate

`/auth/deactivate/{username}/{password}`

 Turns off the given users account so they can no longer log in.

Authentication

Required, see [Auth - Login](#), must come from ikanow.com website on Saas, or must be admin on deployed version

Parameters

user (required)
username of account you want to deactivate

aduser (required)
admin username required to login

adpass (required)
admin password required to login

Example

<http://infinite.ikanow.com/api/auth/deactivate?user=bob@ikanow.com&aduser=admin@ikanow.com&adpass=FyrA89j8fR5BJekoKdGD%2Db83hV3sj48H4wAu3DR6c%3D>

Example Response



```
{
  response: {
    action: "Deactivate Account"
    success: true
    message: "Account deactivated successfully"
    time: 91
  }
}
```

Auth - Forgot Password

</auth/forgotpassword?username={user}&pass={password}>



A 2 stage process to reset a user's password:

- Firstly, when only the username is specified, an email is sent to the specified user's registered email account. This email contains a link to the "forgotpassword" REST call, except contain the user's current password (hashed)
- Secondly, when called with both username and password, generates a random password for the user and sends that in an email to the user's registered address.

The "temporary" password can then be used to set the desired password via the [Update Password](#) call.

Authentication

As described above, can be called without authentication, in order to send an email containing a "reset password" link to the user's registered address; or with authentication (password specified, not logged in normally ie with a cookie) to reset the user's password to a random string (again sent by email to the user's address).

Arguments

- username (required) - the username to reset.
- password (optional) - if *not* present, will just send an email containing a link to the REST calling containing the current (hashed) password; if present, will reset the user's password and send a link containing that new password.

Example

Stage 1

<http://infinite.ikanow.com/api/auth/forgotpassword?username=user@ikanow.com>
Stage 2

<http://infinite.ikanow.com/api/auth/forgotpassword?username=user@ikanow.com&password=WZRHGrSBEsr8wYFZ9sx0tPURuZgG2lmzyvWpwXPKz8U%3D>

Example Response



- Stage 1:

```
{
  response: {
    action: "Reset Password"
    success: true
    message: "Email has been sent containing link to reset
password."
    time: 91
  }
}
```

- Stage 2:

```
{
  response: {
    action: "Reset Password"
    success: true
    message: "Password reset successfully, new password has been
emailed to user."
    time: 91
  }
}
```

Auth - Keep Alive

/auth/keepalive



Updates a users cookie to extend session for another 30 minutes from now. Used to keep a session alive. Will return false if a session has already timed out or is logged out (either manually or because the same user has logged in elsewhere)

Authentication

Required, see [Auth - Login](#)

Arguments

none

Example

<http://infinite.ikanow.com/api/auth/keepalive>

Example Response



```
{
  response: {
    action: "Keep Alive"
    success: true
    message: "Cookie kept alive, 15min left."
    time: 35
  }
}
```

Auth - Login

`/auth/login/{user}/{pass}&returnurl={url}`

`/auth/login/?username={user}&password={pass}&returnurl={url}`

`/auth/login/{user}/{pass}&override=false`

`/auth/login/{user}/{pass}&multi=true (admin only)`



Authenticates a session for the current user/environment. This will return a cookie that is active for 30m from last action.

If an API key is specified (via [REST](#) or the [GUI](#)) then any API command can be authenticated with the URL parameter "infinite_api_key" instead of logging in. API keys can be used in cookies also, in the format "infinitecookie=api:API_KEY;"

Authentication

Not required (this call provides authorization)

Arguments

user (required)

username required to login, usually in the format of bob@ikanow.com

pass (required)

password required to login, encrypted with [SHA-256 \(base 64\)](#) and URL-encoded

returnurl (optional)

after login browser will redirect to this location

override (optional)

If set to false or "0" (true is default), then login will fail if user is already logged in elsewhere (default is that the other location's login will be invalidated)

multi (optional, *admin only*)

If set to true or "1" (false is default), then multiple logins are allowed.



if a returnurl is supplied, &success=true/false will be appended to the url string to indicate success or failure

Example

<http://infinite.ikanow.com/api/auth/login/bob@ikanow.com/7nrsLRbgCQOZEpmJclExmK5vRVN9%2FgeopluojZBFmNU%3B>

<http://infinite.ikanow.com/api/auth/login?username=bob@ikanow.com&password=7nrsLRbgCQOZEpmJclExmK5vRVN9%2FgeopluojZBFmNU%3B>

Example Response



```
{
  "response": {
    "action": "Login",
    "success": true,
    "time": 0
  }
}
```

Auth - Login - Admin

`/auth/login/admin/{user}/{pass}`



Like a standard [Auth - Login](#), except fails if the user is not a system administrator (ie this checks and logs in as admin in one go). Note that admin users can still login using the normal command also and have full admin privileges in both cases.

Authentication

Not required (this call gives auth)

Arguments

user (required)

username required to login, usually in the format of bob@ikanow.com

pass (required)

password required to login, "encrypted" with SHA-256 (Base-64), URL-encoded.

Example

<http://infinite.ikanow.com/api/auth/login/admin/bob@ikanow.com/pFyrA89j8fR5BJekoKdGD%2Bb83hV3sj48H4wAu3DR6c%3D>

Example Response



```
{
  "response": {
    "action": "Login",
    "success": true,
    "time": 0
  }
}
```

Auth - Logout

`/auth/logout`



Removes the current users authentication and cancels any other existing session.

Authentication

Removes authentication, see [Auth - Login](#)

Arguments

none

Example

<http://infinite.ikanow.com/api/auth/logout>

Example Response

```
{
  response: {
    action: "Logout"
    success: true
    message: "User logged out successfully"
    time: 91
  }
}
```

Chrome Source Extension

A Google Chrome Browser extension has been created to add RSS and single web page sources to the system easily.

Installation

1. Download the current [google chrome extension here](#).
2. Open Chrome and navigate to the Extensions Page (`chrome://extensions/` in the url bar or Left click on the options menu in the top right -> Settings -> Extensions)
3. Drag and drop the downloaded extension onto the extensions page.
4. To verify your install is successful you should now have the Ikanow lightbulb on your toolbar.

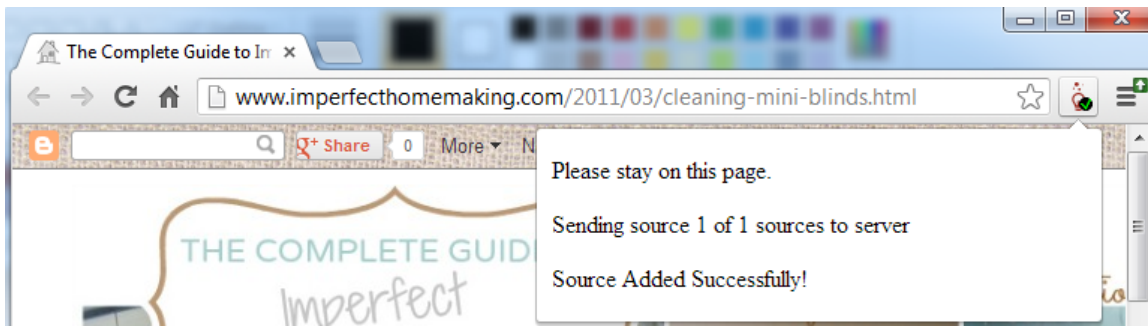
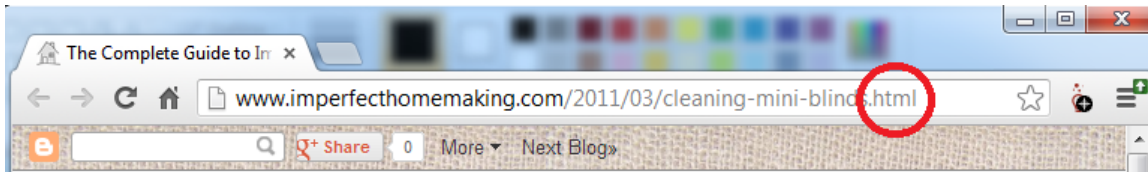
Configuration

1. Right click on the Ikanow lightbulb and click options.
2. Ensure the API points to your current server (probably <http://infinite.ikanow.com/api/>)
3. Type in your username e.g. `user@email.com`
4. Type in your password (warning passwords are stored as text in your html5 localStorage)
5. The timeout can be left at a default 60000
6. Push Login and it should say "Login Saved" and continue to a source selection page.

Usage

The chrome extension allows for simple source creation on single webpages and RSS feeds. I'll go through an example of adding both sources here





RSS Feed Source Creation

⚠ Note that not all the text/entity extractors are necessarily available for your system. For example, AlchemyAPI is a commercial service, and OpenCalais is free but requires an API key. Boilerpipe/Tika/TextRank are all built into the platform and are always available. Contact your system administrator to understand which are available.

1. Open the options page on the Infinite Source builder extension (right click the lightbulb in the top right and select options).
2. Login, you will be directed to a source selection page.
3. Click Create source
4. Fill out a title, description and media type for the source. Select a default extractor template and the community you want your source to be added to.

Source Title:

Source Description:

Source Media Type:

Extractor Template:

Text Extractor:

Entity Extractor:

Community:

To edit a source further, please use the [Source Manager Utility](#)

- Click Create Source, this will hold a temporary source with these settings in memory.
- Navigate to an RSS feed you want to add to Infinite in your Chrome browser, for example http://rss.cnn.com/rss/cnn_topstories.rss
- Left click the Infinite lightbulb, it should give you a success message that the RSS feed was added successfully.

Attempting to add an RSS feed.

Source Added Successfully!

- You can make sure the source was actually added by navigating to the source manager and searching for your source. <http://infinite.ikanow.com/manager/sources.jsp>

The screenshot shows the Infinite Source Manager interface. The top navigation bar includes 'People', 'Communities', 'Sources', 'Home', and 'Logout'. The main content area is titled 'Edit Source' and contains the following fields and controls:


- Source Functions:** Buttons for 'Test Source', 'Save Source', 'Save Source as Template', 'Publish So', and 'Delete docs'.
- Share ID:** 50f57346e4b07878cb6939fa
- Title:** Sample Source
- Description:** testing source creation
- Owner:** Caleb Burch - cburch@ikanow.com
- Community:** Burch Play
- Test Parameters:** Full Text: Number of Documents: 10 Update Test Mode

Below the form fields is a JSON representation of the source data:

```
{
  "_id": "50f572ece4b0f372d57bb050",
  "communityIds": ["506afc85e4b01d98fcf9bf5f"],
  "created": "Jan 15, 2013 10:17:00 AM",
  "description": "testing source creation",
  "extractType": "Feed",
  "harvest": {
    "harvest_status": "in_progress",
    "harvested": "Jan 15, 2013 10:18:27 AM",
    "lastHarvestedBy": "ip-10-224-58-86.ec2.internal",
    "synced": "Jan 16, 2012 10:18:09 AM"
  },
  "harvestBadSource": false,
  "isApproved": true,
  "key": "rss.cnn.com.rss.cnn_topstories.rss.",
  "mediaType": "News",
}
```

- The source should begin harvesting shortly and document will begin to show up in the Infinite GUI.

Single Webpage Source Creation

 Note that not all the text/entity extractors are necessarily available for your system. For example, AlchemyAPI is a commercial service, and OpenCalais is free but requires an API key. Boilerpipe/Tika/TextRank are all built into the platform and are always available. Contact your system administrator to understand which are available.

- Open the options page on the Infinite Source builder extension (right click the lightbulb in the top right and select options).
- Login, you will be directed to a source selection page.
- Click Create source
- Fill out a title, description and media type for the source. Select a default extractor template and the community you want your source to be added to.

Source Title:

Source Description:

Source Media Type:

Extractor Template:

Text Extractor:

Entity Extractor:

Community:

To edit a source further, please use the [Source Manager Utility](#)

- Click Create Source, this will hold a temporary source will these settings in memory.
- Navigate to a webpage you want to harvest, for example <http://www.cnn.com/2013/01/15/opinion/rothkopf-obama-gop-debt-ceiling/>
- Left click the Infinite lightbulb. It should give you a message saying Source added successfully!

Attempting to add a single page.

Source Added Successfully!

- This will have added the cnn article as a single page to the source you just built. You can add multiple pages to a single source, by just navigating to another site and clicking the Infinite icon again.
- You can make sure the source was actually added by navigating to the source manager and searching for your source. <http://infinite.ikanow.com/manager/sources.jsp>

Title:	<input type="text" value="Sample Source 2"/>
Description:	<input type="text" value="testing source creation"/>
Owner:	Caleb Burch - cburch@ikanow.com
Community:	<input type="text" value="Burch Play"/>
Test Parameters:	Full Text: <input type="checkbox"/> Number of Documents: <input type="text" value="10"/> Update T
<pre> "key": "www.cnn.com.2013.01.15.opinion.rothkopf-obama-gop-debt-ce.", "mediaType": "News", "modified": "Jan 15, 2013 10:26:20 AM", "ownerId": "5069dd90e4b09156bad31aba", "rss": {"extraUrls": [{ "description": "(no description)", "title": "GOP holding U.S. economy hostage on debt - CNN.com", "url": "http://www.cnn.com/2013/01/15/opinion/rothkopf-obama-gop-debt-ceiling/" }, { "description": "(no description)", "title": "The Ultimate Guide to Worldwide Etiquette Gives You an Overview of Customs from Around the World", "url": "http://lifehacker.com/5976061/the-ultimate-guide-to-worldwide-etiquette-gives-you-an-overview-of-customs-from-around-the-world" }, { "description": "(no description)", "title": "Learn Beginner and Advanced HTML/CSS Skills for Free", "url": "http://lifehacker.com/5974605/learn-beginner-and-advanced-htmlcss-skills-for-free" }]} </pre>	

- The source should begin harvesting shortly and document will begin to show up in the Infinite gui.

Community ID Regular Expression Usage

These API calls allow the usage of regular expressions to ease the use of multiple communities (see [community data model](#)) at once:

- `config/source/good`
- `config/source/bad`
- `config/source/pending`
- `knowledge/document/query`
- `knowledge/feature/geoSuggest`
- `knowledge/feature/entitySuggest`
- `knowledge/feature/aliasSuggest`
- `knowledge/feature/assocSuggest`

In place of just sending a comma delimited list of community ids you can now send * or *<regex> to signify retrieving multiple community ids matching either all (*) or all matching a regex (*<regex>). See below for some examples on how it can be used.

Take the `config/source/good` api call as an example, it has 1 parameter "communityid-list".

You would normally call this api call via: <http://infinite.ikanow.com/api/config/source/good/4f579d7e8c2200000001b3b,4f579d7e8c2200000001b3c>

This would return the good sources for both community abcde12345 and abcde12346. This would be a pain to use if we wanted to return all good sources so instead we can use the new syntax to return good sources for all communities we have access to via: http://infinite.ikanow.com/api/config/source/good/*

We can also be more specific and only return the good sources for any source with cnn in the name via: http://infinite.ikanow.com/api/config/source/good/*cnn

Config - Source

Overview of the Infinite Data Harvesting Process

The Infinite platform features a robust set of data harvesters that give Infinite a powerful data extraction and transformation (enrichment) capability. Infinite's harvesters are designed to consume data from a variety of sources and media types including:

- Web based content accessible via URL including:
 - Static HTML content;
 - RSS and ATOM based news feeds;
 - Restful web services interfaces.
- Traditional relational database management systems (RDBMS) via Java Database Connectivity (JDBC) drivers;
- Files located on local and network attached storage devices.

The following steps are followed:

1. Extract data from source, turn into documents, extract metadata from sources for XML, PDF etc (**harvesting**)
2. Enrich source data by extracting entities, events, geographic/location data, etc. This is broken down into the following phases (**enrichment**; note: the roadmap is to move this to a completely user-defined UIMA chain):
 - a. *Structured Analysis Handler, phase 1: fill in unstructured document-level fields (title, description, full text) from metadata, if needed.*
 - b. *Unstructured Analysis Handler, phase 1: use regexes and javascript to pull out new metadata fields from the unstructured document-level fields.*
 - i. *(Special case: if Tika is specified as the text extraction engine, then this is performed before any Unstructured Analysis Handler)*
 - c. *Unstructured Analysis Handler, phase 2: use regex replaces to transform the source text, if needed.*
 - d. *Unstructured Analysis Handler, phase 3: use regexes and javascript to pull out new metadata fields from the cleansed unstructured document-level fields.*
 - e. *Standard extraction, phase 1 (text extraction): use a "text extractor" to create the text that is submitted to the entity extraction service in the next phase (if needed, often the entity extraction service will combine the 2 phases).*
 - f. *Standard extraction, phase 2 (entity extraction): use an "entity extractor" (eg *AlchemyAPI*) to pull out entities and associations from the submitted text/URL.*
 - g. *Structured Analysis Handler, phase 2: the remaining document-level field (URL, published data, document geo ... plus the title and description and full text if these returned null before, ie in case the UAH has filled in required fields)*
 - h. *Structured Analysis Handler, phase 3: create new entities from the metadata, combine entities from all phases into associations.*
3. Update entity counts/aggregates (**generic processing - statistics**)
4. Store finished within Infinite's MongoDB data store and Elasticsearch index (**generic processing - aggregation**)

Creating a Source

The following WIKI pages describe detail the steps involved with creating sources:

1. **Specifying a data source**

How to specify the mechanics required to extract data from a source system:

 - a. [Using the Feed Harvester](#)
 - b. [Using the Database Harvester](#)
 - c. [Using the File Harvester](#)
2. **Structured Analysis - Overview**

An introduction to the Structured Analysis Harvester and how to specify the methods for enriching structured data sources with

geographic information, entities, and events.

- a. [Specifying Document Level Geographical Location](#)
- b. [Specifying Entities](#)
- c. [Specifying Associations](#)
- d. [Transforming Data with JavaScript](#)

3. **Unstructured Analysis - Overview**

An introduction to the Unstructured Analysis Harvester.

A simple [web-based GUI](#) is available in conjunction with the structures described in these pages.

Source Reference Documents

Source Document Specification

The following links provide detailed information regarding the objects that make up a Source document and the individual fields within each object to support the introductory materials above.

- [Source configuration objects](#)
 - [Authentication Object](#)
 - [Database Object](#)
 - [File Object](#)
 - [StructuredAnalysis Object](#)
 - [UnstructuredAnalysis Object](#)

Sample Source Documents

The following sample source documents are provided as an aid to learning how to create your own sources:

- [RSS Feed Source](#)
- [MySQL Database Source](#)
- [XML File Source](#)
- [Unstructured File Source](#)
- [Log File Source](#)
- [JSON File Source](#)

Source APIs:

- [API documentation](#)

Source building - reference

By category

Harvest Types

- [Feed](#)
 - [RSS](#)
 - [HTML](#)
 - [Following Links](#)
- [File](#)
 - ["office"](#)
 - [XML](#)
 - [JSON](#)
- [Database](#)
 - [SQL](#)

Search and update cycles

- [Search cycles](#)
- [Update cycles](#)

Generating metadata

- [Using regex](#)
- [Using javascript](#)
 - [Global functions](#)
 - [Accessing external content](#)
- [Using xpath](#)
- [Metadata pipelines](#)

Generating entities from metadata

- [With strings/replacement](#)
- [With javascript](#)
 - [Global functions](#)
- [From metadata arrays](#)

Generating associations from metadata

- With strings/replacement
- With javascript
 - Global functions
- From metadata arrays

Generating associations from entities

- With strings
- With javascript

Retaining and discarding metadata for storage and/or indexing

- Retaining/discarding metadata for storage
- Retaining/discarding entities, associations, metadata

Source gallery



Note that the source format is being modified, so this gallery is not very active - once the new format is finalized, it will be converted to a set of examples in the new format

By category

Harvest Types

- Feed
 - RSS: [Feed Source](#)
 - HTML: [Log File Source Gallery](#), [Web-hosted XML](#)
 - Following links: [Log File Source Gallery](#), [Web-hosted XML](#)
- File
 - "office": [Enron sample](#)
 - Line-separated: [Log File Source Gallery](#)
 - XML: [WITS sample](#)
 - JSON: [GNIP sample](#)
- Database
 - SQL: [DC crime sample](#)

Search and update cycles

- Search cycles
- Update cycles: [DC crime sample](#)

Generating metadata

- Using regex
- Using javascript: [GNIP sample](#), [Log File Source Gallery](#), [Enron sample](#)
 - Global functions
 - Accessing external content
- Using xpath: [Web-hosted XML](#) (web following context)
- Metadata pipelines

Generating entities and associations using NLP

- Text cleansing: [Log File Source Gallery](#), [Enron sample](#)
- Specifying the text extraction engine: [Feed Source](#)
- Specifying the entity extraction engine: [Enron sample](#), [Feed Source](#)

Generating entities from metadata

- With strings/replacement: [GNIP sample](#), [WITS sample](#), [DC crime sample](#), [Log File Source Gallery](#), [Enron sample](#)
- With javascript: [GNIP sample](#), [WITS sample](#), [Log File Source Gallery](#), [Enron sample](#)
 - Global functions: [WITS sample](#)
- From metadata arrays: [GNIP sample](#)

Generating associations from metadata

- With strings/replacement: [GNIP sample](#), [Log File Source Gallery](#), [Enron sample](#)
- With javascript: [GNIP sample](#), [Log File Source Gallery](#), [Enron sample](#)
 - Global functions: [WITS sample](#)
- From metadata arrays: [GNIP sample](#), [Log File Source Gallery](#), [Enron sample](#)

Generating associations from entities

- With strings: [DC crime sample](#), [Log File Source Gallery](#), [Enron sample](#)
- With javascript: [WITS sample](#), [Log File Source Gallery](#), [Enron sample](#)

Retaining and discarding metadata for storage and/or indexing

- Retaining/discarding metadata for storage
- Retaining/discarding entities, associations, metadata for indexing: [GNIP sample](#)

By source

GNIP

[Source, example documents and output](#)

Categories:

- File (JSON)
- Unstructured Analysis
 - Javascript
- Structured Analysis
 - Entities
 - Associations
 - Entities and associations from arrays
 - Javascript

WITS

[Source, example documents and output](#)

Categories:

- File (XML)
- Structured Analysis
 - Entities
 - Associations
 - Entities from arrays
 - Javascript

DC crime data

[Source, example documents and output](#)

Categories:

- Database (mysql)
- Structured Analysis
 - Entities
 - Associations
 - Entities from arrays
 - Javascript

Enron data

[Source, example documents and output](#)

Categories:

- File ("office")
- Entities from NLP
- Unstructured Analysis
 - regex
- Structured Analysis
 - Entities
 - Associations
 - Entities from arrays
 - Javascript

Web-hosted XML

[Source, example documents and output](#)

Categories:

- Feed (Web)
- Following Links
 - xpath
 - javascript
- Unstructured Analysis
 - javascript

Log file data

[Source, example documents and output](#)

Categories:

- Feed (Web), File (line-separated)
- Following Links
 - xpath

- javascript
- Unstructured Analysis
 - javascript
- Structured Analysis
 - Entities
 - Associations
 - Entities from arrays
 - Javascript

Feed Source

Source, example documents and output

Categories:

- Feed (RSS)
- Entities from NLP

DC crime data source gallery

Sample document

nid	ccn	reportdate time	shift	offense	method	block site address	latitude	longitude	city	state	ward	anc	smd	district	psa
955778	11012669	Jan 29, 2011 12:00:00 AM UTC	UNK	THEFT	2	1300 B/O CON NEC TICUT AVE NW	38.90992780287290	-77.04360677659660	WASHINGTON	DC	2	2B	2B02	SECOND	208.0

Source

```
{
  "authentication": {
    "password": "PASSWORD",
    "username": "USERNAME"
  },
  "database": {
    "databaseName": "washingtondc",
    "databaseType": "mysql",
    "deleteQuery": "",
    "deltaQuery": "SELECT * FROM IncidentReport WHERE REPORTDATETIME >= (SELECT ADDDATE(CURDATE(),-7))",
    "hostname": "dbserver.ikanow.com",
    "port": "3306",
    "primaryKey": "NID",
    "publishedDate": "REPORTDATETIME",
    "query": "SELECT * FROM IncidentReport",
    "snippet": "OFFENSE",
    "title": "CCN"
  },
  "description": "Incident reports from the Washington, D.C. MPD",
  "extractType": "Database",
  "harvest": {
    "doccount": 9314,
    "harvest_message": "Error when harvesting DB: Cannot establish connection to the database : jdbc:mysql://ec2-184-72-206-97.compute-1.amazonaws.com:3306/washingtondc",
    "harvest_status": "error",
  }
}
```

```
    "harvested": "Nov 2, 2012 11:26:25 AM",
    "synced": "Nov 2, 2012 11:24:30 AM"
  },
  "isApproved": true,
  "isPublic": true,
  "mediaType": "Record",
  "structuredAnalysis": {
    "associations": [{
      "entity1": "$metadata.offense,$metadata.method",
      "geotag": {},
      "time_start": "$metadata.reportdatetime",
      "verb": "reported",
      "verb_category": "crime"
    }],
    "description": "$metadata.reportdatetime: $metadata.offense,$metadata.method
was reported at: $metadata.blocksiteaddress",
    "docGeo": {
      "lat": "$metadata.latitude",
      "lon": "$metadata.longitude"
    },
    "entities": [
      {
        "dimension": "What",
        "disambiguated_name": "$metadata.offense,$metadata.method",
        "type": "CriminalActivity"
      },
      {
        "dimension": "Where",
        "disambiguated_name":
"$metadata.blocksiteaddress,$metadata.city,$metadata.state",
        "geotag": {},
        "type": "Place"
      },
      {
        "dimension": "Where",
        "disambiguated_name": "$metadata.ward,$metadata.city,$metadata.state",
        "type": "Place"
      },
      {
        "dimension": "Where",
        "disambiguated_name":
"$metadata.district,$metadata.city,$metadata.state",
        "type": "Place"
      }
    ]
  },
  "tags": [
    "crime report",
    "MPD",
    "Washington, D.C."
  ],
  "title": "data.dc.gov - Crime Incidents (ASP)",
  "url":
```

```
"jdbc:mysql://ec2-184-72-206-97.compute-1.amazonaws.com/washingtondc/IncidentReport",
  "useExtractor": "none"
}
```

Sample output

```
{
  "_id": "4ebla6532285aled2af759c",
  "title": "11012669",
  "url": "jdbc:mysql://dbserver.ikanow.com/washingtondc/IncidentReport/955778",
  "created": "Nov 2, 2011 08:21:39 PM UTC",
  "modified": "Nov 2, 2011 08:21:39 PM UTC",
  "publishedDate": "Jan 29, 2011 05:00:00 AM UTC",
  "source": [
    "data.dc.gov - Crime Incidents (ASP)"
  ],
  "sourceKey": [
    "Bergen.washingtondc.IncidentReport"
  ],
  "mediaType": [
    "Record"
  ],
  "description": "Jan 29, 2011 12:00:00 AM: THEFT,2 was reported at: 1300 B/O CONNECTICUT AVE NW",
  "entities": [
    {
      "disambiguated_name": "THEFT,2",
      "index": "theft,2/criminalactivity",
      "actual_name": "THEFT,2",
      "type": "CriminalActivity",
      "frequency": 1,
      "totalfrequency": 2705,
      "doccount": 2705,
      "dimension": "What"
    },
    {
      "disambiguated_name": "1300 B/O CONNECTICUT AVE NW, WASHINGTON, DC",
      "index": "1300 b/o connecticut ave nw, washington, dc/place",
      "actual_name": "1300 B/O CONNECTICUT AVE NW, WASHINGTON, DC",
      "type": "Place",
      "frequency": 1,
      "totalfrequency": 28,
      "doccount": 28,
      "geotag": {
        "lat": 38.9099278028729,
        "lon": -77.0436067765966
      },
      "dimension": "Where",
      "ontology_type": "point"
    },
    {
      "disambiguated_name": "2, WASHINGTON, DC",
      "index": "2, washington, dc/place",
      "actual_name": "2, WASHINGTON, DC",
      "type": "Place",
      "frequency": 1,
    }
  ]
}
```

```
    "totalfrequency": 2326,
    "doccount": 2326,
    "dimension": "Where"
  },
  {
    "disambiguated_name": "SECOND,WASHINGTON,DC",
    "index": "second,washington,dc/place",
    "actual_name": "SECOND,WASHINGTON,DC",
    "type": "Place",
    "frequency": 1,
    "totalfrequency": 2120,
    "doccount": 2120,
    "dimension": "Where"
  }
],
"tags": [
  "crime report",
  "MPD",
  "Washington, D.C."
],
"communityId": [
  "4c927585d591d31d7c37097b"
],
"associations": [
  {
    "entity1": "theft,2",
    "entity1_index": "theft,2/criminalactivity",
    "verb": "reported",
    "verb_category": "crime",
    "time_start": "2011-01-29T00:00:00",
    "geotag": {
      "lat": 38.9099278028729,
      "lon": -77.0436067765966
    },
    "assoc_type": "Summary"
  }
],
"metadata": {
  "nid": [
    [
      {
        "value": "955778.0"
      }
    ]
  ],
  "ccn": [
    [
      {
        "value": "1.1012669E7"
      }
    ]
  ],
  "reportdatetime": [
    [
      "Jan 29, 2011 12:00:00 AM UTC"
    ]
  ],
  "shift": [
    [
```



```
        "UNK"
    ]
  ],
  "offense": [
    [
      "THEFT"
    ]
  ],
  "method": [
    [
      "2"
    ]
  ],
  "blocksiteaddress": [
    [
      "1300 B/O CONNECTICUT AVE NW"
    ]
  ],
  "latitude": [
    [
      "38.90992780287290"
    ]
  ],
  "longitude": [
    [
      "-77.04360677659660"
    ]
  ],
  "city": [
    [
      "WASHINGTON"
    ]
  ],
  "state": [
    [
      "DC"
    ]
  ],
  "ward": [
    [
      {
        "value": "2.0"
      }
    ]
  ],
  "anc": [
    [
      "2B"
    ]
  ],
  "smd": [
    [
      "2B02"
    ]
  ],
  "district": [
    [
      "SECOND"
    ]
  ]
}
```

```
],
  "psa": [
    [
      {
        "value": "208.0"
      }
    ]
  ]
},
"docGeo": {
  "lat": 38.9099278028729,
```

```
"lon": -77.0436067765966
}
}
```

Enron source gallery

Sample input document

```
Message-ID: <32220443.1075841552668.JavaMail.evans@thyme>
Date: Mon, 9 Jul 2001 11:33:32 -0700 (PDT)
From: cara.semperger@enron.com
To: will.smith@enron.com
Subject: RE: Testing Preschedule workspace
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: quoted-printable
X-From: Semperger, Cara </O=ENRON/OU=NA/CN=RECIPIENTS/CN=CSEMPER>
X-To: Smith, Will </O=ENRON/OU=NA/CN=RECIPIENTS/CN=Wsmith>
X-cc:
X-bcc:
X-Folder: \ExMerge - Semperger, Cara\Deleted Items
X-Origin: SEMPERGER-C
X-FileName: cara semperger 6-26-02.PST
```

I am trying to pull it up now, it's taking a long time

```
-----Original Message-----
From: =09Semperger, Cara =20
Sent:=09Monday, July 09, 2001 10:40 AM
To:=09Smith, Will; Atta, Asem
Cc:=09Bentley, Corry; Poston, David
Subject:=09Testing Preschedule workspace
```

Good Morning,

My target testing date today is June 18th, I am running in Test P in Local=Enpower using actual data from our scheduling sheets re-arranged to meet the new guidelines.

The daily deals I coded X in columns J and N, the Month long bookouts and =BOM bookouts I coded R. =20

What worked:

I was able to retrieve my saved workspace with all data intact. I had previously successfully copied and pasted my entire sheet from EXCEL to the PSW.

I was able to run the build route report with the criteria of "Starting On=June 18-PaloVerde-Day of week Mask Activated-Report Changes activated." A check of deals actually scheduled vs. build route results showed that all deals were extracted correctly from Enpower. Because I am working on closed dates, a cumulative test of this app will not be fully testable until production. We are expecting to see the same functionality as the current incarnation of Build route. The data extracted should be read only, time stamped, and when run multiple times additional data should be shown below previous=

ly extracted data. The improvement we are expecting to see is the app should not duplicate deal strips on dates that have no physical power flow. (West Light Load currently does this in Start view, but not Active view)

Navigating around the scheduling sheet itself I was able to accurately execute the sort function on a single criteria at a time. Multiple sorting will continue to be done in excel, or we can do a series of single sorts in the PSW to achieve the same result.

Routing deals: Will had deleted all routes for June 18th, starting me with a clean slate. I made every path be for DAY. I was unable to confirm total unrouted MWH, as the real time position manager does not seem to be functioning in TESTP. The routing appeared to take 19 minutes with the status bar showing steady progress during that time. This time is 15-17 minutes longer than current speed using the Excel Macro system we have now. The error list gave me a row by row description of what did not route, a very useful tool. OK was visible on all rows that the PSW believed that it had routed. I had difficulty checking the routing results, as I kept getting BDE errors in Scheduling after routing had occurred (Local Enpower). Scheduling kept starting up in 1899. I was unable to login to TestP through Terminal server 2, but was able to in Terminal Server 5. The results there were very encouraging! Most routing was done, and a spot check of deals shows that they were routed properly. The deals that were not routed appear to be due to a user error of deal number duplication in the sheet. This is consistent with what I would expect. I will further evaluate routing ability with our more complicated paths later. This routing was very easy, a large point with on peak non shaped deals only.

Things I did not expect that I liked:

When I highlight a group of cells in Build Route, it stays highlighted when I move up to the scheduling sheet to highlight a comparison group of cells. This is very handy for double checking Build route against the scheduler's sheet.

What does not appear to be working at this time:

The physical or not physical flag of path does not seem to be showing up properly in routing.

Path Confirmation: The running time appeared to be over one hour for one sheet, only 70 rows of the sheet being flagged for insertion into confirmation. This current speed will not be sufficient to work in production. Also, many rows that were flagged for confirmation were not imported, and I cannot find an error log to help understand why deals were not imported to path confirmation.

When the path confirmation task was finished, the application simply froze. The status bar was no longer visible, leading me to believe that it was done, however the app was not able to be saved or closed or examined further.

My conclusions:

The build route and routing functions work well enough to use in production=

, the copy-paste function works well for the West desk per our connectivity= issues.

Path Confirmation is not functioning at this point, and appears to be blowi= ng up the app. No data was visible for June 18th even after the PSW ran thr= ough its import function.

Please let me know when the issues I have named have been addressed and are= ready for further testing.

Thanks

Cara
503/464-3814

Source

```
{
  "description": "All of the Enron emails corpus with TextRank keyword extraction
enabled.",
  "extractType": "File",
  "file": {
    "domain": "DOMAIN",
    "password": "PASSWORD",
    "username": "USER"
  },
  "isPublic": true,
  "mediaType": "Email",
  "searchCycle_secs": -1,
  "structuredAnalysis": {
    "associations": [
      {
        "associations": [
          {
            "assoc_type": "Event",
            "entity1": "$SCRIPT( return
_doc.metadata._FILE_METADATA_[0].metadata.Author[0];)",
            "entity2": "$SCRIPT(return _value;)",
            "iterateOver": "Message-To",
            "time_start": "$SCRIPT( return _doc.publishedDate;)",
            "verb": "emailed",
            "verb_category": "emailed/communicated"
          }
        ],
        "iterateOver": "email_meta"
      }
    ],
    "entities": [
      {
        "dimension": "What",
        "disambiguated_name": "$SCRIPT( return
_doc.metadata._FILE_METADATA_[0].metadata.Author[0];)",
        "type": "Account",
        "useDocGeo": false
      },
      {
        "entities": [
          {
            "dimension": "What",
            "disambiguated_name": "",
            "iterateOver": "Message-To",
            "type": "Account",
            "useDocGeo": false
          }
        ],
        "iterateOver": "email_meta"
      }
    ]
  }
}
```

```

    ],
    "scriptEngine": "JavaScript",
    "title": "$SCRIPT( return
_doc.metadata._FILE_METADATA_[0].metadata.subject[0];)"
  },
  "tags": [
    "enron",
    "email",
    "fraud"
  ],
  "title": "All Enron Emails (TextRank)",
  "unstructuredAnalysis": {
    "meta": [
      {
        "context": "All",
        "fieldName": "email_meta",
        "flags": "m",
        "script": "var x=_metadata._FILE_METADATA_[0].metadata;x;",
        "scriptlang": "javascript"
      }
    ],
    "simpleTextCleanser": [
      {
        "field": "fullText",
        "flags": "md",
        "replacement": " ",
        "script": "(?:\\[.*?\\])",
        "scriptlang": "regex"
      },
      {
        "field": "description",
        "flags": "md",
        "replacement": " ",
        "script": "(?:\\[.*?\\])",
        "scriptlang": "regex"
      },
      {
        "field": "fullText",
        "flags": "md",
        "replacement": ". ",
        "script": "<.*?>",
        "scriptlang": "regex"
      },
      {
        "field": "description",
        "flags": "md",
        "replacement": ". ",
        "script": "<.*?>",
        "scriptlang": "regex"
      },
      {
        "field": "fullText",
        "flags": "md",
        "replacement": ". ",
        "script": "(?:>|<)",
        "scriptlang": "regex"
      },
      {
        "field": "description",

```

```
    "flags": "md",
    "replacement": ". ",
    "script": "(?:>|<)",
    "scriptlang": "regex"
  },
  {
    "field": "fullText",
    "replacement": " ",
    "script": "(?:[-]{4,}(.*[-]{4,}|\n))",
    "scriptlang": "regex"
  },
  {
    "field": "description",
    "replacement": " ",
    "script": "(?:[-]{4,}(.*[-]{4,}|\n))",
    "scriptlang": "regex"
  },
  {
    "field": "fullText",
    "replacement": " ",
    "script": "(?:\\*{2,})",
    "scriptlang": "regex"
  },
  {
    "field": "description",
    "replacement": " ",
    "script": "(?:\\*{2,})",
    "scriptlang": "regex"
  }
]
},
"url": "smb://modus:139/enron/enron_mail_20110402/maildir/",
```



```
"useExtractor": "textrank",
"useTextExtractor": "none"
}
```

Sample output document

```
{
  "_id": "5048efb0e4b01fd6455420ee",
  "title": "RE: Testing Preschedule workspace",
  "url": "smb://modus.139/enron/testing/semperger-c/deleted_items/37QTKE-3",
  "created": "Sep 6, 2012 06:42:01 PM UTC",
  "modified": "Jul 24, 2012 01:13:02 AM UTC",
  "publishedDate": "Jul 9, 2001 06:33:32 PM UTC",
  "source": [
    "Enron Emails (TextRank)"
  ],
  "sourceKey": [
    "modus.139.enron.testing.."
  ],
  "mediaType": [
    "Email"
  ],
  "description": "I am trying to pull it up now, it's taking a long time\r\n\r\n\r\n\r\nFrom: \tSmith, Will \r\nSent:\tMonday, July 09, 2001 11:28 AM\r\nTo:\tSemperger, Cara\r\nSubject:\tRE: Testing Preschedule workspace\r\n\r\n\r\nYes, but Vish made the changes in Table Edit. : - )\r\n\r\n\r\nWill\r\n\r\n\r\n\r\nFrom: \tSemperger, Cara \r\nSent:\tMonday, July 09, 2001 1:20 PM\r\nTo:\tSmith, Will\r\nSubject:\tRE: Testing Preschedule workspace\r\n\r\n\r\nSo, this table edit that Brett is asking me to test is really from ",
  "entities": [
    {
      "disambiguated_name": "on- june 18-paloverde-day",
      "index": "on- june 18-paloverde-day/keyword",
      "actual_name": "on- june 18-paloverde-day",
      "type": "Keyword",
      "relevance": 0.10585404743253149,
      "frequency": 1,
      "totalfrequency": 12,
      "doccount": 12,
      "dimension": "What"
    },
    {
      "disambiguated_name": "mulitple times additional data",
      "index": "mulitple times additional data/keyword",
      "actual_name": "mulitple times additional data",
      "type": "Keyword",
      "relevance": 0.18088061045762382,
      "frequency": 1,
      "totalfrequency": 12,
      "doccount": 12,
      "dimension": "What"
    },
    {
      "disambiguated_name": "scheduling sheets",
      "index": "scheduling sheets/keyword",
      "actual_name": "scheduling sheets",

```

```
    "type": "Keyword",
    "relevance": 0.15086086188384693,
    "frequency": 1,
    "totalfrequency": 20,
    "doccount": 20,
    "dimension": "What"
  },
  {
    "disambiguated_name": "app",
    "index": "app/keyword",
    "actual_name": "app",
    "type": "Keyword",
    "relevance": 0.20415634782171557,
    "frequency": 1,
    "totalfrequency": 58,
    "doccount": 58,
    "dimension": "What"
  },
  {
    "disambiguated_name": "data",
    "index": "data/keyword",
    "actual_name": "data",
    "type": "Keyword",
    "relevance": 0.1361375118885727,
    "frequency": 1,
    "totalfrequency": 3323,
    "doccount": 3323,
    "dimension": "What"
  },
  {
    "disambiguated_name": "paths",
    "index": "paths/keyword",
    "actual_name": "paths",
    "type": "Keyword",
    "relevance": 0.2041916488834702,
    "frequency": 1,
    "totalfrequency": 99,
    "doccount": 99,
    "dimension": "What"
  },
  {
    "disambiguated_name": "build route report",
    "index": "build route report/keyword",
    "actual_name": "build route report",
    "type": "Keyword",
    "relevance": 0.11476307758997932,
    "frequency": 1,
    "totalfrequency": 36,
    "doccount": 36,
    "dimension": "What"
  },
  {
    "disambiguated_name": "testing preschedule workspace cara",
    "index": "testing preschedule workspace cara/keyword",
    "actual_name": "testing preschedule workspace cara",
    "type": "Keyword",
    "relevance": 0.16803833041631702,
    "frequency": 1,
    "totalfrequency": 8,
```

```
    "doccount": 8,
    "dimension": "What"
  },
  {
    "disambiguated_name": "physical power flow",
    "index": "physical power flow/keyword",
    "actual_name": "physical power flow",
    "type": "Keyword",
    "relevance": 0.11805512187037151,
    "frequency": 1,
    "totalfrequency": 17,
    "doccount": 17,
    "dimension": "What"
  },
  {
    "disambiguated_name": "i",
    "index": "i/keyword",
    "actual_name": "i",
    "type": "Keyword",
    "relevance": 0.13651904141534263,
    "frequency": 1,
    "totalfrequency": 18162,
    "doccount": 18162,
    "dimension": "What"
  },
  {
    "disambiguated_name": "total running time",
    "index": "total running time/keyword",
    "actual_name": "total running time",
    "type": "Keyword",
    "relevance": 0.11233232851584997,
    "frequency": 1,
    "totalfrequency": 10,
    "doccount": 10,
    "dimension": "What"
  },
  {
    "disambiguated_name": "time",
    "index": "time/keyword",
    "actual_name": "time",
    "type": "Keyword",
    "relevance": 0.34020922533185516,
    "frequency": 1,
    "totalfrequency": 17102,
    "doccount": 17102,
    "dimension": "What"
  },
  {
    "disambiguated_name": "psw",
    "index": "psw/keyword",
    "actual_name": "psw",
    "type": "Keyword",
    "relevance": 0.13625985262266815,
    "frequency": 1,
    "totalfrequency": 46,
    "doccount": 46,
    "dimension": "What"
  }
}
```

```
"disambiguated_name": "semperger",
"index": "semperger/keyword",
"actual_name": "semperger",
"type": "Keyword",
"relevance": 0.2724417241053495,
"frequency": 1,
"totalfrequency": 226,
"doccount": 226,
"dimension": "What"
},
{
  "disambiguated_name": "peak non shaped deals",
  "index": "peak non shaped deals/keyword",
  "actual_name": "peak non shaped deals",
  "type": "Keyword",
  "relevance": 0.19127581970645322,
  "frequency": 1,
  "totalfrequency": 12,
  "doccount": 12,
  "dimension": "What"
},
{
  "disambiguated_name": "table edit",
  "index": "table edit/keyword",
  "actual_name": "table edit",
  "type": "Keyword",
  "relevance": 0.21207334129182112,
  "frequency": 1,
  "totalfrequency": 32,
  "doccount": 32,
  "dimension": "What"
},
{
  "disambiguated_name": "week mask activated-report changes",
  "index": "week mask activated-report changes/keyword",
  "actual_name": "week mask activated-report changes",
  "type": "Keyword",
  "relevance": 0.1484580867667756,
  "frequency": 1,
  "totalfrequency": 12,
  "doccount": 12,
  "dimension": "What"
},
{
  "disambiguated_name": "excel macro system",
  "index": "excel macro system/keyword",
  "actual_name": "excel macro system",
  "type": "Keyword",
  "relevance": 0.12208201691477336,
  "frequency": 1,
  "totalfrequency": 12,
  "doccount": 12,
  "dimension": "What"
},
{
  "disambiguated_name": "real time position manager",
  "index": "real time position manager/keyword",
  "actual_name": "real time position manager",
  "type": "Keyword",
```

```
"relevance": 0.19213464212989614,
"frequency": 1,
"totalfrequency": 39,
"doccount": 39,
"dimension": "What"
},
{
  "disambiguated_name": "testing preschedule workspace",
  "index": "testing preschedule workspace/keyword",
  "actual_name": "testing preschedule workspace",
  "type": "Keyword",
  "relevance": 0.17652180791002264,
  "frequency": 1,
  "totalfrequency": 12,
  "doccount": 12,
  "dimension": "What"
},
{
  "disambiguated_name": "cara",
  "index": "cara/keyword",
  "actual_name": "cara",
  "type": "Keyword",
  "relevance": 0.20414801224595303,
  "frequency": 1,
  "totalfrequency": 736,
  "doccount": 736,
  "dimension": "What"
},
{
  "disambiguated_name": "smith",
  "index": "smith/keyword",
  "actual_name": "smith",
  "type": "Keyword",
  "relevance": 0.27217844252943296,
  "frequency": 1,
  "totalfrequency": 783,
  "doccount": 783,
  "dimension": "What"
},
{
  "disambiguated_name": "david subject",
  "index": "david subject/keyword",
  "actual_name": "david subject",
  "type": "Keyword",
  "relevance": 0.15139765579194864,
  "frequency": 1,
  "totalfrequency": 930,
  "doccount": 930,
  "dimension": "What"
},
{
  "disambiguated_name": "sheet",
  "index": "sheet/keyword",
  "actual_name": "sheet",
  "type": "Keyword",
  "relevance": 0.20416968108320477,
  "frequency": 1,
  "totalfrequency": 436,
  "doccount": 436,
```

```
    "dimension": "What"
  },
  {
    "disambiguated_name": "total unrouted mwh",
    "index": "total unrouted mwh/keyword",
    "actual_name": "total unrouted mwh",
    "type": "Keyword",
    "relevance": 0.1141385057566826,
    "frequency": 1,
    "totalfrequency": 16,
    "doccount": 16,
    "dimension": "What"
  },
  {
    "disambiguated_name": "target testing date today",
    "index": "target testing date today/keyword",
    "actual_name": "target testing date today",
    "type": "Keyword",
    "relevance": 0.18726422286448255,
    "frequency": 1,
    "totalfrequency": 12,
    "doccount": 12,
    "dimension": "What"
  },
  {
    "disambiguated_name": "deals",
    "index": "deals/keyword",
    "actual_name": "deals",
    "type": "Keyword",
    "relevance": 0.34025706056156424,
    "frequency": 1,
    "totalfrequency": 5740,
    "doccount": 5261,
    "dimension": "What"
  },
  {
    "disambiguated_name": "double checking build route",
    "index": "double checking build route/keyword",
    "actual_name": "double checking build route",
    "type": "Keyword",
    "relevance": 0.18886230001363824,
    "frequency": 1,
    "totalfrequency": 12,
    "doccount": 12,
    "dimension": "What"
  },
  {
    "disambiguated_name": "path confirmation task",
    "index": "path confirmation task/keyword",
    "actual_name": "path confirmation task",
    "type": "Keyword",
    "relevance": 0.12326679747563907,
    "frequency": 1,
    "totalfrequency": 16,
    "doccount": 16,
    "dimension": "What"
  },
  {
    "disambiguated_name": "routes",
```

```
"index": "routes/keyword",
"actual_name": "routes",
"type": "Keyword",
"relevance": 0.40825322818399834,
"frequency": 1,
"totalfrequency": 142,
"doccount": 142,
"dimension": "What"
},
{
  "disambiguated_name": "west light load",
  "index": "west light load/keyword",
  "actual_name": "west light load",
  "type": "Keyword",
  "relevance": 0.11288042191103252,
  "frequency": 1,
  "totalfrequency": 16,
  "doccount": 16,
  "dimension": "What"
},
{
  "disambiguated_name": "rows",
  "index": "rows/keyword",
  "actual_name": "rows",
  "type": "Keyword",
  "relevance": 0.2721919612854695,
  "frequency": 1,
  "totalfrequency": 72,
  "doccount": 72,
  "dimension": "What"
},
{
  "disambiguated_name": "path confirmation",
  "index": "path confirmation/keyword",
  "actual_name": "path confirmation",
  "type": "Keyword",
  "relevance": 0.2124247462661659,
  "frequency": 1,
  "totalfrequency": 169,
  "doccount": 169,
  "dimension": "What"
},
{
  "disambiguated_name": "month long bookouts",
  "index": "month long bookouts/keyword",
  "actual_name": "month long bookouts",
  "type": "Keyword",
  "relevance": 0.12514486683483175,
  "frequency": 1,
  "totalfrequency": 18,
  "doccount": 18,
  "dimension": "What"
},
{
  "disambiguated_name": "deal number duplication",
  "index": "deal number duplication/keyword",
  "actual_name": "deal number duplication",
  "type": "Keyword",
  "relevance": 0.12910499876653425,
```

```
    "frequency": 1,
    "totalfrequency": 16,
    "doccount": 16,
    "dimension": "What"
  },
  {
    "disambiguated_name": "minutes",
    "index": "minutes/keyword",
    "actual_name": "minutes",
    "type": "Keyword",
    "relevance": 0.13613482658399254,
    "frequency": 1,
    "totalfrequency": 1234,
    "doccount": 1172,
    "dimension": "What"
  },
  {
    "disambiguated_name": "cara.semperger@enron.com",
    "index": "cara.semperger@enron.com/account",
    "actual_name": "cara.semperger@enron.com",
    "type": "Account",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 3251,
    "doccount": 3251,
    "dimension": "What"
  },
  {
    "disambiguated_name": "will.smith@enron.com",
    "index": "will.smith@enron.com/account",
    "actual_name": "will.smith@enron.com",
    "type": "Account",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 408,
    "doccount": 408,
    "dimension": "What"
  }
],
"tags": [
  "enron",
  "email",
  "fraud"
],
"communityId": [
  "500df237e4b00e332fe993aa"
],
"associations": [
  {
    "entity1": "cara.semperger@enron.com",
    "entity1_index": "cara.semperger@enron.com/account",
    "verb": "emailed",
    "verb_category": "emailed/communicated",
    "entity2": "will.smith@enron.com",
    "entity2_index": "will.smith@enron.com/account",
    "time_start": "2001-07-09T14:33:32",
    "assoc_type": "Event"
  }
],
```



```
"metadata": {
  "_FILE_METADATA_": [
    [
      {
        "metadata": {
          "Creation-Date": [
            "2001-07-09T18:33:32Z"
          ],
          "subject": [
            "RE: Testing Preschedule workspace"
          ],
          "Message-From": [
            "cara.semperger@enron.com"
          ],
          "Author": [
            "cara.semperger@enron.com"
          ],
          "Message-To": [
            "will.smith@enron.com"
          ],
          "date": [
            "2001-07-09T18:33:32Z"
          ],
          "Content-Type": [
            "message/rfc822"
          ]
        }
      ]
    ],
    "email_meta": [
      [
        {
          "Creation-Date": [
            "2001-07-09T18:33:32Z"
          ],
          "Message-To": [
            "will.smith@enron.com"
          ],
          "Content-Type": [
            "message/rfc822"
          ],
          "subject": [
            "RE: Testing Preschedule workspace"
          ],
          "date": [
            "2001-07-09T18:33:32Z"
          ],
          "Author": [
            "cara.semperger@enron.com"
          ],
          "Message-From": [
            "cara.semperger@enron.com"
          ]
        }
      ]
    ]
  ]
}
```

```
    ]
  }
}
```

Feed Source

The following code samples show how to create a basic source that extracts data from an RSS feed.

Sample RSS Feed Source

```
{
  "description" : "Article on Medical Issues",
  "extractType" : "Feed",
  "harvestBadSource" : false,
  "isApproved" : true,
  "isPublic" : true,
  "key" : "http://www.mayoclinic.com/rss/blog.xml",
  "mediaType" : "News",
  "modified" : "Oct 19, 2010 11:31:59 AM",
  "tags" : [
    "topic:healthcare",
    "industry:healthcare",
    "mayo clinic",
    "health"
  ],
  "title" : "MayoClinic: General Topics",
  "url" : "http://www.mayoclinic.com/rss/blog.xml"
  "useTextExtractor": "AlchemyAPI",
  "useExtractor": "OpenCalais"
}
```

Sample Output from RSS Feed Source Above

```
{
  "_id" : "4e1c8afa7d56bb818ed10f76",
  "created" : "1310493434159",
  "description" : "Clarify the role of carbohydrates in the Dr. Bernstein diet and
find a
  healthy eating plan that works for you.",
  "entities" : [
    {
      "actual_name" : "certified diabetes",
      "dimension" : "What",
      "disambiguous_name" : "certified diabetes",
      "doccount" : NumberLong(38),
      "frequency" : 3,
      "gazateer_index" : "certified diabetes/medicalcondition",
      "relevance" : "0.711",
      "totalfrequency" : NumberLong(114),
      "type" : "MedicalCondition"
    },
    {
```

```

    "actual_name" : "Diabetes Unit",
    "dimension" : "Who",
    "disambiguous_name" : "Diabetes Unit",
    "doccount" : NumberLong(38),
    "frequency" : 1,
    "gazateer_index" : "diabetes unit/organization",
    "relevance" : "0.235",
    "totalfrequency" : NumberLong(38),
    "type" : "Organization"
  },
  {
    "actual_name" : "Mayo Clinic",
    "dimension" : "What",
    "disambiguous_name" : "Mayo Clinic",
    "doccount" : NumberLong(514),
    "frequency" : 2,
    "gazateer_index" : "mayo clinic/facility",
    "relevance" : "0.305",
    "totalfrequency" : NumberLong(1033),
    "type" : "Facility"
  },
  {
    "actual_name" : "Rochester",
    "dimension" : "Where",
    "disambiguous_name" : "Rochester,Minnesota,United States",
    "doccount" : NumberLong(345),
    "frequency" : 2,
    "gazateer_index" : "rochester,minnesota,united states/city",
    "geotag" : {
      "latitude" : "44.0217",
      "longitude" : "-92.4697",
      "loc" : [
        44.0217,
        -92.4697
      ]
    },
    "linkdata" : [
      "http://d.opencalais.com/er/geo/city/ralg-geol/9d780656-b9a4-6789-bcaf-370cab32490"
    ],
    "relevance" : "0.305",
    "totalfrequency" : NumberLong(404),
    "type" : "City"
  },
  {
    "actual_name" : "Minnesota",
    "dimension" : "Where",
    "disambiguous_name" : "Minnesota,United States",
    "doccount" : NumberLong(2103),
    "frequency" : 2,
    "gazateer_index" : "minnesota,united states/provinceorstate",
    "geotag" : {
      "latitude" : "46.0",
      "longitude" : "-94.0",
      "loc" : [
        46,
        -94
      ]
    },
  },

```

```

"linkdata" : [
"http://d.opencalais.com/er/geo/provinceorstate/ralg-geol/b99d1bcd-ec35-113e-54ad-4d6c
44682cel"
],
"relevance" : "0.305",
"totalfrequency" : NumberLong(3134),
"type" : "ProvinceOrState"
},
{
"actual_name" : "R.N.",
"dimension" : "Who",
"disambiguous_name" : "R.N.",
"doccount" : NumberLong(108),
"frequency" : 1,
"gazateer_index" : "r.n./position",
"relevance" : "0.138",
"totalfrequency" : NumberLong(130),
"type" : "Position"
},
{
"actual_name" : "coordinator",
"dimension" : "Who",
"disambiguous_name" : "coordinator",
"doccount" : NumberLong(220),
"frequency" : 1,
"gazateer_index" : "coordinator/position",
"relevance" : "0.235",
"totalfrequency" : NumberLong(238),
"type" : "Position"
},
{
"actual_name" : "Division of Endocrinology, Diabetes, Metabolism, &
Nutrition",
"dimension" : "Who",
"disambiguous_name" : "Division of Endocrinology, Diabetes, Metabolism, &
Nutrition",
"doccount" : NumberLong(38),
"frequency" : 2,
"gazateer_index" : "division of endocrinology, diabetes, metabolism, &
nutrition/organization",
"relevance" : "0.305",
"totalfrequency" : NumberLong(76),
"type" : "Organization"
},
{
"actual_name" : "self-management",
"dimension" : "What",
"disambiguous_name" : "self-management",
"doccount" : NumberLong(72),
"frequency" : 1,
"gazateer_index" : "self-management/medicaltreatment",
"relevance" : "0.076",
"totalfrequency" : NumberLong(78),
"type" : "MedicalTreatment"
},
{
"actual_name" : "Peggy Moreland",
"dimension" : "Who",

```

```

    "disambiguous_name" : "Peggy Moreland",
    "doccount" : NumberLong(38),
    "frequency" : 3,
    "gazateer_index" : "peggy moreland/person",
    "relevance" : "0.358",
    "totalfrequency" : NumberLong(114),
    "type" : "Person"
  },
  {
    "actual_name" : "University of Phoenix",
    "dimension" : "What",
    "disambiguous_name" : "University of Phoenix",
    "doccount" : NumberLong(64),
    "frequency" : 1,
    "gazateer_index" : "university of phoenix/facility",
    "relevance" : "0.102",
    "totalfrequency" : NumberLong(68),
    "type" : "Facility"
  },
  {
    "actual_name" : "registered nurse",
    "dimension" : "Who",
    "disambiguous_name" : "registered nurse",
    "doccount" : NumberLong(69),
    "frequency" : 1,
    "gazateer_index" : "registered nurse/position",
    "relevance" : "0.335",
    "totalfrequency" : NumberLong(70),
    "type" : "Position"
  },
  {
    "actual_name" : "American Diabetes Association",
    "dimension" : "Who",
    "disambiguous_name" : "American Diabetes Association",
    "doccount" : NumberLong(120),
    "frequency" : 1,
    "gazateer_index" : "american diabetes association/organization",
    "relevance" : "0.102",
    "totalfrequency" : NumberLong(226),
    "type" : "Organization"
  },
  {
    "actual_name" : "N.\n Peggy Moreland",
    "dimension" : "Who",
    "disambiguous_name" : "N.\n Peggy Moreland",
    "doccount" : NumberLong(38),
    "frequency" : 1,
    "gazateer_index" : "n.\n peggy moreland/person",
    "relevance" : "0.146",
    "totalfrequency" : NumberLong(38),
    "type" : "Person"
  },
  {
    "actual_name" : "insulin pump",
    "dimension" : "What",
    "disambiguous_name" : "insulin pump",
    "doccount" : NumberLong(52),
    "frequency" : 1,
    "gazateer_index" : "insulin pump/medicaltreatment",

```

```

    "relevance" : "0.184",
    "totalfrequency" : NumberLong(69),
    "type" : "MedicalTreatment"
  },
  {
    "actual_name" : "M.S.N.\n Peggy",
    "dimension" : "Who",
    "disambiguous_name" : "M.S.N.\n Peggy",
    "doccount" : NumberLong(38),
    "frequency" : 2,
    "gazateer_index" : "m.s.n.\n peggy/person",
    "relevance" : "0.482",
    "totalfrequency" : NumberLong(76),
    "type" : "Person"
  },
  {
    "actual_name" : "Nancy Klobassa Davidson",
    "dimension" : "Who",
    "disambiguous_name" : "Nancy Klobassa Davidson",
    "doccount" : NumberLong(38),
    "frequency" : 5,
    "gazateer_index" : "nancy klobassa davidson/person",
    "relevance" : "0.659",
    "totalfrequency" : NumberLong(190),
    "type" : "Person"
  },
  {
    "actual_name" : "University of Phoenix",
    "dimension" : "Who",
    "disambiguous_name" : "University of Phoenix",
    "doccount" : NumberLong(65),
    "frequency" : 1,
    "gazateer_index" : "university of phoenix/organization",
    "relevance" : "0.102",
    "totalfrequency" : NumberLong(69),
    "type" : "Organization"
  },
  {
    "actual_name" : "diabetes",
    "dimension" : "What",
    "disambiguous_name" : "diabetes",
    "doccount" : NumberLong(1702),
    "frequency" : 3,
    "gazateer_index" : "diabetes/medicalcondition",
    "relevance" : "0.733",
    "totalfrequency" : NumberLong(4284),
    "type" : "MedicalCondition"
  },
  {
    "actual_name" : "insulin therapy",
    "dimension" : "What",
    "disambiguous_name" : "insulin therapy",
    "doccount" : NumberLong(53),
    "frequency" : 1,
    "gazateer_index" : "insulin therapy/medicaltreatment",
    "relevance" : "0.235",
    "totalfrequency" : NumberLong(57),
    "type" : "MedicalTreatment"
  },

```

```

{
  "actual_name" : "American Association of Diabetes Educators",
  "dimension" : "Who",
  "disambiguous_name" : "American Association of Diabetes Educators",
  "doccount" : NumberLong(38),
  "frequency" : 1,
  "gazateer_index" : "american association of diabetes educators/organization",
  "relevance" : "0.102",
  "totalfrequency" : NumberLong(38),
  "type" : "Organization"
},
{
  "actual_name" : "member",
  "dimension" : "Who",
  "disambiguous_name" : "member",
  "doccount" : NumberLong(795),
  "frequency" : 1,
  "gazateer_index" : "member/position",
  "relevance" : "0.102",
  "totalfrequency" : NumberLong(881),
  "type" : "Position"
}
],
"events" : [
{
  "entity1" : "nancy klobassa davidson",
  "entity1_index" : "nancy klobassa davidson/person",
  "verb" : "be",
  "verb_category" : "generic relations",
  "entity2" : "coordinator of the diabetes unit's intensive insulin therapy
    program within the division of endocrinology",
  "event_type" : "Summary"
},
{
  "entity1" : "peggy moreland",
  "entity1_index" : "peggy moreland/person",
  "verb" : "work",
  "verb_category" : "generic relations",
  "event_type" : "Summary"
},
{
  "entity1" : "nancy klobassa davidson",
  "entity1_index" : "nancy klobassa davidson/person",
  "verb" : "current",
  "verb_category" : "career",
  "entity2" : "coordinator",
  "entity2_index" : "coordinator/position",
  "event_type" : "Fact"
},
{
  "entity1" : "m.s.n.\n peggy",
  "entity1_index" : "m.s.n.\n peggy/person",
  "verb" : "current",
  "verb_category" : "career",
  "entity2" : "r.n.",
  "entity2_index" : "r.n./position",
  "event_type" : "Fact"
}
],
{

```

```
"entity1" : "peggy moreland",
"entity1_index" : "peggy moreland/person",
"verb" : "graduate",
"verb_category" : "generic relations",
"event_type" : "Summary"
},
{
"entity1" : "n.\n peggy moreland",
"entity1_index" : "n.\n peggy moreland/person",
"verb" : "be",
"verb_category" : "generic relations",
"entity2" : "a certified diabetes educator",
"event_type" : "Summary"
},
{
"entity1" : "nancy klobassa davidson",
"entity1_index" : "nancy klobassa davidson/person",
"verb" : "be",
"verb_category" : "generic relations",
"entity2" : "a certified diabetes educator",
"event_type" : "Summary"
},
{
"entity1" : "peggy moreland",
"entity1_index" : "peggy moreland/person",
"verb" : "current",
"verb_category" : "career",
"entity2" : "r.n.",
"entity2_index" : "r.n./position",
"event_type" : "Fact"
},
{
"entity1" : "nancy klobassa davidson",
"entity1_index" : "nancy klobassa davidson/person",
"verb" : "be",
"verb_category" : "generic relations",
"entity2" : "a registered nurse",
"event_type" : "Summary"
},
{
"entity1" : "nancy klobassa davidson",
"entity1_index" : "nancy klobassa davidson/person",
"verb" : "work",
"verb_category" : "generic relations",
"event_type" : "Summary"
},
{
"entity1" : "nancy klobassa davidson",
"entity1_index" : "nancy klobassa davidson/person",
"verb" : "current",
"verb_category" : "career",
"entity2" : "registered nurse",
"entity2_index" : "registered nurse/position",
"event_type" : "Fact"
},
{
"entity1" : "peggy moreland",
"entity1_index" : "peggy moreland/person",
"verb_category" : "person education",
```




```
    "entity2" : "master of science",
    "event_type" : "Summary"
  },
  {
    "entity1" : "peggy moreland",
    "entity1_index" : "peggy moreland/person",
    "verb" : "current",
    "verb_category" : "career",
    "entity2" : "member",
    "entity2_index" : "member/position",
    "event_type" : "Fact"
  }
],
"groupids" : [ "4c927585d591d31d7b37097a" ],
"index" : "doc_4c927585d591d31d7b37097a",
"mediaType" : "News",
"modified" : "1310493434159",
"publishedDate" : "Fri Jul 08 01:00:00 EDT 2011",
"source" : "MayoClinic: General Topics",
"sourceKey" : "http.www.mayoclinic.com.rss.blog.xml",
"tags" : [
  "topic:healthcare",
  "industry:healthcare",
  "mayo clinic",
  "health"
],
"title" : "Dr. Bernstein diet and beyond",
```

```
"url" : "http://www.mayoclinic.com/health/dr-bernstein-diet/MY01817/rss=5"
}
```

File source to ingest the results of the query results (from either the API or the GUI export)

Source

 Note needs to be run on v0.1.7714+ since earlier versions have a bug that will hang the harvester/API test if 1) the max number of docs is exceeded and 2) the data/documents is not the last JSON element in the file (it won't normally be unless manually edited)

See [here](#) for details on bulk export of data from the API.

This version has a few noteworthy limitations:

- Tags and mediaType cannot be dynamically specified, so you should segment the files into different directories where possible and then hardcode them in the source fields (see TODO below)
- The fullText is not generated - this is a fundamental limitation since it is not in the exported JSON. Note that the title, description, entities, and associations are all indexed, so documents will normally have a decent "full text signature" regardless
 - *(If needed, fullText could be populated with a more complex "Feed" source that used "rss.searchConfig" but was otherwise similar to this)*
- The metadata from the old document is stored in metadata.prevmeta (this cannot be worked around unless you know a priori the metadata fieldnames, in which case it can be easily fixed in the "unstructuredAnalysis,meta" block below)
- Associations are not copied across. This can be easily added if needed.

```
{
  "description": "Re-imports the documents output from Infinit.e's query API or GUI
export",
  "extractType": "File",
  "file": {
    "type": "json",
    "XmlPrimaryKey": "url",
    "XmlRootLevelValues": [
      "documents",
      "data"
    ],
    "XmlSourceName": ""
  },
  "isPublic": true,
  "mediaType": "TODO_INSERT_TYPE_HERE",
  "searchIndexFilter": {
    "metadataFieldList": ""
  },
  "structuredAnalysis": {
    "associations": [{
      "assoc_type": "$assoc_type",
      "entity1": "$entity1",
      "entity1_index": "$entity1_index",
      "entity2": "$entity2",
      "entity2_index": "$entity2_index",
      "iterateOver": "json.associations",
      "verb": "$verb",
      "verb_category": "$verb_category"
    }],
    "description": "$metadata.json.description",
    "entities": [
      {
        "actual_name": "$SCRIPT( return _iterator.actual_name == null ?
```

```
_iterator.disambiguated_name : _iterator.actual_name; )",
    "dimension": "$dimension",
    "disambiguated_name": "$disambiguated_name",
    "geotag": {
        "lat": "$geotag.lat",
        "lon": "$geotag.lon"
    },
    "iterateOver": "json.entities",
    "linkdata": "$linkdata",
    "ontology_type": "$ontology_type",
    "relevance": "$relevance",
    "frequency": "$frequency",
    "sentiment": "$sentiment",
    "type": "$type"
}
],
"metadataFields": "-json",
"publishedDate": "$SCRIPT( return new
Date(_doc.metadata.json[0].publishedDate).toString(); )",
"title": "$metadata.json.title"
},
"tags": [
    "TODO_INSERT_TAGS_HERE"
],
"title": "Infinite Import Template",
"unstructuredAnalysis": {
    "meta": [
        {
            "context": "First",
            "fieldName": "prevmeta",
            "flags": "m",
            "script": "var retval = _metadata.json[0].metadata; retval;",
            "scriptlang": "javascript"
        }
    ]
}
]
```

```
    },
    "url": "file:///path/to/files/"
}
```

GNIP source gallery

Sample document

Twitter document

```
{
  "id": "tag:search.twitter.com,2005:266601489475186688",
  "objectType": "activity",
  "actor": {
    "objectType": "person",
    "id": "id:twitter.com:835627776",
    "link": "http://www.twitter.com/FocalCRM",
    "displayName": "CRM Buddy",
    "postedTime": "2012-09-20T13:59:56.000Z",
    "image":
"http://a0.twimg.com/profile_images/2630355549/8cad59efadd57283dbb159332336744_normal
.jpeg",
    "summary": "",
    "links": [
      {
        "href": null,
        "rel": "me"
      }
    ],
    "friendsCount": 0,
    "followersCount": 245,
    "listedCount": 6,
    "statusesCount": 3688,
    "twitterTimeZone": null,
    "verified": false,
    "utcOffset": null,
    "preferredUsername": "FocalCRM",
    "languages": [
      "en"
    ]
  },
  "verb": "post",
  "postedTime": "2012-11-08T18:02:02.000Z",
  "generator": {
    "displayName": "dlvr.it",
    "link": "http://dlvr.it"
  },
  "provider": {
    "objectType": "service",
    "displayName": "Twitter",
    "link": "http://www.twitter.com"
  },
  "link": "http://twitter.com/FocalCRM/statuses/266601489475186688",
  "body": "Amex Teams With Halo 4 on Master Chief Incentives http://t.co/IvwmjJyV
#crm",
  "object": {
```

```
    "objectType": "note",
    "id": "object:search.twitter.com,2005:266601489475186688",
    "summary": "Amex TeamsWith Halo 4 on Master Chief Incentives
http://t.co/IvwmjJyV #crm",
    "link": "http://twitter.com/FocalCRM/statuses/266601489475186688",
    "postedTime": "2012-11-08T18:02:02.000Z"
  },
  "twitter_entities": {
    "urls": [
      {
        "display_url": "dlvr.it/2S6sjV",
        "indices": [
          50,
          70
        ],
        "expanded_url": "http://dlvr.it/2S6sjV",
        "url": "http://t.co/IvwmjJyV"
      }
    ],
    "user_mentions": [],
    "hashtags": [
      {
        "text": "crm",
        "indices": [
          71,
          75
        ]
      }
    ]
  }
},
"retweetCount": 0,
"gnip": {
  "language": {
    "value": "en"
  }
},
"matching_rules": [
  {
    "value": "halo 4",
    "tag": null
  }
],
"klout_score": 48,
"urls": [
  {
    "url": "http://t.co/IvwmjJyV",
    "expanded_url": "http://www.crmbuyer.com/rsstory/76578.html"
  }
]
```

```
    ]
  }
}
```

Source

```
//TODO (INF-1865): need to distinguish between "tweets_to" and "retweets"...
//TODO (INF-1865): looks like body is HTML encoded
//TODO (INF-1865): aggregate sentiment vs user?
//TODO (INF-1865): distinguish between tweets and mentions
{
  "description": "A large set of tweets related to Super Storm Sandy",
  "extractType": "File",
  "extractorOptions": {
    "app.alchemyapi-metadata.batchSize": 100,
    "app.alchemyapi-metadata.numKeywords": 5,
    "app.alchemyapi-metadata.strict": "true"
  },
  "file": {
    "XmlPrimaryKey": "link",
    "XmlSourceName": "",
    "XmlRootLevelValues": [],
    "domain": "XXX",
    "password": "XXX",
    "username": "XXX"
  },
  "isApproved": true,
  "isPublic": false,
  "mediaType": "Social",
  "structuredAnalysis": {
    "rejectDocCriteria": "$SCRIPT( if (null == _doc.metadata.json[0].link || null ==
_doc.metadata.json[0].object) return 'reject'; )",
    "metadataFieldList": "",
    "docGeo": {
      "lat": "$SCRIPT( try {return _doc.metadata.json[0].geo.coordinates[0];} catch (err)
{return '';} )",
      "lon": "$SCRIPT( try {return _doc.metadata.json[0].geo.coordinates[1];} catch (err)
{return '';} )"
    },
    "associations": [
      {
        "assoc_type": "Event",
        "creationCriteriaScript": "$SCRIPT( return (null !=
_doc.metadata.json[0].object.actor); )",
        "entity1_index": "$SCRIPT( return
_doc.metadata.json[0].actor.preferredUsername + '/twitterhandle'; )",
        "entity2_index": "$SCRIPT( return
_doc.metadata.json[0].object.actor.preferredUsername + '/twitterhandle'; )",
        "verb": "retweets",
        "verb_category": "retweets"
      },
      {
        "assoc_type": "Event",
        "creationCriteriaScript": "$SCRIPT( return (null !=
_doc.metadata.json[0].object.actor) && (null !=
_doc.metadata.json[0].object.actor.location); )",
```

```

        "entity1_index": "$SCRIPT( return
_doc.metadata.json[0].object.actor.preferredUsername + '/twitterhandle;)',
        "entity2_index": "$SCRIPT( return
_doc.metadata.json[0].object.actor.location.displayName+ '/location;)',
        "verb": "twitter_location",
        "verb_category": "twitter_location"
    },
    {
        "assoc_type": "Event",
        "entity1_index": "$SCRIPT( return
_doc.metadata.json[0].actor.preferredUsername + '/twitterhandle;)',
        "entity2_index": "$SCRIPT( return _iterator.text + '/hashtag; )",
        "iterateOver": "json.twitter_entities.hashtags",
        "verb": "tweets_about",
        "verb_category": "tweets_about"
    },
    {
        "assoc_type": "Event",
        "entity1_index": "$SCRIPT( return
_doc.metadata.json[0].actor.preferredUsername + '/twitterhandle;)',
        "entity2_index": "$SCRIPT( return _iterator.screen_name +
'/twitterhandle; )",
        "iterateOver": "json.twitter_entities.user_mentions",
        "verb": "tweets_to",
        "verb_category": "tweets_to"
    },
    {
        "assoc_type": "Event",
        "entity1_index": "$SCRIPT( return
_doc.metadata.json[0].actor.preferredUsername + '/twitterhandle;)',
        "entity2_index": "$SCRIPT( return _iterator.expanded_url + '/url; )",
        "iterateOver": "json.gnip.urls",
        "verb": "tweets_link",
        "verb_category": "tweets_link"
    }
],
"description": "$metadata.json.body",
"entities": [
{
    "actual_name": "$metadata.json.actor.displayName",
    "dimension": "Who",
    "disambiguated_name": "$metadata.json.actor.preferredUsername",
    "linkdata": "$metadata.json.actor.link",
    "type": "TwitterHandle"
},
{
    "iterateOver": "json.twitter_entities.user_mentions",
    "actual_name": "$SCRIPT(return _iterator.name;)",
    "dimension": "Who",
    "disambiguated_name": "$SCRIPT(return _iterator.screen_name;)",
    "linkdata": "$SCRIPT(return 'http://www.twitter.com/' +
_iterator.screen_name;)",
    "type": "TwitterHandle"
},
{
    "actual_name": "$metadata.json.object.actor.displayName",
    "dimension": "Who",
    "disambiguated_name": "$metadata.json.object.actor.preferredUsername",
    "linkdata": "$metadata.json.object.actor.link",

```

```

    "type": "TwitterHandle"
  },
  {
    "dimension": "Where",
    "disambiguated_name": "$metadata.json.actor.location.displayName",
    "geotag": {
      "city": "$SCRIPT( return getAddressVal(
_doc.metadata.json[0].actor.location.displayName, 0 ) )",
      "stateProvince": "$SCRIPT( return getRegion(getAddressVal(
_doc.metadata.json[0].actor.location.displayName, 1 ) ) )",
      "countryCode" : "US",
      "alternatives": [
        {
          "stateProvince": "$SCRIPT( return getRegion(getAddressVal(
_doc.metadata.json[0].actor.location.displayName, 1 ) ) )",
          "countryCode" : "US"
        }
      ]
    },
    "type": "Location"
  },
  {
    "dimension": "Where",
    "disambiguated_name": "$metadata.json.object.actor.location.displayName",
    "type": "Location"
  },
  {
    "disambiguated_name": "$SCRIPT(return _iterator.text;)",
    "iterateOver": "json.twitter_entities.hashtags",
    "type": "HashTag"
  },
  {
    "actual_name": "$SCRIPT(return _iterator.url)",
    "disambiguated_name": "$SCRIPT(return _iterator.expanded_url;)",
    "iterateOver": "json.gnip.urls",
    "type": "URL"
  }
],
"fullText": "$metadata.json.body",
"script": "function getAddressVal( addressStr, number) { try { var addressArray
= addressStr.split(/ *, */); if (addressArray != null && addressArray.length > 0) { if
(addressArray[number].toLowerCase()=='ny') { return 'new york'; } else if
(addressArray[number].toLowerCase()=='long island' ||
addressArray[number].toLowerCase()=='li') { return 'medford'; } else { return
addressArray[number]; } } else { return ''; } } catch (err) { return ''; } } function
getRegion( code ) { if (code.toLowerCase()=='ny') {return 'New York';} else if
(code.toLowerCase()=='nj') {return 'New Jersey';} else if (code.toLowerCase()=='ct')
{return 'Connecticut';} else if (code.toLowerCase()=='md') {return 'Maryland';} else
if (code.toLowerCase()=='va') {return 'Virginia';} else if (code.toLowerCase()=='pa')
{return 'Pennsylvania';} else if (code.toLowerCase()=='nj') {return 'New Jersey';}
else {return 'New York';} }",
  "scriptEngine": "javascript",
  "title": "$metadata.json.body",
  "url": "$metadata.json.link",
  "publishedDate": "$SCRIPT(return
_doc.metadata.json[0].postedTime.replace(/.[0-9]{3}Z/, 'Z'))"
},
"tags": [
  "twitter",

```



```
    "gnip"  
  ],  
  "title": "Super Storm Sandy - Twitter: SANDY_SUBSTRING",  
  "url": "smb://HOST:139/SHARE/PATH/TO/",
```

```
"useExtractor": "AlchemyAPI-metadata",
"useTextExtractor": "none"
}
```

Sample output

```
{
  "associations": [
    {
      "assoc_type": "Event",
      "entity1": "focalcrm",
      "entity1_index": "focalcrm/twitterhandle",
      "entity2": "crm",
      "entity2_index": "crm/hashtag",
      "verb": "tweets_about",
      "verb_category": "tweets_about"
    },
    {
      "assoc_type": "Event",
      "entity1": "focalcrm",
      "entity1_index": "focalcrm/twitterhandle",
      "entity2": "http://www.crmbuyer.com/rsstory/76578.html",
      "entity2_index": "http://www.crmbuyer.com/rsstory/76578.html/url",
      "verb": "tweets_link",
      "verb_category": "tweets_link"
    }
  ],
  "communityId": ["506dc16dfbf042893dd6b8f2"],
  "created": "May 16, 2013 12:28:09 PM UTC",
  "description": "Amex Teams With Halo 4 on Master Chief Incentives
http://t.co/IvwmjJyV #crm",
  "entities": [
    {
      "actual_name": "CRM Buddy",
      "dimension": "Who",
      "disambiguated_name": "FocalCRM",
      "doccount": 0,
      "frequency": 1,
      "index": "focalcrm/twitterhandle",
      "linkdata": ["http://www.twitter.com/FocalCRM"],
      "relevance": 0,
      "totalfrequency": -1,
      "type": "TwitterHandle"
    },
    {
      "actual_name": "crm",
      "dimension": "What",
      "disambiguated_name": "crm",
      "doccount": 0,
      "frequency": 1,
      "index": "crm/hashtag",
      "relevance": 0,
      "totalfrequency": -1,
      "type": "HashTag"
    }
  ]
}
```

```
"actual_name": "http://t.co/IvwmjJyV",
"dimension": "What",
"disambiguated_name": "http://www.crmbuyer.com/rsstory/76578.html",
"doccount": 0,
"frequency": 1,
"index": "http://www.crmbuyer.com/rsstory/76578.html/url",
"relevance": 0,
"totalfrequency": -1,
"type": "URL"
},
{
  "actual_name": "Amex Teams",
  "dimension": "What",
  "disambiguated_name": "Amex Teams",
  "doccount": -1,
  "frequency": 1,
  "index": "amex teams/keyword",
  "relevance": 0.758636,
  "sentiment": 0.160753,
  "totalfrequency": -1,
  "type": "Keyword"
},
{
  "actual_name": "Halo",
  "dimension": "What",
  "disambiguated_name": "Halo",
  "doccount": -1,
  "frequency": 1,
  "index": "halo/keyword",
  "relevance": 0.461833,
  "sentiment": 0.168822,
  "totalfrequency": -1,
  "type": "Keyword"
},
{
  "actual_name": "Master Chief Incentives",
  "dimension": "What",
  "disambiguated_name": "Master Chief Incentives",
  "doccount": -1,
  "frequency": 1,
  "index": "master chief incentives/keyword",
  "relevance": 0.981457,
  "sentiment": 0.168876,
  "totalfrequency": -1,
  "type": "Keyword"
},
{
  "actual_name": "http://t.co/IvwmjJyV",
  "dimension": "What",
  "disambiguated_name": "http://t.co/IvwmjJyV",
  "doccount": -1,
  "frequency": 1,
  "index": "http://t.co/ivwmjjyv/keyword",
  "relevance": 0.212007,
  "sentiment": 0.126168,
  "totalfrequency": -1,
  "type": "Keyword"
},
{
```

```

        "actual_name": "crm",
        "dimension": "What",
        "disambiguated_name": "crm",
        "doccount": -1,
        "frequency": 1,
        "index": "crm/keyword",
        "relevance": 0.404086,
        "sentiment": 0.103838,
        "totalfrequency": -1,
        "type": "Keyword"
    }
],
"mediaType": ["Social"],
"metadata": {"json": [{"
    "actor": {
        "displayName": "CRM Buddy",
        "followersCount": "245",
        "friendsCount": "0",
        "id": "id:twitter.com:835627776",
        "image":
"http://a0.twimg.com/profile_images/2630355549/8cad59efadd57283dbb159332336744_normal
.jpeg",
        "languages": ["en"],
        "link": "http://www.twitter.com/FocalCRM",
        "links": [{"rel": "me"}],
        "listedCount": "6",
        "objectType": "person",
        "postedTime": "2012-09-20T13:59:56.000Z",
        "preferredUsername": "FocalCRM",
        "statusesCount": "3688",
        "summary": "",
        "verified": "false"
    },
    "body": "Amex Teams With Halo 4 on Master Chief Incentives
http://t.co/IvwmjJyV #crm",
    "generator": {
        "displayName": "dlvr.it",
        "link": "http://dlvr.it"
    },
    "gnip": {
        "klout_score": "48",
        "language": {"value": "en"},
        "matching_rules": [{"value": "halo 4"}],
        "urls": [{
            "expanded_url": "http://www.crmbuyer.com/rsstory/76578.html",
            "url": "http://t.co/IvwmjJyV"
        }]
    },
    "id": "tag:search.twitter.com,2005:266601489475186688",
    "link": "http://twitter.com/FocalCRM/statuses/266601489475186688",
    "object": {
        "id": "object:search.twitter.com,2005:266601489475186688",
        "link": "http://twitter.com/FocalCRM/statuses/266601489475186688",
        "objectType": "note",
        "postedTime": "2012-11-08T18:02:02.000Z",
        "summary": "Amex Teams With Halo 4 on Master Chief Incentives
http://t.co/IvwmjJyV #crm"
    },
    "objectType": "activity",

```

```
"postedTime": "2012-11-08T18:02:02.000Z",
"provider": {
  "displayName": "Twitter",
  "link": "http://www.twitter.com",
  "objectType": "service"
},
"retweetCount": "0",
"twitter_entities": {
  "hashtags": [{
    "indices": [
      "71",
      "75"
    ],
    "text": "crm"
  }],
  "urls": [{
    "display_url": "dlvr.it/2S6sjV",
    "expanded_url": "http://dlvr.it/2S6sjV",
    "indices": [
      "50",
      "70"
    ],
    "url": "http://t.co/IvwmjJyV"
  }],
  "user_mentions": []
},
"verb": "post"
}],
"modified": "Nov 8, 2012 06:02:44 PM UTC",
"publishedDate": "Nov 8, 2012 06:02:02 PM UTC",
"source": ["gnip test"],
"sourceKey": [".mnt.fileshare.datasift.gnip."],
"sourceUrl": "file:/mnt/filesshare/datasift/gnip/gnip.json",
"tags": [
  "twitter",
  "gnip"
],
"title": "Amex Teams With Halo 4 on Master Chief Incentives http://t.co/IvwmjJyV"
```

```
#crm",
  "url": "http://twitter.com/FocalCRM/statuses/266601489475186688"
}
```

Log File Source Gallery

Input format sample

```
Date,Device,SrcIP,dstIP,Alert,Country
SCANNER_1,2012-01-01T13:43:00,10.0.0.1,66.66.66.66,DUMMY_ALERT_TYPE_1,United States
SCANNER_2,2012-02-01T14:21:00,SCANNER_2,10.0.0.2,66.66.66.66,DUMMY_ALERT_TYPE_2,United
Kingdom
SCANNER_3,2012-03-01T15:17:00,10.0.0.1,99.66.99.66,DUMMY_ALERT_TYPE_3,Netherlands
```

Source #1a - fileshare, manual parsing

```
{
  "description": "For cyber demo",
  "extractType": "File",
  "file": {
    "XmlRootLevelValues": [],
    "domain": "DOMAIN",
    "password": "PASSWORD",
    "type": "csv",
    "username": "USER"
  },
  "isPublic": false,
  "mediaType": "Log",
  "searchCycle_secs": 3600,
  "searchIndexFilter": {
    "metadataFieldList": ""
  },
  "structuredAnalysis": {
    "associations": [
      {
        "entity1": "$metadata.info.dstIP",
        "entity2": "$metadata.info.srcIP",
        "geo_index": "$SCRIPT( return _doc.metadata.info[0].country +
'/country'; )",
        "time_start": "$SCRIPT( return _doc.metadata.info[0].date; )",
        "verb": "$SCRIPT( return _doc.metadata.info[0].alert; )",
        "verb_category": "$SCRIPT( return _doc.metadata.info[0].alert; )"
      }
    ],
    "entities": [
      {
        "dimension": "What",
        "disambiguated_name": "$metadata.info.srcIP",
        "type": "PrivateIP"
      },
      {
        "dimension": "What",
        "disambiguated_name": "$metadata.info.dstIP",
```

```

        "geotag": {
            "country": "$SCRIPT( return _doc.metadata.info[0].country; )"
        },
        "ontology_type": "country",
        "type": "PublicIP"
    },
    {
        "actual_name": "$metadata.info.country",
        "dimension": "Where",
        "disambiguated_name": "$SCRIPT( return _doc.metadata.info[0].country;
)",
        "geotag": {
            "country": "$SCRIPT( return _doc.metadata.info[0].country; )"
        },
        "ontology_type": "country",
        "type": "Country"
    },
    {
        "dimension": "What",
        "disambiguated_name": "$metadata.info.device",
        "type": "Sensor"
    },
    {
        "dimension": "What",
        "disambiguated_name": "$metadata.info.alert",
        "type": "AlertType"
    }
],
"publishedDate": "$SCRIPT( return _doc.metadata.info[0].date; )",
"script": "",
"scriptEngine": "javascript",
"title": "$metadata.info.alert @ $metadata.info.date [$metadata.info.device]:
$metadata.info.dstIP -> $metadata.info.srcIP"
},
"tags": [
    "cyber",
    "structured"
],
"title": "Cyber Logs Test",
"unstructuredAnalysis": {
    "meta": [
        {
            "context": "First",
            "fieldName": "info",
            "script": "var info = decode(text); info;",
            "scriptlang": "javascript"
        }
    ],
    "script": "function decode(x)\n{\n    var info = {};\n    \n    var rec =
x.split(',');\n    \n    info.device = rec[0];\n    info.date = rec[1];\n    info.srcIP =
rec[2];\n    info.dstIP = rec[3];\n    info.alert = rec[4];\n    info.country =
rec[5];\n    return info;\n}",
    "simpleTextCleanser": [
        {
            "field": "fullText",
            "flags": "md",
            "replacement": " , ",
            "script": ",,",
            "scriptlang": "regex"
        }
    ]
}

```

```
    },
    {
      "field": "description",
      "flags": "md",
      "replacement": " , ",
      "script": ",",
      "scriptlang": "regex"
    }
  ]
},
"useExtractor": "none",
```



```
"useTextExtractor": "none",
"url": "smb://FILESHARE:139/cyber_logs/"
}
```

Source #1b - fileshare, automated parsing

```
{
  "description": "For cyber demo",
  "extractType": "File",
  "file": {
    "XmlRootLevelValues": [ "device", "date", "srcIP", "dstIP", "alert", "country"
  ],
  "XmlIgnoreValues": [ "device,date,srcIP" ],
  "domain": "DOMAIN",
  "password": "PASSWORD",
  "type": "csv",
  "username": "USER"
},
"isPublic": false,
"mediaType": "Log",
"searchCycle_secs": 3600,
"searchIndexFilter": {
  "metadataFieldList": ""
},
"structuredAnalysis": {
  "associations": [
    {
      "entity1": "$metadata.csv.dstIP",
      "entity2": "$metadata.csv.srcIP",
      "geo_index": "$SCRIPT( return _doc.metadata.csv[0].country +
'/country'; )",
      "time_start": "$SCRIPT( return _doc.metadata.csv[0].date; )",
      "verb": "$SCRIPT( return _doc.metadata.csv[0].alert; )",
      "verb_category": "$SCRIPT( return _doc.metadata.csv[0].alert; )"
    }
  ],
  "entities": [
    {
      "dimension": "What",
      "disambiguated_name": "$metadata.csv.srcIP",
      "type": "PrivateIP"
    },
    {
      "dimension": "What",
      "disambiguated_name": "$metadata.csv.dstIP",
      "geotag": {
        "country": "$SCRIPT( return _doc.metadata.csv[0].country; )"
      },
      "ontology_type": "country",
      "type": "PublicIP"
    },
    {
      "actual_name": "$metadata.csv.country",
      "dimension": "Where",
      "disambiguated_name": "$SCRIPT( return _doc.metadata.csv[0].country;
```

```
)",
    "geotag": {
        "country": "$SCRIPT( return _doc.metadata.csv[0].country; )"
    },
    "ontology_type": "country",
    "type": "Country"
},
{
    "dimension": "What",
    "disambiguated_name": "$metadata.csv.device",
    "type": "Sensor"
},
{
    "dimension": "What",
    "disambiguated_name": "$metadata.csv.alert",
    "type": "AlertType"
}
],
"publishedDate": "$SCRIPT( return _doc.metadata.csv[0].date; )",
"script": "",
"scriptEngine": "javascript",
"title": "$metadata.csv.alert @ $metadata.csv.date [$metadata.csv.device]:
$metadata.csv.dstIP -> $metadata.csv.srcIP"
},
"tags": [
    "cyber",
    "structured"
],
"title": "Cyber Logs Test",
"useExtractor": "none",
```

```
"useTextExtractor": "none",
"url": "smb://FILESHARE:139/cyber_logs/"
}
```

Source #2a - web (including uploaded fileshares), manual parsing

It is slightly more complicated to parse CSV files over the Web, but still quite possible, using the searchConfig capability. Note that one neat trick is to upload a share to Infinite, and then use an API key to access the REST interface. Users can allocate themselves an API key from the [People Manager](#).

i Note that when accessing Web documents you must use "rss.extraUrls" and specify minimally "url" and "title" fields, and not the top-level "url" (otherwise the URL is treated as an RSS feed rather than a standalone web page)

```
{
  "description": "For cyber demo",
  "extractType": "Feed",
  "isPublic": false,
  "mediaType": "Log",
  "searchCycle_secs": 3600,
  "rss": {
    "extraUrls": [
      { "url":
"http://INFINITE_ENDPOINT/api/share/get/51ad28a440b4a4f0f757824c?infinite_api_key=API_
KEY" }
    ],
    "searchConfig": { "script": "var retVals = [];\nvar n = -1;\nvar url =
_doc.url.replace(/[?].*/,\\"");\nvar start = 0;\nwhile (start < text.length) {\n
var end = text.indexOf('\n', start);\n  if (end == -1) end = text.length;\n  var
line = text.substr(start,end-1);\n  start = end + 1;  \n  \n  n++;\n  if (0
== n) continue;\n  \n  var title = 'line #' + n.toString();\n  var url2 = url +
'#' + n.toString();\n  var fullText = line;\n  var retVal = { 'title':title,
'url':url2, 'fullText':line }; \n  retVals.push(retVal); \n}\nreturn retVals;\n  "
    },
    "searchIndexFilter": {
      "metadataFieldList": ""
    },
    "structuredAnalysis": {
      "associations": [
        {
          "entity1": "$metadata.info.dstIP",
          "entity2": "$metadata.info.srcIP",
          "geo_index": "$SCRIPT( return _doc.metadata.info[0].country +
'/country'; )",
          "time_start": "$SCRIPT( return _doc.metadata.info[0].date; )",
          "verb": "$SCRIPT( return _doc.metadata.info[0].alert; )",
          "verb_category": "$SCRIPT( return _doc.metadata.info[0].alert; )"
        }
      ],
      "entities": [
        {
          "dimension": "What",
          "disambiguated_name": "$metadata.info.srcIP",
          "type": "PrivateIP"
        },
        {
          "dimension": "What",
          "disambiguated_name": "$metadata.info.dstIP",
```

```

        "geotag": {
            "country": "$SCRIPT( return _doc.metadata.info[0].country; )"
        },
        "ontology_type": "country",
        "type": "PublicIP"
    },
    {
        "actual_name": "$metadata.info.country",
        "dimension": "Where",
        "disambiguated_name": "$SCRIPT( return _doc.metadata.info[0].country;
)",
        "geotag": {
            "country": "$SCRIPT( return _doc.metadata.info[0].country; )"
        },
        "ontology_type": "country",
        "type": "Country"
    },
    {
        "dimension": "What",
        "disambiguated_name": "$metadata.info.device",
        "type": "Sensor"
    },
    {
        "dimension": "What",
        "disambiguated_name": "$metadata.info.alert",
        "type": "AlertType"
    }
],
"publishedDate": "$SCRIPT( return _doc.metadata.info[0].date; )",
"script": "",
"scriptEngine": "javascript",
"title": "$metadata.info.alert @ $metadata.info.date [$metadata.info.device]:
$metadata.info.dstIP -> $metadata.info.srcIP"
},
"tags": [
    "cyber",
    "structured"
],
"title": "Cyber Logs Test",
"unstructuredAnalysis": {
    "meta": [
        {
            "context": "First",
            "fieldName": "info",
            "script": "var info = decode(text); info;",
            "scriptlang": "javascript"
        }
    ],
    "script": "function decode(x)\n{\n    var info = {};\n    \n    var rec =
x.split(',');\n    \n    info.device = rec[0];\n    info.date = rec[1];\n    info.srcIP =
rec[2];\n    info.dstIP = rec[3];\n    info.alert = rec[4];\n    info.country =
rec[5];\n    return info;\n}",
    "simpleTextCleanser": [
        {
            "field": "fullText",
            "flags": "md",
            "replacement": " , ",
            "script": ",,",
            "scriptlang": "regex"
        }
    ]
}

```

```
    },
    {
      "field": "description",
      "flags": "md",
      "replacement": " , ",
      "script": ",",
      "scriptlang": "regex"
    }
  ]
},
```

```
"useExtractor": "none",
"useTextExtractor": "none"
}
```

Note that the API key is not visible in any of the extracted documents (it is removed in the "searchConfig.script" code), and is also not visible in the source to anyone but the source owner and administrator (due to the "isPublic:false" field). As an alternative (from June 2013), a cookie can be used: (eg) "rss.httpFields": {"Cookie": "infinitecookie=api:API_KEY;"}

Source #2b - web (including uploaded fileshares), manual parsing using Java

As above, except "unstructuredAnalysis.script" will look like:

```
var parser = new Packages.au.com.bytecode.opencsv.CSVParser();
function decode(x)
{
    var rec = parser.parseLine(x.toString());
    var info = {};
    info.device = '' + rec[0];
    info.date = '' + rec[1];
    info.srcIP = '' + rec[2];
    info.dstIP = '' + rec[3];
    info.alert = '' + rec[4];
    info.country = '' + rec[5];
    return info;
}
```

(Note that the "" + <string-variable>" construct is necessary to convert from Java strings to javascript strings)

See the [opencsv documentation](#) for more details.

Output sample

(For source 1b, metadata.info is called metadata.csv)

```
{
  "associations": [{
    "assoc_type": "Event",
    "entity1": "66.66.66.66",
    "entity1_index": "66.66.66.66/publicip",
    "entity2": "10.0.0.1",
    "entity2_index": "10.0.0.1/privateip",
    "geo_index": "united states/country",
    "time_start": "2012-01-01T13:43:00",
    "verb": "DUMMY_ALERT_TYPE_1",
    "verb_category": "DUMMY_ALERT_TYPE_1"
  }],
  "communityId": ["506dc16dfbf042893dd6b8f2"],
  "created": "Jun 4, 2013 12:54:34 AM UTC",
  "entities": [
    {
      "actual_name": "10.0.0.1",
      "dimension": "What",
      "disambiguated_name": "10.0.0.1",
      "doccount": 0,
      "frequency": 1,
      "index": "10.0.0.1/privateip",
      "relevance": 0,
      "totalfrequency": -1,
    }
  ]
}
```

```

        "type": "PrivateIP"
    },
    {
        "actual_name": "66.66.66.66",
        "dimension": "What",
        "disambiguated_name": "66.66.66.66",
        "doccount": 0,
        "frequency": 1,
        "index": "66.66.66.66/publicip",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "PublicIP"
    },
    {
        "actual_name": "United States",
        "dimension": "Where",
        "disambiguated_name": "United States",
        "doccount": 0,
        "frequency": 1,
        "index": "united states/country",
        "ontology_type": "country",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "Country"
    },
    {
        "actual_name": "SCANNER_1",
        "dimension": "What",
        "disambiguated_name": "SCANNER_1",
        "doccount": 0,
        "frequency": 1,
        "index": "scanner_1/sensor",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "Sensor"
    },
    {
        "actual_name": "DUMMY_ALERT_TYPE_1",
        "dimension": "What",
        "disambiguated_name": "DUMMY_ALERT_TYPE_1",
        "doccount": 0,
        "frequency": 1,
        "index": "dummy_alert_type_1/alerttype",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "AlertType"
    }
],
"fullText": "SCANNER_1 , 2012-01-01T13:43:00 , 10.0.0.1 , 66.66.66.66 ,
DUMMY_ALERT_TYPE_1 , United States",
"mediaType": ["Log"],
"metadata": {"info": [{
    "alert": "DUMMY_ALERT_TYPE_1 ",
    "country": "United States",
    "date": "2012-01-01T13:43:00",
    "device": "SCANNER_1 ",
    "dstIP": "66.66.66.66",
    "srcIP": " 10.0.0.1"
}]},

```

```
"modified": "Jun 4, 2013 12:54:34 AM UTC",
"publishedDate": "January 1, 2012 13:43:00 PM UTC",
"source": ["Cyber Logs Test"],
"sourceKey": ["INFINITE_ENDPOINT.api.share.get.51ad28a440b4a4f0f757824c.25.26"],
"tags": [
  "cyber",
  "structured"
],
"title": "DUMMY_ALERT_TYPE_1 @ 2012-01-01T13:43:00 [SCANNER_1 ]: 66.66.66.66 ->
```



```
10.0.0.1",
  "url": "http://INFINITE_ENDPOINT/api/share/get/51ad28a440b4a4f0f757824c#1"
}
```


Simple web-hosted XML containing many documents

Overview

This is similar to the [WITS example](#), except that the XML is hosted on a web server instead of in a fileshare. Because the [Feed Harvester](#) does not have the same built-in decoding capabilities as the [File Harvester](#), this makes life a little bit more complicated.

Example data

<http://www.w3schools.com/xml/simple.xml>

 Note that when accessing Web documents you must use "rss.extraUrls" and specify minimally "url" and "title" fields, and not the top-level "url" (otherwise the URL is treated as an RSS feed rather than a standalone web page)

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>two of our famous Belgian Waffles with plenty of real maple
syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>light Belgian waffles covered with strawberries and whipped
cream</description>
    <calories>900</calories>
  </food>
  <food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>light Belgian waffles covered with an assortment of fresh berries
and whipped cream</description>
    <calories>900</calories>
  </food>
  <food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>thick slices made from our homemade sourdough bread</description>
    <calories>600</calories>
  </food>
  <food>
    <name>Homestyle Breakfast</name>
    <price>$6.95</price>
    <description>two eggs, bacon or sausage, toast, and our ever-popular hash
browns</description>
    <calories>950</calories>
  </food>
</breakfast_menu>
```

Source

Note the use of XPath to identify easily how to convert the top-level XML document into lots of little documents - the "rss,searchConfig.script" is then boilerplate and converts the XML into lots of small documents, with the "fullText" of each containing the JSON representation of the selected XML. This is then converted into metadata by the "unstructuredAnalysis" block. Normally a "structuredAnalysis" block would finally be used to set the per-document titles, descriptions, entities etc.

```
{
  "description": "wiy",
  "extractType": "Feed",
  "isPublic": true,
  "mediaType": "News",
  "rss": {
    "searchConfig": {
      "extraMeta": [
        {
          "context": "First",
          "fieldName": "convert_to_json",
          "flags": "o",
          "script": "//breakfast_menu/food[*]",
          "scriptlang": "xpath"
        }
      ],
      "script": "function convert_to_docs(jsonarray, url)\n{\n  var docs =\n  [];\n  for (var docIt in jsonarray) {\n    var predoc = jsonarray[docIt];\n    delete predoc.content;\n    var doc = {};\n    doc.url =\n    _doc.url.replace(/[?].*/,\"#\") + '#' + docIt;\n    doc.fullText = predoc;\n    doc.title = \"TBD\";\n    doc.description = \"TBD\";\n    docs.push(doc);\n  }\n  return docs;\n}\nvar docs = convert_to_docs(_doc.metadata['convert_to_json'],\n  _doc.url);\ndocs;",
      "scriptflags": "d"
    },
    "extraUrls": [
      {
        "url": "http://www.w3schools.com/xml/simple.xml"
      }
    ],
    "updateCycle_secs": 86400
  },
  "tags": [
    "tag1"
  ],
  "title": "aaa xml test",
  "unstructuredAnalysis": {
    "meta": [
      {
        "context": "First",
        "fieldName": "json",
        "script": "var json = eval('(' + text + '); json;",
        "scriptlang": "javascript"
      }
    ]
  },
  "useExtractor": "none",
  "useTextExtractor": "none"
}
```

Output

```
{
  "communityId": ["4d38b72c054548f038a0414a"],
  "created": "Jun 5, 2013 09:12:15 PM UTC",
  "description": "TBD",
  "fullText": "{ \"calories\" : \"650\" , \"description\" : \"two of our famous
Belgian Waffles with plenty of real maple syrup\" , \"price\" : \"$5.95\" , \"name\" :
\"Belgian Waffles\"}",
  "mediaType": ["News"],
  "metadata": {"json": [{
    "calories": "650",
    "description": "two of our famous Belgian Waffles with plenty of real maple
syrup",
    "name": "Belgian Waffles",
    "price": "$5.95"
  }]}},
  "modified": "Jun 5, 2013 09:12:15 PM UTC",
  "publishedDate": "Jun 5, 2013 09:12:15 PM UTC",
  "source": ["aaa xml test"],
  "sourceKey": ["www.w3schools.com.xml.simple.xml"],
  "tags": ["tag1"],
  "title": "TBD",
  "url": "http://www.w3schools.com/xml/simple.xml#0"
}
{
  "communityId": ["4d38b72c054548f038a0414a"],
  "created": "Jun 5, 2013 09:12:15 PM UTC",
  "description": "TBD",
  "fullText": "{ \"calories\" : \"900\" , \"description\" : \"light Belgian waffles
covered with strawberries and whipped cream\" , \"price\" : \"$7.95\" , \"name\" :
\"Strawberry Belgian Waffles\"}",
  "mediaType": ["News"],
  "metadata": {"json": [{
    "calories": "900",
    "description": "light Belgian waffles covered with strawberries and whipped
cream",
    "name": "Strawberry Belgian Waffles",
    "price": "$7.95"
  }]}},
  "modified": "Jun 5, 2013 09:12:15 PM UTC",
  "publishedDate": "Jun 5, 2013 09:12:15 PM UTC",
  "source": ["aaa xml test"],
  "sourceKey": ["www.w3schools.com.xml.simple.xml"],
  "tags": ["tag1"],
  "title": "TBD",
  "url": "http://www.w3schools.com/xml/simple.xml#1"
}
{
  "communityId": ["4d38b72c054548f038a0414a"],
  "created": "Jun 5, 2013 09:12:15 PM UTC",
  "description": "TBD",
  "fullText": "{ \"calories\" : \"900\" , \"description\" : \"light Belgian waffles
covered with an assortment of fresh berries and whipped cream\" , \"price\" :
\"$8.95\" , \"name\" : \"Berry-Berry Belgian Waffles\"}",
  "mediaType": ["News"],
  "metadata": {"json": [{
    "calories": "900",
```

```
        "description": "light Belgian waffles covered with an assortment of fresh
berries and whipped cream",
        "name": "Berry-Berry Belgian Waffles",
        "price": "$8.95"
    }},
    "modified": "Jun 5, 2013 09:12:15 PM UTC",
    "publishedDate": "Jun 5, 2013 09:12:15 PM UTC",
    "source": ["aaa xml test"],
    "sourceKey": ["www.w3schools.com.xml.simple.xml"],
    "tags": ["tag1"],
    "title": "TBD",
    "url": "http://www.w3schools.com/xml/simple.xml#2"
}
{
    "communityId": ["4d38b72c054548f038a0414a"],
    "created": "Jun 5, 2013 09:12:15 PM UTC",
    "description": "TBD",
    "fullText": "{ \"calories\" : \"600\" , \"description\" : \"thick slices made from
our homemade sourdough bread\" , \"price\" : \"$4.50\" , \"name\" : \"French
Toast\"}",
    "mediaType": ["News"],
    "metadata": {"json": [{
        "calories": "600",
        "description": "thick slices made from our homemade sourdough bread",
        "name": "French Toast",
        "price": "$4.50"
    }]}},
    "modified": "Jun 5, 2013 09:12:15 PM UTC",
    "publishedDate": "Jun 5, 2013 09:12:15 PM UTC",
    "source": ["aaa xml test"],
    "sourceKey": ["www.w3schools.com.xml.simple.xml"],
    "tags": ["tag1"],
    "title": "TBD",
    "url": "http://www.w3schools.com/xml/simple.xml#3"
}
{
    "communityId": ["4d38b72c054548f038a0414a"],
    "created": "Jun 5, 2013 09:12:15 PM UTC",
    "description": "TBD",
    "fullText": "{ \"calories\" : \"950\" , \"description\" : \"two eggs, bacon or
sausage, toast, and our ever-popular hash browns\" , \"price\" : \"$6.95\" , \"name\"
: \"Homestyle Breakfast\"}",
    "mediaType": ["News"],
    "metadata": {"json": [{
        "calories": "950",
        "description": "two eggs, bacon or sausage, toast, and our ever-popular hash
browns",
        "name": "Homestyle Breakfast",
        "price": "$6.95"
    }]}},
    "modified": "Jun 5, 2013 09:12:15 PM UTC",
    "publishedDate": "Jun 5, 2013 09:12:15 PM UTC",
    "source": ["aaa xml test"],
    "sourceKey": ["www.w3schools.com.xml.simple.xml"],
    "tags": ["tag1"],
```

```
"title": "TBD",
"url": "http://www.w3schools.com/xml/simple.xml#4"
}
```

WITS source gallery

Sample document

WITS XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<IncidentList xmlns="http://wits.nctc.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wits.nctc.gov WITS.XSD">

  <Incident>
    <ICN>200459257</ICN>
    <Subject>l parliamentary candidate killed by suspected LTTE in Batticaloa, North
    Eastern Province, Sri Lanka</Subject>
    <Summary>On 1 March 2004, during the pre-dawn hours, in Batticaloa, North Eastern
    Province, Sri Lanka, an armed assailant fired upo
    n and killed a Tamil parliamentary candidate as he was recovering from a previous
    assassination attempt on 28 February 2004. No gr
    oup claimed responsibility, although authorities believed the Liberation Tigers of
    Tamil Eelam (LTTE) was responsible.</Summary>
    <IncidentDate>03/01/2004</IncidentDate>
    <ApproximateDate>No</ApproximateDate>
    <MultipleDays>No</MultipleDays>
    <EventTypeList>
    <EventType>Armed Attack</EventType>
    </EventTypeList>
    <Assassination>Yes</Assassination>
    <Suicide>No</Suicide>
    <WeaponTypeList>
    <WeaponType>Firearm</WeaponType>
    </WeaponTypeList>
    <IED>No</IED>
    <Location>
    <Region>South Asia</Region>
    <Country>Sri Lanka</Country>
    <CityStateProvinceList><CityStateProvince>
    <City>Batticaloa</City>
    <StateProvince>North Eastern Province</StateProvince>
    </CityStateProvince>
    </CityStateProvinceList>
    </Location>
    <VictimList>
    <Victim>
    <VictimType>Politically Affiliated</VictimType>
    <Combatant>No</Combatant>
    <Nationality>Sri Lanka</Nationality>
    <DefiningCharacteristicList>
    <DefiningCharacteristic>Unknown</DefiningCharacteristic>
    </DefiningCharacteristicList>
```

```
<TargetedCharacteristicList>
<TargetedCharacteristic>Unknown</TargetedCharacteristic>
</TargetedCharacteristicList>
<Indicator>Targeted</Indicator>
<Child>No</Child>
<DeadCount>1</DeadCount>
<WoundedCount>0</WoundedCount>
<HostageCount>0</HostageCount>
</Victim>
</VictimList>
<FacilityList/>
<PerpetratorList>
<Perpetrator>
<Nationality>Sri Lanka</Nationality>
<Characteristic>Secular/Political/Anarchist</Characteristic>
</Perpetrator>
</PerpetratorList>
</Incident>
```

```
<!--...-->

</IncidentList>
```

Source

```
{
  "description": "wits test",
  "extractType": "File",
  "file": {
    "XmlIgnoreValues": [
      "DefiningCharacteristicList",
      "TargetedCharacteristicList",
      "WeaponTypeList",
      "PerpetratorList",
      "VictimList",
      "EventTypeList",
      "CityStateProvinceList",
      "FacilityList"
    ],
    "XmlPrimaryKey": "icn",
    "XmlRootLevelValues": [
      "Incident"
    ],
    "XmlSourceName":
"https://wits.nctc.gov/FederalDiscoverWITS/index.do?N=0&Ntk=ICN&Ntx=mode%20match&Ntt="
,
    "domain": "XXX",
    "password": "XXX",
    "username": "XXX"
  },
  "isPublic": true,
  "mediaType": "Report",
  "searchCycle_secs": -1,
  "structuredAnalysis": {
    "associations": [
      {
        "creationCriteriaScript": "$FUNC( isOrganizationSpecified(); )",
        "entity1": "$SCRIPT( return
_doc.metadata.perpetrator[0].characteristic + ' from ' +
_doc.metadata.perpetrator[0].nationality; )",
        "entity2": "$FUNC( getOrganizationName(); )",
        "verb": "member of",
        "verb_category": "generic relations"
      },
      {
        "creationCriteriaScript": "$FUNC( isOrganizationSpecified(); )",
        "entity1": "Organization",
        "entity2": "FacilityType",
        "geo_index": "Location",
        "iterateOver": "entity1/entity2/geo_index",
        "time_start": "$SCRIPT( return _doc.metadata.incidentdate[0]; )",
        "verb": "attacked",
        "verb_category": "assault/attack"
      }
    ],
  },
}
```

```

        "creationCriteriaScript": "$FUNC( isOrganizationSpecified(); )",
        "entity1": "Organization",
        "entity2": "VictimType",
        "geo_index": "Location",
        "iterateOver": "entity1/entity2/geo_index",
        "time_start": "$SCRIPT( return _doc.metadata.incidentdate[0]; )",
        "verb": "attacked",
        "verb_category": "assault/attack"
    },
    {
        "creationCriteriaScript": "$FUNC( isOrganizationSpecified(); )",
        "entity1": "Organization",
        "entity2": "HostageType",
        "geo_index": "Location",
        "iterateOver": "entity1/entity2/geo_index",
        "time_start": "$SCRIPT( return _doc.metadata.incidentdate[0]; )",
        "verb": "took hostage",
        "verb_category": "assault/attack"
    },
    {
        "creationCriteriaScript": "$SCRIPT( if (isOrganizationSpecified() ==
false) return true; )",
        "entity1": "PersonPerpetrator",
        "entity2": "FacilityType",
        "geo_index": "Location",
        "iterateOver": "entity1/entity2/geo_index",
        "time_start": "$SCRIPT( return _doc.metadata.incidentdate[0]; )",
        "verb": "attacked",
        "verb_category": "assault/attack"
    },
    {
        "creationCriteriaScript": "$SCRIPT( if (isOrganizationSpecified() ==
false) return true; )",
        "entity1": "PersonPerpetrator",
        "entity2": "VictimType",
        "geo_index": "Location",
        "iterateOver": "entity1/entity2/geo_index",
        "time_start": "$SCRIPT( return _doc.metadata.incidentdate[0]; )",
        "verb": "attacked",
        "verb_category": "assault/attack"
    },
    {
        "creationCriteriaScript": "$SCRIPT( if (isOrganizationSpecified() ==
false) return true; )",
        "entity1": "PersonPerpetrator",
        "entity2": "HostageType",
        "geo_index": "Location",
        "iterateOver": "entity1/entity2/geo_index",
        "time_start": "$SCRIPT( return _doc.metadata.incidentdate[0]; )",
        "verb": "took hostage",
        "verb_category": "assault/attack"
    }
],
"description": "$metadata.summary",
"docGeo": {
    "city": "$SCRIPT( return _doc.metadata.location[0].citystateprovince.city;
)",
    "country": "$SCRIPT( return _doc.metadata.location[0].country; )",
    "stateProvince": "$SCRIPT( return

```



```

_doc.metadata.location[0].citystateprovince.stateprovince; )"
  },
  "entities": [
    {
      "creationCriteriaScript": "$FUNC( isOrganizationSpecified(); )",
      "dimension": "Who",
      "disambiguated_name": "$metadata.organization",
      "type": "Organization",
      "useDocGeo": false
    },
    {
      "dimension": "Where",
      "disambiguated_name": "$FUNC( getLocationEntity(); )",
      "iterateOver": "location",
      "type": "Location",
      "useDocGeo": true
    },
    {
      "entities": [
        {
          "dimension": "Who",
          "disambiguated_name": "$FUNC( getVictim(); )",
          "frequency": "$FUNC( getVictimCount(); )",
          "type": "VictimType",
          "useDocGeo": false
        },
        {
          "dimension": "Who",
          "disambiguated_name": "$FUNC( getVictim(); )",
          "frequency": "$hostagecount",
          "type": "HostageType",
          "useDocGeo": false
        }
      ],
      "iterateOver": "victim",
      "useDocGeo": false
    },
    {
      "dimension": "Who",
      "disambiguated_name": "$characteristic from $nationality",
      "iterateOver": "perpetrator",
      "type": "PersonPerpetrator",
      "useDocGeo": false
    },
    {
      "dimension": "What",
      "disambiguated_name": "",
      "frequency": "1",
      "iterateOver": "weapontype",
      "type": "WeaponType",
      "useDocGeo": false
    },
    {
      "dimension": "What",
      "disambiguated_name": "$FUNC( getEventType(); )",
      "frequency": "1",
      "iterateOver": "eventtype",
      "type": "EventType",
      "useDocGeo": false
    }
  ]
}

```

```

    },
    {
        "dimension": "What",
        "disambiguated_name": "$SCRIPT( var s = (_iterator.indicator ==
'Targeted') ? 'Targeted,' : '' ; s+= _iterator.facilitytype; return s; )",
        "frequency": "$quantity",
        "iterateOver": "facility",
        "type": "FacilityType",
        "useDocGeo": false
    }
],
    "publishedDate": "$metadata.incidentdate",
    "script": "function getLocationEntity() { var s =
(_iterator.citystateprovince.city != null) ? _iterator.citystateprovince.city : '';
s+= (s.length > 0) ? ',' : ''; s+= (_iterator.citystateprovince.stateprovince != null)
? _iterator.citystateprovince.stateprovince : ''; s+= (s.length > 0) ? ',' : ''; s+=
(_iterator.country != null) ? _iterator.country : ''; return s; } function getVictim()
{ var indicator = (_iterator.indicator != 'Unknown') ? _iterator.indicator : ''; var
victimType = (_iterator.victimtype != 'Unknown') ? _iterator.victimtype : ''; var
child = (_iterator.child == 'Yes') ? 'Child' : 'Adult'; var combatant =
(_iterator.combatant == 'Yes') ? 'Combatant' : ''; var targeted =
(_iterator.targetedcharacteristic != 'None' && _iterator.targetedcharacteristic !=
'Unknown') ? _iterator.targetedcharacteristic : ''; var defining =
(_iterator.definingcharacteristic != 'None' && _iterator.definingcharacteristic !=
'Unknown') ? _iterator.definingcharacteristic : ''; var s = indicator; if
(victimType.length > 0) { if (s.length > 0) { s += ', ' ; } s += victimType; } if
(s.length > 0) { s += ', ' ; } s += child; if (combatant.length > 0) { if (s.length >
0) { s += ', ' ; } s += combatant; } if (targeted.length > 0) { if (s.length > 0) { s
+= ', ' ; } s += targeted; } if (defining.length > 0) { if (s.length > 0) { s += ', ' ;
} s += defining; } if (s.length > 0) { s += ' from ' ; } s += _iterator.nationality;
return s; } function getVictimCount() { var count = parseInt(_iterator.deadcount, 10)
+ parseInt(_iterator.woundedcount, 10); return count; } function getEventType() { var
s = _value; if (_doc.metadata.assassination[0] == 'Yes') s += ', Assassination'; if
(_doc.metadata.suicide[0] == 'Yes') s += ', Suicide'; if (_doc.metadata.ied[0] ==
'Yes') s += ', IED'; return s; } function getEventTypeFull() { var s =
_doc.metadata.eventtype[0]; if (_doc.metadata.assassination[0] == 'Yes') s += ',
Assassination'; if (_doc.metadata.suicide[0] == 'Yes') s += ', Suicide'; if
(_doc.metadata.ied[0] == 'Yes') s += ', IED'; return s; } function
isOrganizationSpecified() { if (_doc.metadata.organization != null &&
_doc.metadata.organization[0].toString().toLowerCase() == 'no group') { return false;
} else { return true; } }function getOrganizationName() { if
(_doc.metadata.organization != null &&
_doc.metadata.organization[0].toString().toLowerCase() != 'no group') { return
_doc.metadata.organization[0]; } }",
        "scriptEngine": "JavaScript",
        "title": "$metadata.subject"
    },
    "tags": [
        "incidents",
        "nctc",
        "terrorism",
        "wits",
        "events",
        "worldwide"
    ],
    "title": "wits test",
    "unstructuredAnalysis": {
        "meta": [
            {

```

```
    "context": "All",
    "fieldName": "organization",
    "groupNum": 1,
    "script": "believed the (.*)?(?: \\([^)]*\\))? (was|were)
responsible",
    "scriptlang": "regex"
  },
  {
    "context": "All",
    "fieldName": "organization",
    "groupNum": 1,
    "script": "believed (.*)?(?: \\([^)]*\\))? (was|were) responsible",
    "scriptlang": "regex"
  },
  {
    "context": "All",
    "fieldName": "organization",
    "groupNum": 1,
    "script": ". ([^.]*)?(?: \\([^)]*\\))? claimed responsibility\\.\\.$",
    "scriptlang": "regex"
  }
]
},
```

```
"url": "smb://modus:139/wits/allfiles/",
"useExtractor": "none"
}
```

Sample output

```
{
  "associations": [],
  "communityId": ["506dc16dfbf042893dd6b8f2"],
  "created": "May 16, 2013 12:39:20 PM UTC",
  "description": "On 1 July 2009, in the morning, in Manugay, Konar, Afghanistan,
assailants fired a rocket at a residence, killing two children, one civilian, wounding
four civilians, and damaging the residence. No group claimed responsibility.",
  "entities": [
    {
      "actual_name": "Manugay,Kunar,Afghanistan",
      "dimension": "Where",
      "disambiguated_name": "Manugay,Kunar,Afghanistan",
      "doccount": 0,
      "frequency": 1,
      "index": "manugay,kunar,afghanistan/location",
      "relevance": 0,
      "totalfrequency": -1,
      "type": "Location"
    },
    {
      "actual_name": "Targeted, Civilian, Adult from Afghanistan",
      "dimension": "Who",
      "disambiguated_name": "Targeted, Civilian, Adult from Afghanistan",
      "doccount": 0,
      "frequency": 5,
      "index": "targeted, civilian, adult from afghanistan/victimtype",
      "relevance": 0,
      "totalfrequency": -1,
      "type": "VictimType"
    },
    {
      "actual_name": "Targeted, Civilian, Child from Afghanistan",
      "dimension": "Who",
      "disambiguated_name": "Targeted, Civilian, Child from Afghanistan",
      "doccount": 0,
      "frequency": 2,
      "index": "targeted, civilian, child from afghanistan/victimtype",
      "relevance": 0,
      "totalfrequency": -1,
      "type": "VictimType"
    },
    {
      "actual_name": "Islamic Extremist (Sunni) from Unknown",
      "dimension": "Who",
      "disambiguated_name": "Islamic Extremist (Sunni) from Unknown",
      "doccount": 0,
      "frequency": 1,
      "index": "islamic extremist (sunni) from unknown/personperpetrator",
      "relevance": 0,
      "totalfrequency": -1,
    }
  ]
}
```

```

        "type": "PersonPerpetrator"
    },
    {
        "actual_name": "Missile/Rocket",
        "dimension": "What",
        "disambiguated_name": "Missile/Rocket",
        "doccount": 0,
        "frequency": 1,
        "index": "missile/rocket/weapontype",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "WeaponType"
    },
    {
        "actual_name": "Armed Attack",
        "dimension": "What",
        "disambiguated_name": "Armed Attack",
        "doccount": 0,
        "frequency": 1,
        "index": "armed attack/eventtype",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "EventType"
    },
    {
        "actual_name": "Targeted,Residence",
        "dimension": "What",
        "disambiguated_name": "Targeted,Residence",
        "doccount": 0,
        "frequency": 1,
        "index": "targeted,residence/facilitytype",
        "relevance": 0,
        "totalfrequency": -1,
        "type": "FacilityType"
    }
],
"mediaType": ["Report"],
"metadata": {
    "approximatedate": ["No"],
    "assassination": ["No"],
    "eventtype": ["Armed Attack"],
    "facility": [{
        "combatant": "No",
        "damage": "Light",
        "definingcharacteristic": "Unknown",
        "facilitytype": "Residence",
        "indicator": "Targeted",
        "nationality": "Afghanistan",
        "quantity": "1",
        "targetedcharacteristic": "Unknown"
    }
    ],
    "icn": ["200906281"],
    "ied": ["No"],
    "incidentdate": ["07/01/2009"],
    "location": [{
        "citystateprovince": {
            "city": "Manugay",
            "stateprovince": "Kunar"
        }
    }
    ],

```

```
        "country": "Afghanistan",
        "region": "South Asia"
    }],
    "multipliedays": ["No"],
    "organization": ["No group"],
    "perpetrator": [{
        "characteristic": "Islamic Extremist (Sunni)",
        "nationality": "Unknown"
    }],
    "subject": ["2 children, 1 civilian killed, 4 civilians wounded in rocket
attack in Manugay, Konar, Afghanistan"],
    "suicide": ["No"],
    "summary": ["On 1 July 2009, in the morning, in Manugay, Konar, Afghanistan,
assailants fired a rocket at a residence, killing two children, one civilian, wounding
four civilians, and damaging the residence. No group claimed responsibility."],
    "victim": [
        {
            "child": "No",
            "combatant": "No",
            "deadcount": "1",
            "definingcharacteristic": "Unknown",
            "hostagecount": "0",
            "indicator": "Targeted",
            "nationality": "Afghanistan",
            "targetedcharacteristic": "Unknown",
            "victimtype": "Civilian",
            "woundedcount": "4"
        },
        {
            "child": "Yes",
            "combatant": "No",
            "deadcount": "2",
            "definingcharacteristic": "Unknown",
            "hostagecount": "0",
            "indicator": "Targeted",
            "nationality": "Afghanistan",
            "targetedcharacteristic": "Unknown",
            "victimtype": "Civilian",
            "woundedcount": "0"
        }
    ],
    "weapontype": ["Missile/Rocket"]
},
"modified": "Jan 15, 2013 08:26:55 PM UTC",
"publishedDate": "Jul 1, 2009 12:00:00 AM UTC",
"source": ["wits test"],
"sourceKey": ["modus.139.wits.test."],
"sourceUrl": "smb://modus:139/wits/test/WITS_2009_07.xml",
"tags": [
    "incidents",
    "nctc",
    "terrorism",
    "wits",
    "events",
    "worldwide"
],
"title": "2 children, 1 civilian killed, 4 civilians wounded in rocket attack in
Manugay, Konar, Afghanistan",
"url":
```

```
"https://wits.nctc.gov/FederalDiscoverWITS/index.do?N=0&Ntk=ICN&Ntx=mode%20match&Ntt=200906281"
}
```

Specifying a Data Source

The following document describe the basics of specifying how to extract data from different types of source systems (RSS feeds, databases, files, etc.).

- [Using the Feed Harvester](#)
- [Using the Database Harvester](#)
- [Using the File Harvester](#)

Using the Database Harvester

There is a separate reference page for the Database Harvester configuration object.

Infini.e supports harvesting data from traditional RDBMS (Relational Database Management Systems) using JDBC (Java Database Connectivity) drivers. The Sample Database Harvester Specification below demonstrates how to connect to and extract data from a database using the harvester:

Sample Database Harvester Specification

```
source : {
  ...
  "extractType" : "Database",
  "authentication" : {
    "username" : "username",
    "password" : "password"},
  "database" : {
    "databaseType" : "mysql",
    "hostname" : "my.databaseserver.com",
    "port" : "3306"
    "databaseName" : "database",
    "query" : "SELECT * FROM IncidentReport",
    "deltaQuery" : "SELECT * FROM IncidentReport WHERE REPORTDATETIME >= (SELECT
ADDDATE(CURDATE(),-7))",
    "deleteQuery" : "",
    "primaryKey" : "NID",
    "title" : "CCN",
    "snippet" : "OFFENSE",
    "publishedDate" : "REPORTDATETIME"
  },
  "useExtractor" : "none",
  ...
}
```

- **extractType**
The extractType field is used to tell the harvester the type of source to extract from, i.e.: **Database**. Other valid values include: File, Feed, etc.
- **authentication**
The Authentication object of the Source document is a subset of the full Authentication object in that it only uses the 'username' and 'password' fields. The Database Harvester uses the username and password from the Authentication object as database credentials (if needed).
 - **username**
 - **password**
Note: The password field in the Authentication object is currently clear text. If the string value placed in password is clear text it is not encrypted by Infini.e. Encryption of the password field is planned for a future release.
- **database**
The Database object is used to specify of how to access the data to be extracted and how to extract the individual fields within the source file data records.

- **databaseType**
The type of RDBMS to connect to. Valid values currently include: mysql, db2, oracle, mssqlserver, sybase.
- **hostname**
The hostname of the database server to connect to, i.e. "my.databaseserver.com" in the example above.
- **port**
The port that the database accepts incoming connections on.
- **databaseName**
The name of the database to connect to.
- **query**
The query field is used to specify the SQL used to perform a full extraction of data for the source. This is generally used the first time the harvester extracts data from a source with incremental extractions being specified using the deltaQuery below.
- **deltaQuery**
The deltaQuery field is used to specify the SQL that extracts data from the source RDBS based on one or more delta values, i.e. created or modified date for a record.
- **deleteQuery**
Note: The deleteQuery functionality of the Database Harvester is not implemented in the Beta version of Infini.e.
- **primaryKey**
Primary key field in data set, used to help identify whether a record is new or previously harvested.
- **title**
Record field used to populate the document's title field.
- **snippet**
Record field used to populate the document's description field.
- **publishedDate**
Record field used to populate the document's published date field.
- **useExtractor**
Additional extractor to use (i.e. other than, or in addition to, the Structured Analysis Harvester) to use to extract entity and event data.

Note: A complete example of the above source including a sample database document harvested from the source can be found here: [Sample Database Source](#).

Using the Feed Harvester

There is a separate reference for the [Feed Harvester configuration object](#).

Infini.e supports harvesting data from RSS feeds in a number of common formats (Atom, RSS 1.0, RSS 2.0, etc.). The Feed Harvester also allows for collection of specified URLs, and link scraping.

The Sample Feed Harvester Specification below demonstrates how to connect to and extract data from a feed using the harvester:

Sample Feed Harvester Specification

```
source : {
  ...
  "extractType" : "Feed",
  "authentication" : {
    "username" : "username",
    "password" : "password"},
  "url" : "http://www.mayoclinic.com/rss/blog.xml",
  "rss": {
    "waitTimeOverride_ms": 10000, // (a standard "politeness" sleep for consecutive
    accesses to the same web-site, system default is 10s)
    ...
    // "Advanced" control functionality
    "updateCycle_secs": 86400, // If specified (eg value shown is 1 day) then will
    re-extract the URL document with that periodicity
    "regexInclude": ".*" // (Optional) regular expression, anything not matching if
    discarded
    "regexExclude": ".*\\.pdf", // (Optional) eg this example will discard PDFs
    // "Advanced" extraction functionality
    "userAgent": "for emulating a specific browser, defaults to FireFox",
    "extraUrls": {...}, // See the reference - for collecting specified URLs
    "searchConfig": { ... } // See the reference and the description below - for
    link scraping
  }
  ...
}
```

Note: A complete example of the above source including a sample feed document harvested from the source can be found here: [Feed Source](#).

- **extractType**

The extractType field is used to tell the harvester the type of source to extract from, i.e.: **Feed**. Other valid values include: Database, Feed, etc.

- **authentication (optional)**

The Authentication object of the Source document is a subset of the full Authentication object in that it only uses the 'username' and 'password' fields. The Feed Harvester uses the username and password from the Authentication object as feed credentials (if needed).

- **username**
- **password**

Note: The password field in the Authentication object is currently clear text. If the string value placed in password is clear text it is not encrypted by Infnit.e. Encryption of the password field is planned for a future release.

- **url**

The URL to retrieve the RSS feed from.

- **extraUrls**

Allows collection of specified URLs

- **searchConfig**

Described below

API parsing, Link following, web crawling, and similar activities

It will often be the case that a base URL will not contain useful content, but will contain links to useful content. Examples include:

- Internet and Intranet searches, lists of PDFs of academic publications, etc.
- Many site APIs (eg LinkedIn, TechCrunch, Flickr)
- Directory or disambiguation pages (eg Wiki)

In addition, it will often be the case that pages with useful content also contain links to more pages with useful content, ie a standard web crawling issue.

In addition, it will often be the case that both XML and JSON APIs will return a single URL containing many "documents" ie independent content snippets.

In some cases pagination from APIs is achieved by passing a field containing the URL of the next page; in other cases the URL follows a standard pattern.

The "searchConfig" field of the Feed Harvester provides a nice interface for all aspects of handling JSON APIs (splitting replies into many documents, fetching links, and following "next page" links). It can also be used for more standard HTML link-following and web-crawling, though is harder to use in that context.

The remainder of this section describes the basic usage of the "searchConfig" object (ie suitable for API parsing), and then provides a brief description of the more complex activity of HTML parsing.

Basic use of searchConfig

The "script" field of the "searchConfig" object needs to contain javascript (so the "scriptlang" field should always be set to "javascript"). The javascript is passed a single variable "text", containing the response to the specified URL (or URLs if "extraUrls" is specified), and needs to return an array of the following objects:

```
{
  "url": string, // Mandatory - this URL is copied into the "URL" field of the
generated document,
  // and is used to fetch the content unless "fullText" is set.
  "title": string, // Mandatory (unless "spiderOut" set, see below) - this is used to
generate the document's title.
  "description": string, // Optional, if set then used to generate the document's
description.
  "publishedDate": string, // Optional, if not set then the current date is used
instead.

  "fullText": string, // Optional, if set then this is the content for the generated
document, ie "url" is not followed.


  "spiderOut": integer //Optional, if set to true then the searchConfig.script is
applied to the resulting document,
  // for a depth of up to "searchConfig.maxDepth" times
  // Note spiderOut only works if rss.extraUrls is non-empty (ie use that instead
of url)
}
```

So the basic use of searchConfig on JSON-based APIs should be quite straightforward, ie along the following lines:

Outline API parsing using searchConfig

```
var json = eval('(' + text + ')');
var retval = [];
// For each "result" in the array
// Extract URL, title, description, eg for the flickr blogs API
// (http://www.flickr.com/services/api/response.json.html)
for (x in json.blogs.blog) {
  var blog = json.blogs.blog[x];
  var retobj = { url: blog.url, title: blog.name };
  retval.push(retobj);
}
// Alternatively set retobj.fullText to specify the content from the API response
// In addition set retobj.spiderOut: true, to run this script on the corresponding
URL, eg:
if (null != json.nextPageUrl)
  retval.push({url: json.nextPageUrl, spiderOut: true});
retval; // annoying feature of our javascript engine, instead of returning you just
evaluate the var to return
```

For XML APIs the basic principle is the same, but the XML object needs to be parsed using embedded Java calls (since the Rhino javascript engine currently in use does not support e4x - it is on our roadmap to upgrade to a version that does).

 When Javascript is used, the same security restrictions as elsewhere apply.

Advanced use of searchConfig There are 2 main differences in using "searchConfig" to parse HTML:

- The HTML needs to be parsed - this is discussed below ("using xpath to parse HTML")
- It will often be the case (eg for Intranet search engines) that multiple pages must be traversed (eg 10 results/page). The following sub-fields of "searchConfig" are intended to handle these cases:
 - **numPages**: the total number of pages that will be checked each search cycle.
 - **pageChangeRegex**: a regex that must have at least one capturing group and must match the *entire* part of the URL that controls the page number. See example below.
 - **pageChangeReplace**: the above string that controls the page number, with \$1 used to represent the page number.
 - (slightly misnamed) **numResultsPerPage**: If the "page number" in the URL is actually a result offset *and not a page offset*, then this field should be the number of results per page (which is then multiplied by the page number to generate the "\$1" string mentioned above). See example.

For example, consider a URL of the form:

- **http://www.blahblah.com/search?q=search_terms&page=1**


Then the following parameters would be used: "pageChangeRegex": "(page=\\d+)", "pageChangeReplace": "page=\$1", "numResultsPerPage": 1

And for a URL of the form:

- **http://www.blahblahblah.com/search?q=search_terms&pagesize=20&start_result=0**

The the following parameters would be used: "pageChangeRegex": "(start_result=\\d+)", "pageChangeReplace": "start_result=\$1", "numResultsPerPage": 20

Finally, it is more likely that standard web-crawling measures are needed such as custom user-agents, and per-page wait times. Because these might well be different from the search engine to the pages themselves, "searchConfig" has its own "waitTimeBetweenPages_ms", "userAgent" fields (if not specified these are inherited from the parent "rss" object).

 Note that "fullText" can be set to a JSON object, and it is then converted into a string containing the JSON (ie ready to be converted back into JSON with eval) in the derived document. This is handy because Rhino does not support "JSON.stringify".

Using Xpath to parse HTML and XML

The "searchConfig" object has a field "extraMeta" that enables other script types to be used. The main use case for this is using the "xpath" scripting language (with "groupNum": -1 to generate objects) to extract the URLs required, and then use the existing "script" field (with "scriptflags": "m") to tidy up those objects into the required "url"/"title"/"description"/"publishedData" format.

The "extraMeta" array works identically to the "meta" array in the [unstructured analysis harvester](#), except that the metadata is not appended to any documents, ie after it has been passed to "script" to generate URL links it is discarded.

The "rss.searchConfig.script" javascript (ie the last element in the processing chain) can access the fields created from "extraMeta" (ie extraMeta[*].fieldName) from the "_metadata" variable that is automatically passed in if no flags are specified (otherwise make sure the "m" flag is specified - or equivalently "d" to use "_doc,metadata").

The "extraMeta" field can also be used for 2 debugging/error handling cases:

- If a field called "_ONERROR_" is generated then if no links are returned from the first page (ie likely due to a formatting error) then the contents of _ONERROR_ (assumed to be a string) are dumped to the harvest message.
- Only when running from the "[Config - Source - Test](#)" API call (including from the [Source Editor GUI](#)), then for every page, all of the _ONDEBUG_ field values (can be string or object) are dumped to the harvest message.

Example uses of *ONERROR* and *ONDEBUG*

```
"rss": {
  "searchConfig": {
    "extraMeta": [
      {
        "context": "First",
        "fieldName": "_ONERROR_",
        "scriptlang": "javascript",
        "script": "var page = text; page;"
      },
      {
        "context": "First",
        "fieldName": "title", // (eg)
        "scriptlang": "javascript",
        //... (can return string or object)
      },
      {
        "context": "First",
        "fieldName": "_ONDEBUG_",
        "scriptlang": "javascript",
        "flags": "m",
        "script": "var ret = _metadata.title; ret;"
      },
      //...
    ]
  }
}
```

Using the File Harvester

Infini.t.e supports harvesting files from Windows/Samba shares or a harvester's local filesystem.

Infini.t.e supports harvesting data from a variety of file formats including unstructured text files, semi-structured text files, CSV files, and XML files.

[There is a separate reference for the File Harvester configuration object.](#)

There are a number of typical strategies for dealing with standard file formats:

- "Office" documents (PDF, Word, Powerpoint): converted to text by Tika, can then be entity-extracted as normal.
- Text-based documents (eg emails): can be entity-extracted as normal.
 - The text can also be cleansed (eg of header/footer information) using the [Unstructured Analysis Harvester](#), which can also extract "structured" information such as author, email distribution, subject, send date, etc etc.
- CSV files: can be turned into metadata using the [Unstructured Analysis Harvester](#) or using the automated parser configured as described below. [See the source gallery for examples of the different permutations.](#)
- XML and JSON (see below): is automatically turned into [metadata](#), which can then be converted to [entities](#) and [associations](#) using the [Structured Analysis Harvester](#).
- [Infini.t.e shares](#): Uploaded "binary files" are treated as office/text documents as above (except ZIP files see later), JSON shares are processed as JSON. Uploaded ZIP files are automatically decompressed on harvest and treated as a directory of other files which are handled as described above.
- The results of [Infini.t.e plugins](#): The results of map/reduce jobs (or less commonly saved queries), eg as could be obtained from [Custom - Get Results](#) can be treated as a directory of JSON files.
 - **NOTE:** *this last input type (custom) has one important limitation: if/when the custom job is re-run all the "_ids" change, which means that the old documents are retained rather than being overwritten by new documents. Therefore the documents for the source should be manually deleted from the API or GUI if (as will normally be the case) this is not desired. There is a roadmap item to address this better in the near future.*

Whenever the [Unstructured Analysis Harvester](#) is used to generate [metadata](#), the [Structured Analysis Harvester](#) is then used to turn the [metadata](#) into [entities](#) and [associations](#).

Harvesting XML Files

Sample XML File Harvester Specification

```
source : {
  "url": "string", // see below
  ...
  "file" : {
    "username" : "username",
    "password" : "password",
    "domain" : "WORKGROUP",
    "type": "xml",

    "pathInclude": "^.*[.]xml$",
    "pathExclude": "^.*schema[.]xml$",
    "XmlRootLevelValues" : ["Incident"],
    "XmlIgnoreValues" : [
      "DefiningCharacteristicList",
      "TargetedCharacteristicList",
      "WeaponTypeList",
      "PerpetratorList",
      "VictimList",
      "EventTypeList",
      "CityStateProvinceList",
      "FacilityList"
    ],
    "XmlSourceName" :
"https://wits.nctc.gov/FederalDiscoverWITS/index.do?N=0&Ntk=ICN&Ntx=mode%20match&Ntt="
,
    "XmlPrimaryKey" : "icn"
  },
  "useExtractor" : "none",
  ...
}
```

• url

- The URL needs to be in the following format "**file://<path including leading '/'>**", in which case the filesystem local to the harvester will be used, or "**smb://server:port/path**", in which case the harvester will attempt to connect to the specified Windows/Samba share, or "**s3://**" for S3 parsing (see below for S3 install details), or "**inf://share/<shareid>/<ignored>**" to process Infinite shares, or "**inf://custom/<customid-or-jobtitle>/<ignored>**" to process results of Infinite custom jobs.
 - *Note the leading "/" is required, eg if the path was "/mnt/test_data", then the URL would be "file:///mnt/test_data", ie 3 slashes.*
 - **Note also that "file://" sources can only be run by administrators.**
 - Note also that the local filesystem version is mostly intended for testing, debugging, and "micro installations". The "tomcat" user must have read access to the directories and files on the path.
 - **Check that uploaded files are readable by tomcat ("file:") or the username account ("smb://")**
 - **Note finally that if any of username/password/domain are specified, the URL will be assumed to point to a Windows/Samba share or S3 (see below).**
- S3 is also supported (URL is in the format "**s3://bucket_name>/**" or "**s3://<bucket_name>/path/**").
 - The files in the S3 bucket should be readable by the account specified by the access key.
 - **Note for Admins: S3 is not supported by default, the AWS SDK JAR must be copied into the classpath as described [here](#).**
- Infinite share harvesting is described above, and is indicated with the format "inf://share/<shareid>/" (share id can be obtained in the URL field of the [file uploader](#))
 - After the "<shareid>/" portion of the URL, any arbitrary text can be added to make the role of the share clearer. This text is ignored by the file harvester.
 - The source must share at least one community with the share in order to be processed.
- Infinite custom harvesting is described above, and is indicated with the format "inf://custom/<customid-or-jobtitle>" (custom id and title can be obtained in the URL field of the [plugin manager](#))
 - After the "<customid-or-jobtitle>/" portion of the URL, any arbitrary text can be added to make the role of the share clearer. This text is ignored by the file harvester.
 - The source must share at least one community with the custom plugin in order to be processed.
- Regular expressions can be optionally specified to include (**pathInclude**) only specified files, and/or exclude specified files and

directories (**pathExclude**).

- **extractType**

The extractType field is used to tell the harvester the type of source to extract from, i.e.: **File**. Other valid values include: Database, Feed, etc.

- **file**

The File object is used to specify the specifics of how to access the data to be extracted and how to extract the individual fields within the source file.

- **username**
- **password**

Note: The password field in the Authentication object is currently clear text. If the string value placed in password is clear text it is not encrypted by Infinit.e. Encryption of the password field is planned for a future release. For S3, the Access ID should be entered into the "username", and the Secret Key into the "password" (**note - it is recommended for security that you create a separate AWS user with no permissions other than S3 read/list on the directories**)

- **domain**

The port that the database accepts incoming connections on. (Can be left blank for S3).

- **type**

One of "json", "xml", "tika", "**sv", or null to auto decide

- **XmlRootLevelValues**

The root level field of the XML file at which parsing should begin. (Also works for JSON)

- For "**sv" files, this results in CSV parsing occurring automatically, and the records are mapped into a metadata object called "csv", with the fieldnames corresponding to the values of this array (eg the 3rd value is named after XmlRootLevelValues[2] etc)

- **XmlIgnoreValues**

XML nodes to ignore when parsing the document. (Ignored for JSON)

- For "**sv" files the start of each line is compared to each of the strings in this array - if they match the line is ignored. This allows header lines to be ignored.

- **XmlSourceName**

If specified, the document URL is build as "XmlSourceName" + xml("XmlPrimaryKey"). Also works for JSON, and "**sv" (when XmlRootLevelValues is set)

- **XmlPrimaryKey**

Primary key field in data set, used to help identify whether a record is new or previously harvested. Also works for JSON (dot notation is supported), and "**sv" (when XmlRootLevelValues is set)

- **XmlAttributePrefix**

For XML only, this string is pre-pended to XML attributes before they become JSON fields.

- For "**sv" files when XmlRootLevelValues is set controls the separators as follows: the first char in the string is the separator, the (optional) second char in the string is the quote, and the (optional) third char in the string is the escape character (eg the default is "\",'"")

- **pathInclude, pathExclude:** see above under "url"

- **renameAfterParse:** If a file is fully harvested then it can be moved or deleted. To delete just set this field to "". To rename, this field should be set to the full path to rename the file, using the substitution variables "\$path" (path of directory in which file is currently located) and "\$file" (just the filename portion), eg "\$path/done/\$file", "\$path/\$file.PROCESSED" etc - it is best used in conjunction with pathExclude, eg pathExclude: ".*DELETED" or ".*done/*".

- **useExtractor**

Additional extractor to use (i.e. other than, or in addition to, the Structured Analysis Harvester) to use to extract entity and association data. Note that the "useTextExtractor" field is not used for files.



For XML and JSON file (or "**csv" files where XmlRootLevelValues is set), Where the document(s) within the file references a unique network resource that is of the format "CONSTANT_URL_PATH + VARIABLE_ID" (eg "http://www.website.com?pageNum=3454354"), and the "VARIABLE_ID" component is one of the fields in the XML/JSON object, then "XmlSourceName" and "XmlPrimaryKey" can be used to specify the two components. Note that for JSON the dot notation can be used in "XmlPrimaryKey" for nested fields.

If it is not possible to specify the URL in this manner (but there is a single - not necessarily unique - URI that is related to the document - eg either a network resource or a file in a sub-directory of the fileshare), it is recommended to use the structured analysis handler to set the "displayUrl" parameter.

Structured Analysis - Overview

There is also a [reference page for the Structured Analysis configuration object](#).

The Infinit.e Structured Analysis Harvester is designed to take data ingested from structured data sources (database tables, XML documents, etc.) and enrich the data via the assignment of geospatial information, entities and events. The Structured Analysis Harvester is also capable of transforming source data via basic string concatenation (using simple regular expression support) and more complex transformations using JavaScript. The example Source.structuredAnalysis object below demonstrates the basic features of specifying how to enrich harvested structured data.

Source.structuredAnalysis object

```
source : {
  ...
  structuredAnalysis : {
    docGeo : {"lat": "$metadata.latitude", "lon": "$metadata.longitude"},
    description : "$metadata.reportdatetime: $metadata.offense, $metadata.method
was reported at: $metadata.blocksiteaddress",
    //other document level fields, see reference
    entities : [
      {disambiguous_name: "$metadata.offense, $metadata.method",
dimension: "What",
      type: "CriminalActivity"},

{disambiguous_name: "$metadata.blocksiteaddress, $metadata.city, $metadata.state",
      dimension: "Where", type: "Place", geotag:
{latitude: "$metadata.latitude",
      longitude: "$metadata.longitude"} }],
    "associations" : [

{entity1: "$metadata.offense, $metadata.method", verb: "reported", verb_category: "crime",
      time_start: "$metadata.reportdatetime", "geo_index" : "Location",
      geotag: {lat: "$metadata.latitude", lon: "$metadata.longitude"} } ]

    }
  }
  ...
}
```

Display URL

"displayUrl" sets the corresponding [document JSON](#) field. It is guaranteed not to be used by the Infinit.e platform. It is therefore useful for linking documents to external content. For reference, the way that it is used in the [Infinit.e GUI](#) is as follows:

- If it starts with "http://" then it is treated as a web link
- Otherwise, it is assumed to be a relative file path to the fileshare specified in the [source url](#) field. (eg you can use the "[Document - File - Get](#)" call with the "sourceKey" concatenated to the "displayUrl" to retrieve the file directly from the fileshare).

Using the \$ Operator to Extract Document Data

When structured data is extracted from a source (via the File, Database, or other harvester), each field extracted is captured in the Feed.metadata object. Within the Structured Analysis Harvester data stored in the Metadata object can be access using the \$ operator to signify that we are attempting to retrieve data from a field in our document. For example, in the document above you can extract the Offense field using the following syntax:

```
$metadata.offense or ${metadata.offense}
```

Other fields at the document top level ("title", "\$description", etc) can also be referenced this way

Note: When data is extracted and added to the Metadata object all field name are converted to lowercase.

Note: If the metadata field is an array, the above syntax grabs the first element only. To go deeper into arrays, [javascript must be used](#).

Note: When [iterating over entities](#) or metadata (for either [entity](#) or [association](#) building), the "\$" sign is relative to the iterator, not the document (eg the metadata object being looped over). However when iterating over metadata fields that are strings, then the above document-level referencing is still valid, or "\$value"/"\${value}" can be used to reference the value itself.

Document updates and metadata

Existing documents can be updated in a number of different cases:

- [Files](#) can be updated (changing their "modified time")
- For [RSS feeds/URLs](#), the source parameter "updateCycle_secs" will periodically update the file.
- [Database sources](#) can be updated as the result of a SQL call.

When a document is updated it is essentially equivalent to deleting and the re-creating it, [except that its "_id" field is preserved](#)). The Structured

Analysis Harvester provides a mechanism to do the following useful activities:

- Preserve metadata from the old document (eg so the entities/associations can be recreated)
- Generate new metadata (and thence entities/associations) based on the differences between successive documents.

A script can be placed into ("onUpdateScript" - note the "\$SCRIPT" convention used in [entity/association scriptlets](#) is **not** required here). This script has access to the following Javascript objects:

- "_old_doc": The document object that is about to be deleted
- "_doc": The newly created document object *after all metadata/entity/association creation*.

The last evaluated expression in the script (eg you don't "return val;" you just end the script "val;"), which can be a string, an object, or an array of objects is placed in a metadata field called "_PERSISTENT_". For example the following code just saves the entirety of the old document's metadata:

```
// SOURCE CONFIG:
"structredAnalysis": {
  "scriptEngine": "javascript",
  "onUpdateScript": "var retVal = _old_doc.metadata; retVal;"
}
// RESULT (IN THE CASE OF A DOCUMENT THAT DOESN'T CHANGE):
{
  // Usual document fields
  "metadata": {
    "test1": "test",
    "test2": { "field": "value" },
    "_PERSISTENT_": [{
      "test1": "test",
      "test2": { "field": "value" },
    }]
  }
}
```

And the following script shows a very simple example of comparing the old and new documents:

```
"structredAnalysis": {
  "scriptEngine": "javascript",
  "onUpdateScript": "var delta = _old_doc.metadata.length - _doc.metadata.length; var
retVal = { 'delta': delta }; retVal;"
}
```

Further Reading

- [Specifying Document Level Geographical Location](#)
- [Specifying Entities](#)
- [Specifying Associations](#)
- [Transforming data with JavaScript](#)

Specifying Associations

What is An Association?

[Associations](#) can be "something that happens or is regarded as happening; an occurrence, especially one of some importance", "the outcome, issue, or result of anything", or "something that occurs in a certain place during a particular interval of time" (Definitions found here: <http://dictionary.reference.com/browse/event>). Within Infnit.e events are typically a combination of entities assembled in the form of Noun - Verb - Noun, e.g. "a car crashed into a building", "the plane flew to San Diego". In addition to the Noun - Verb - Noun form events can include geographic information (i.e., where an event happened) as well as a start and/or end time for an event. The following section describes how to specify the extraction of events from a data source using the Structured Analysis Harvester.



Note that at least one of the entity1/entity2/geo_index fields must point to an [entity](#) in the document (either extracted using [Natural Language Processing](#) or using the "entities" block of the [Structured Analysis object](#)). The different ways in which this can be achieved are described below.

Basic Association Specification

The following code demonstrates how to specify a basic association (**Note:** The sample event specification and sample event output below is extracted from a sample [MySQL Database Source](#)):

Basic Event Specification Example

```
{
  //...
  "associations" : [
    {
      "entity1" : "$metadata.offense,$metadata.method",
      "verb" : "reported",
      "verb_category" : "crime",
      "time_start" : "$metadata.reportdatetime",
      "geo_index" : "Location",
      "geotag" : {
        "lat" : "$metadata.latitude",
        "lon" : "$metadata.longitude"}
    },
  ],
  //...
}
```

In the basic example above the following fields have been specified:

- **entity1**
A free form text field containing information about the event "subject", i.e. an entity's disambiguous name.
- **entity1_index**
If present this is the "index" field of the entity matching the entity1 disambiguous name above.
- **verb**
A free form text field describing the event "verb"
- **verb_category**
Also a free form text field describing the event "verb", but intended to group related verbs together (eg "travel" for verbs: "flew", "drove")
- **geo_index**
If the event geotag maps into an entity from the parent document then this field is the "index" of that entity. The "geo_index" can also be directly specified as an entity index or an entity type (in "iterateOver" cases), and the geotag is then derived.
- **geotag**
 - **lat**
String containing a floating point representation of latitude
 - **lon**
String containing a floating point representation of longitude

The result of the association specification above can be seen in the sample output below:

Example Event Output

```
{
  //...
  "associations" : [
    {
      "entity1" : "robbery gun",
      "entity1_index" : "robbery gun/criminalactivity",
      "verb" : "reported",
      "verb_category" : "crime",
      "geotag" : {
        "lat" : "38.9051666534795",
        "lon" : "-77.0121735726172"
      },
      "geo_index" : "1100 b/o 1st st nw washington dc/place",
      "assoc_type" : "Event"
    }
  ],
  //...
}
```

In the sample output above please note that the Infinite harvester automatically generates the following fields as appropriate:

- **event_type**

"Event", "Fact", "Summary"

The "assoc_type" field sub-categorizes the "event" object into one of three types, "Event", "Fact", or "Summary". Examples provided below should make the distinction clearer, but it can be simply described as follows:

- "Event": link multiple entities (via "entity1_index", "entity2_index", "geo_index") and represent a transient activity (eg travel)
- "Fact": link multiple entities like "Events" but represent (transient or permanent) relationships (eg being president)
- "Summary": generally link 1 entity to a free text (eg a quotation: "Obama says...").

Specifying Multiplicative Associations

Multiplicative association are associations that are created by "multiplying" a combination of entities, locations, and times together to determine the number of associations to extract from the source data. For example, in the following sample document describing a terrorist attack one terrorist (perpetrator) attacked two different types of victims (police officers and military personnel) in Sri Lanka. The association specification uses the multiplicative format to create events using the following math to determine the total number of associations: Entity1 (PersonPerpetrator) * Entity2 (VictimType) * Geo_index (Location) = Total Number of Associations.

The Structured Analysis Harvester supports the creation of events using the following specification format for multiplicative events:

Multiplicative Event Specification Example

```
{
  //...
  "associations" : [
    {
      "iterateOver" : "entity1/entity2/geo_index",
      "verb" : "attacked",
      "verb_category" : "assault/attack",
      "entity1" : "PersonPerpetrator",
      "entity2" : "VictimType",
      "geo_index" : "Location",
      "time_start" : "$SCRIPT( return _doc.metadata.incidentdate[0]; )"
    }
  ],
  //...
}
```

Multiplicative Associations are specified by specifying which entity types to use to populate the entity1, entity2, geo_index, time_start, and time_end fields of the events created. The iterateOver field is used to specify the order in which the entity types to use are multiplied to determine the total number of associations to create.

- **iterateOver**

The iterateOver field specifies the order in which the Structured Analysis Harvester multiplies each field in order to create the right number and type of events. Each of the entity fields to use are separated by the '/' character and specifies the entity type used to populate the matching field in the event object.



Association fields generated from the entity loop are placed in "_iterator". For example, for "iterateOver": "entity1/entity2/geo_index", an _iterator object with the following fields is available in the Javascript: "_iterator.entity1_index", "_iterator.entity2_index", "_iterator.geo_index".

These fields can be usefully used together with "creationCriteriaScript" scriptlets to filter out unwanted associations, eg when looping over entity1 and entity2 with the same entity type, the following script would ensure the association didn't involve the same entity:

```
"creationCriteriaScript": "$SCRIPT( return _iterator.entity1_index != _iterator.entity2_index; )", "iterateOver": "entity1/entity2", "entity1": "EmailAddress", "entity2": "EmailAddress", //etc
```

The creationCriteriaScript runs before the association is generated (so can be safely used to remove items that would return errors).

In the example source below there are four entities that are being shown: one location entity, two victim entities, and one perpetrator entity. These visible entities are the entities referenced in the example specification above.

Multiplicative Event Source Example

```
{
  //...
  "entities" : [
    {
      "actual_name" : "Batticaloa,North Eastern Province,Sri Lanka",
      "dimension" : "Where",
      "disambiguated_name" : "Batticaloa,North Eastern Province,Sri Lanka",
      "doccount" : NumberLong(18),
      "frequency" : 1,
      "index" : "batticaloa,north eastern province,sri lanka/location",
      "geotag" : {
        "lat" : "7.7166667",
        "lon" : "81.7"
      },
      "ontology_type": "countrysubsidiary",
      "totalfrequency" : NumberLong(18),
      "type" : "Location"
    },
    {
      "actual_name" : "Targeted, Police, Adult from Sri Lanka",
      "dimension" : "Who",
      "disambiguated_name" : "Targeted, Police, Adult from Sri Lanka",
      "doccount" : NumberLong(47),
      "frequency" : 3,
      "index" : "targeted, police, adult from sri lanka/victimtype",
      "totalfrequency" : NumberLong(161),
      "type" : "VictimType"
    },
    {
      "actual_name" : "Targeted, Military, Adult, Combatant from Sri Lanka",
      "dimension" : "Who",
      "disambiguated_name" : "Targeted, Military, Adult, Combatant from Sri Lanka",
      "doccount" : NumberLong(20),
      "frequency" : 1,
      "index" : "targeted, military, adult, combatant from sri lanka/victimtype",
      "totalfrequency" : NumberLong(147),
      "type" : "VictimType"
    },
    {
      "actual_name" : "Secular/Political/Anarchist from Sri Lanka",
      "dimension" : "Who",
      "disambiguated_name" : "Secular/Political/Anarchist from Sri Lanka",
      "doccount" : NumberLong(200),
      "frequency" : 1,
      "index" : "secular/political/anarchist from sri lanka/personperpetrator",
      "totalfrequency" : NumberLong(200),
      "type" : "PersonPerpetrator"
    }
  ],
  //...
},
// ...
}
```

The following Multiplicative Event Output example shows how the Structured Analysis Harvester would generate two events from the source data and specification show above:

Multiplicative Event Output Example

```
{
  //...
  "associations" : [
    {
      "entity1" : "secular/political/anarchist from sri lanka",
      "entity1_index" : "secular/political/anarchist from sri
lanka/personperpetrator",
      "verb" : "attacked",
      "verb_category" : "assault/attack",
      "entity2" : "targeted, police, adult from sri lanka",
      "entity2_index" : "targeted, police, adult from sri lanka/victimtype",
      "time_start" : "09/07/2005",
      "geotag" : {
        "lat" : "7.7166667",
        "lon" : "81.7"
      },
      "geo_index" : "batticaloa,north eastern province,sri lanka/location",
      "assoc_type" : "Event"
    },
    {
      "entity1" : "secular/political/anarchist from sri lanka",
      "entity1_index" : "secular/political/anarchist from sri
lanka/personperpetrator",
      "verb" : "attacked",
      "verb_category" : "assault/attack",
      "entity2" : "targeted, military, adult, combatant from sri lanka",
      "entity2_index" : "targeted, military, adult, combatant from sri
lanka/victimtype",
      "time_start" : "09/07/2005",
      "geotag" : {
        "lat" : "7.7166667",
        "lon" : "81.7"
      },
      "geo_index" : "batticaloa,north eastern province,sri lanka/location",
      "assoc_type" : "Event"
    }
  ],
  //...
}
```



- If the "iterateOver" field contains neither "," or "/" ("," is for additive associations, see below) then it is treated as an iterator over a metadata field, just as described under [Specifying Entities](#), section "Create Entities from Arrays of Items".
- To iterate just over a single entity type, use "dummy", eg "entity1/dummy" or "entity2,dummy" (The '/' vs ',' are equivalent in this case).

Specifying Additive Associations

Additive associations cover the less common case where (eg) 2 entity types have the same number of elements and are ordered "in lock step". For example:

```
"entities": [
  { "index": "alex/person", ... },
  { "index": "craig/person", ... },
  { "index": "baltimore/city", ... },
  { "index": "washington dc/city", ...},
  ...
]
```

In this case the additive association specification:

```
{
  "iterateOver": "entity1,entity2", // note "," instead of "/"
  "entity1": "Person",
  "entity2": "City",
  "verb_category": "lives in",
  ...
}
```

Would generate the 2 associations "alex/person lives in baltimore/city" and "craig/person lives in washintgon dc/city".

Further Reading

- [Specifying Document Level Geographical Location](#)
- [Specifying Entities](#)
- [Transforming data with JavaScript](#)

Specifying Document Level Geographical Location

The Structured Analysis Harvester supports assigning a geographical location to a document using latitude and longitude values. There are two ways in which you can have the harvester assign a location to a document in the harvesting process:

Specify the Latitude and Longitude Explicitly

The docGeo object allows you to specify the fields to extract from your structured data source to use as your latitude and longitude values. In the example below the "lat" (or latitude) value is being extracted from the document's metadata.latitude field while the "lon" (or longitude) value is being extracted from the metadata.longitude field of the extract document.

Source.structuredAnalysis object

```
"structuredAnalysis" : {
  ...
  docGeo : {
    "lat" : "$metadata.latitude",
    "lon": "$metadata.longitude"},
  ...
}
```

Specify City, State/Province, and Country

The second method specifying a document level geographical location is to specify city, state/province, and country in the the docGeo object as shown in the example below (**Note:** The example below uses embedded JavaScript. Information on using JavaScript within the StructuredAnalysis object can be found here: [Transforming data with JavaScript](#)):

Source.structuredAnalysis object

```
"structuredAnalysis" : {
  ...
  "docGeo" : {
    "city" : "$SCRIPT( return _doc.metadata.location[0].citystateprovince.city;
)",
    "stateProvince" : "$SCRIPT( return
_doc.metadata.location[0].citystateprovince.stateprovince; )",
    "country" : "$SCRIPT( return _doc.metadata.location[0].country; )"
  },
  ...
}
```

The Structured Analysis Harvester attempts to use the city, state/province, and country information supplied to retrieve latitude and longitude values from the Infinite GeoReference table via `GeoReference.enrichGeoInfo()` method.

Note: The `enrichGeoInfo()` function can be called specifying whether or not to return an exact match (Boolean `exactMatchOnly`). If the `exactMatchOnly` parameter is set to `false` the `enrichGeoInfo` function will attempt to broaden its search if it is unable to match the original search parameters (i.e. city, state, country). Ultimately the `enrichGeoInfo` function will apply the average latitude and longitude values for a location's country (the geographical center point) if it is not possible to obtain a more precise match.

Alternatives

In some cases, it will not be clear what geographical type a field is (eg a freeform field that might be city, state, or country). The geographical specification allows you to specify alternatives, eg:

```
"structuredAnalysis" : {
  ...
  "docGeo" : {
    "city" : "$SCRIPT( return _doc.metadata.cityOrStateOrProvinceOrCountry; )",
    "country" : "USA"
  "alternatives": [
    {
      "stateProvince" : "$SCRIPT( return
_doc.metadata.cityOrStateOrProvinceOrCountry; )",
      "country" : "USA"
    },
    {
      "country" : "$SCRIPT( return _doc.metadata.cityOrStateOrProvinceOrCountry;
)"
    }
  ]
  },
  ...
}
```

The alternatives are tried in order until one of them works or there are no more to try.

Further Reading

- [Specifying Entities](#)
- [Specifying Associations](#)
- [Transforming data with JavaScript](#)

Specifying Entities

What is An Entity?

Entities are the who, what, and where's contained within a record (i.e. people, places, and things).

Basic Entity Specification

The following code demonstrates how to specify a basic Where (Place) entity (**Note:** The sample entity specification and sample entity output

below is extracted from a sample [MySQL Database Source](#).):

Basic Entity Specification Example

```
{
  //...
  "entities" : [
    {
      "disambiguated_name" :
"$metadata.blocksiteaddress,$metadata.city,$metadata.state",
      "dimension" : "Where",
      "type" : "Place",
      "geotag" : {
        "lat" : "$metadata.latitude",
        "lon" : "$metadata.longitude"
      },
      "ontology_type": "point"
    },
    ],
  //...
}
```

In the basic example above the following fields have been specified:

- **disambiguated_name**
For a given "type", this is (aside from case) a unique identifier for the entity
- **dimension**
One of "Who" (people, organizations), "Where" (places), or "What" (everything else)
- **type**
The entity type, i.e. if dimension is equal to What, type might be equal to Automobile, Airplane, Ship, etc.
- **geotag**
 - **lat**
String containing a floating point representation of latitude
 - **lon**
String containing a floating point representation of longitude

Data is extracted from the source using the **\$ operator**. For example, in the case of the **geotag.lat** field the data is extracted from the **metadata.latitude** field using the following definition:

```
"lat" : "$metadata.latitude"
```

The **\$ operator** can also be used to combine multiple source data fields into more complex literal strings as used to specify the document's description field:

```
"description" : "$metadata.reportdatetime: $metadata.offense,$metadata.method
was reported at: $metadata.blocksiteaddress"
```

Which is converted into the following literal string:

```
"description" : "Mar 10, 2011 12:00:00 AM: ROBBERY GUN was reported at the 1100 B/O
1ST ST NW"
```

Note: More advanced data transformations can be performed within the Structured Analysis Harvester using JavaScript as documented here: [Transforming data with JavaScript](#).

The result of the entity specification above can be seen in the sample output below:

Example Place Entity Output

```
{
  //...
  "entities" : [
    {
      "actual_name" : "1100 B/O 1ST ST NW WASHINGTON DC",
      "dimension" : "Where",
      "disambiguated_name" : "1100 B/O 1ST ST NW WASHINGTON DC",
      "doccount" : 3,
      "frequency" : 1,
      "index" : "1100 b/o 1st st nw washington dc/place",
      "geotag" : {
        "lat" : "38.9051666534795",
        "lon" : "-77.0121735726172"
      },
      "ontology_type": "point",
      "relevance" : "0",
      "totalfrequency" : 3,
      "type" : "Place"
    },
  ],
  //...
}
```

In the sample output above please note that the Infnit.e harvest automatically generates the following fields as appropriate:

- **doccount**
The number of documents in which the entity occurs in the Infnit.e database
- **frequency**
The number of times the entity occurs in the document (**Note:** the system defaults the frequency count to 1 however it is possible to specify a frequency count within a source document)
- **totalfrequency**
The number of times the entity occurs in all documents in the Infnit.e database
- **relevance**
A value between 0 and 1(in the form of a string containing a floating point number), indicating the entity extraction engine's "opinion" on the entity's relevance within the document

Create Entities from Arrays of Items - Basic Example

The Structured Analysis Harvester is capable of extracting data contained with JSON arrays (i.e. `_doc.metadata.someTypeOfEntity[]`) using the **iterateOver** field of the entity object as show below:

Basic Array of Entities Specification Example

```
"entities" : [
  //...
  {
    "iterateOver" : "location",
    "disambiguated_name" : "$SCRIPT( return
_iterator.citystateprovince.city+', '+_iterator.citystateprovince.stateprovince+', '+_it
erator.country; ); )",
    // (use _value if the iterating field is a primitive type eg a string;
    // _iterator if it is an object; _iterator.X to access the field X of the object,
    etc)
    "actual_name": "$citystateprovince.city,$citystateprovince.stateprovince,$country",
    // (or use the $ format - note that when using iterateOver,
    // you can't access $metadata.FIELD any more, the $ is offset from the last clause
    of "iterateOver")
    "useDocGeo" : true,
    "dimension" : "Where",
    "type" : "Location"
  }
  //...
]
```

In the example above the **iterateOver** value is set to "location" meaning that the Structured Analysis Harvester will iterate (or loop) over the **meta data.location[]** objects and create an entity for each object in the array. The source example below shows a location array with one location object:

Basic Array of Entities Source Example

```
"metadata" :
{
  //...
  "location" : [
    {
      "region" : "East Asia-Pacific",
      "citystateprovince" : {
        "stateprovince" : "Narathiwat",
        "city" : "Sungai Padi"
      },
      "country" : "Thailand"
    }
  ],
  //...
}
```



Nesting is supported using the "dot notation" eg if in the above instance, the location was inside an object (or array of objects) called "more_information", then the "iterateOver" field would be set to "more_information.location".

(This would be equivalent to the less tidy technique of nesting the Entity Specification JSON object, the first having "iterateOver": "more_information", and containing a second Entity Specification JSON object identical to the original example).

A few useful tips for using "iterateOver":

- Arrays and objects are treated equally in the dot-notation (ie an object is just treated like an array of size 1)
 - eg for both "{ A: { B: { C: "value" } } }" and "{ A: [B: [{ C: [value] }]] }", you would use "iterateOver": "A.B.C" to get to "value"

- If any of the fields point to primitives (eg **B**: ["val1", "val2"] in the example above) then an error is thrown unless the "creation criteria" script for the nested object is specified (C in this example).
 - (You can still throw errors from the script by checking if "(_iterator==null)" if you want to) This enables writing objects that will handle fields being either primitives or objects (eg by checking vs _iterator and _value).
- For non-nested entity specification objects, the first field in the "iterateOver" field refers to the metadata object, eg "iterateOver": "location" refers to "_doc.metadata.location".
 - (For nested objects, the first field refers to the "parent" object, but you shouldn't be using nesting now that dot notation is available!)
- Reminder: if you are iterating over:
 - An object, then use "_iterator.FIELD" in scripts, "\$FIELD" for normal strings.
 - (Note that "\$metadata.X" won't work inside "iterateOver" clauses, you have to use constructs like "\$SCRIPT(return _doc.metadata.X[0];)" to get at the top-level fields. We will probably fix this at some point.)
 - A value then use "_value" in scripts, "\$" for normal strings.

The example below demonstrates how a location entity is created from source data (**Note**: The full source and sample output created from the source can be found here: [XML File Source](#)).

Basic Array of Entities Output Example

```
"entities" : [
  //...
  {
    "actual_name" : "Sungai Padi,Narathiwat,Thailand",
    "dimension" : "Where",
    "disambiguated_name" : "Sungai Padi,Narathiwat,Thailand",
    "doccount" : NumberLong(51),
    "frequency" : 1,
    "index" : "sungai padi,narathiwat,thailand/location",
    "geotag" : {
      "lat" : "6.085833",
      "lon" : "101.881389"
    },
    "ontology_type": "city",
    "totalfrequency" : NumberLong(51),
    "type" : "Location"
  },
  //...
],
```

Create Entities from Arrays of Items - Advanced Example

The following example of how to specify entities from within an array of items is similar to the basic example above but expands on it by showing how to extract multiple entities from each array item, by nesting them using the "entities" field:

Advanced Array of Entities Specification Example

```
"entities" : [
  //...
  {
    "iterateOver" : "victim",
    "entities" : [
      {
        "disambiguated_name" : "$SCRIPT( getVictim( _iterator ); )",
        "frequency" : "$FUNC( getVictimCount(); )",
        "dimension" : "Who",
        "type" : "VictimType"
      },
      {
        "disambiguated_name" : "$SCRIPT( getVictim( _iterator ); )",
        "frequency" : "$hostagecount",
        "dimension" : "Who",
        "type" : "HostageType"
      }
    ]
  }
  //...
]
```

The entity specification above is designed to extract two entities from each victim object in the example source below. In the source sample below note that there are three counts for each victim object: wounded count, dead count, and hostage count. The two entities to be created from each victim object are identical except for following two differences:

- **frequency**
 - Entity 1: the frequency count will be equal to the sum of "woundedcount" + "deadcount"
 - Entity 2: the frequency count will be equal to "hostagecount"
- **type**
 - Entity 1: type will equal "VictimType"
 - Entity 2: type will equal "HostageType"

Advanced Array of Entities Source Example

```
"metadata" :
{
  //...
  "victim" : [
    {
      "child" : "No",
      "indicator" : "Targeted",
      "nationality" : "Thailand",
      "targetedcharacteristic" : "Unknown",
      "woundedcount" : "1",
      "deadcount" : "1",
      "definingcharacteristic" : "None",
      "combatant" : "No",
      "hostagecount" : "2",
      "victimtype" : "Police"
    }
  ],
  //...
}
```

The example output below demonstrates how two victim entities are created from the source data and entity specification found above:

Advanced Array of Entities Output Example

```
"entities" : [
  //...
  {
    "actual_name" : "Targeted, Police, Adult from Thailand",
    "dimension" : "Who",
    "disambiguated_name" : "Targeted, Police, Adult from Thailand",
    "doccount" : NumberLong(186),
    "frequency" : 2,
    "index" : "targeted, police, adult from thailand/victimtype",
    "totalfrequency" : NumberLong(478),
    "type" : "VictimType"
  },
  {
    "actual_name" : "Targeted, Police, Adult from Thailand",
    "dimension" : "Who",
    "disambiguated_name" : "Targeted, Police, Adult from Thailand",
    "doccount" : NumberLong(40),
    "frequency" : 2,
    "index" : "targeted, police, adult from thailand/hostagetype",
    "totalfrequency" : NumberLong(53),
    "type" : "HostageType"
  },
  //...
],
```



Nesting is not recommended except in cases where nested fields can refer to either primitives or objects. Just use the "dot notation" described above where possible.

You can nest entities an arbitrary number of times, provided there is a valid "iterateOver" field specified at every level. You can also make an entity object both specify an entity and contain an array, although this is not recommended.

Further Reading

- [Specifying Document Level Geographical Location](#)
- [Specifying Associations](#)
- [Transforming data with JavaScript](#)

Transforming data with JavaScript

Overview

The Infinet platform supports scripting the transformation of source data using JavaScript via Rhino, Mozilla's open-source JavaScript implementation (<http://www.mozilla.org/rhino/>). The following document provides an introduction to specifying JavaScript based data transformation via the Structured Analysis Harvester object.



Note that unless turned off from the [configuration files](#) (via the "harvest.security" property), Javascript is prevented by the Java security manager from doing the following:

- "Internal" network access (ie to addresses 127.*.* , 10.*.* or 192.168.*.)
- File access.

As noted above the beta version of Infit.e currently supports data transformation via JavaScript functions. To enable the use of JavaScript you need to include the **scriptEngine** key/value pair within the Structured Analysis object:

Source.structuredAnalysis object

```
"structuredAnalysis" : {  
  ...  
  "scriptEngine" : "JavaScript",  
  ...  
}
```

Note: Although scripting support within Infit.e is currently limited to JavaScript it is possible that support for other languages will be added in the future.

Importing JavaScript Functions

The Infit.e Structured Analysis Harvester supports importing of JavaScript functions in two ways currently:

- The inclusion of functions in the **script** key/value pair (see below for an example).
- External JavaScript files that can be accessed via URL by the Infit.e Structured Analysis Harvester (see the **scriptFiles** key/value pair below for an example);

Source.structuredAnalysis object

```
"structuredAnalysis" : {  
  ...  
  "script" : "function getEventType() { var s = _value; ...; return s; }",  
  "scriptFiles" : [ "http://localhost/script1.js",  
    "http://localhost/script2.js" ],  
  ...  
}
```



JavaScript functions imported via either of the two means described above are passed to the Rhino script engine via the `ScriptEngine.eval` method which allows the functions to be called within the scope of the current document being harvested. Examples of imported functions can be found below.

Note: the "script" context does not have access to any of the objects described below (like "_doc"), it can only be used for declaring functions to be used in the entity/association/docGeo scriptlets.

Accessing Document Data via the _doc Object

As the Structured Analysis Harvester iterates over documents to be harvested it passes each document to the ScriptEngine (if a ScriptEngine has been instantiated, see Specifying the ScriptEngine above) via the `ScriptEngine.put` method. The JSON based document passed into the ScriptEngine is then converted into an object via JavaScript's `eval()` method (i.e. `var _doc = eval('document')`). Fields within the document are then available to functions and inline scripts using the JavaScript dot or subscript operators as shown below:

```
var description = _doc.metadata.description[0];
```

Note that `_doc` is persistent across all entity/association script calls, and therefore can be used to store temporary variables, eg for deduplication, eg:

```
"$SCRIPT( if (null == _doc.dedupMap) _doc.dedupMap = new Object(); /* ... */ if  
(_doc.dedupMap[val] == null) { _doc.dedupMap[val] = 1; return val; } else return null;  
)"
```



Note that `"_doc.metadata.FIELDNAME[0].*"` is equivalent to `"$metadata.FIELDNAME.*"` (the [0] disappears because the \$ method just treats arrays as objects equal to the first element in the array - to access other elements, the "\$SCRIPT" technique must be used)

Basic Field Level Transformations

Individual fields from a data source can be transformed using JavaScript by either calling an imported function (described above) or by using inline JavaScript.

Source.structuredAnalysis object

```
"structuredAnalysis" : {
  ...
  "title" : "$FUNC( getDocumentTitle(); )",
  "description" : "$SCRIPT( return 'Description: ' +
  _doc.metadata.description[0];)",
  ...
}
```

Inline JavaScript

In the example JSON above the "description" field contains inline JavaScript enclosed within \$SCRIPT(). During the harvesting process the Structured Analysis Harvester:

1. Extracts the JavaScript code contained within the \$SCRIPT() block
2. Wraps the script with the following generic script block:

```
function getValue() { ... }
```

3. Passes the function to the ScriptEngine using the ScriptEngine.eval() method
4. Calls the getValue() method using the .invokeFunction() method.

Calling Imported JavaScript Functions

Calling functions previously imported into the ScriptEngine is done by enclosing the function name to be called within the \$FUNC() block as shown above in the "title" field. During the harvesting process the harvester:

1. Extracts the name of the function to execute from the \$FUNC() block
2. Calls the specified function using the .invokeFunction() method.

Note: \$FUNC only has meaning when it encloses the entirety of the string. For calling functions inside \$SCRIPT blocks (described above), just invoke the function normally.

Using the _iterator Object

When specifying entities or events to create from source data it is possible to specify that data be extracted from JSON arrays within the metadata field using the IterateOver field (see [Specifying Entities](#) and [Specifying Events](#) for more information). When the harvester iterates over a JSON array each item in the array is passed into the ScriptEngine and is made accessible via an object named: _iterator.

The code block below demonstrates how a field within a document's metadata might hold items in a typical JSON array object.

```
{
  ...
  metadata : {
    cars : [
      {"make" : "Ferrari", "model" : "599", "year" : "2011"},
      {"make" : "Ferrari", "model" : "California", "year" : "2011"},
      {"make" : "Ferrari", "model" : "458 Italia", "year" : "2011"}
    ]
  }
  ...
}
```

The sample JavaScript below demonstrates how to access each field within an array item as the harvester iterates over the array:

```
var make = _iterator.make;
var model = _iterator.model;
var year = _iterator.year;
```

Passing Values to Scripts via `_value`

In the above case, if the script engine is iterating over an array of primitives (eg "`_doc.metadata.cars == ['Ferrari', 'Alfa Romeo', 'Fiat']`") then the values are passed into `_value` instead of `_iterator`.

```
var s = _value;
```

Note: Each time a value is passed into the ScriptEngine the content of `_value` is over written.

Extracting Data from Arrays with `_index`

The harvester supports passing an index value into the ScriptEngine that can be used to access a specific item in an array by its index. An example of how the `_index` variable can be used is show below:

```
var make = _doc.metadata.cars[_index].make;
```

JavaScript Error Messages

If the ScriptEngine encounters errors when executing a script the Structured Analysis Harvester traps the error and stores it in memory until the process of harvesting all of the documents for a given source has completed. When a source has been harvested the harvester updates the Harvest object within the Source document (`harvest.harvested`, `harvest.harvest_status`, and `harvest.harvest_message` fields). If the harvester encountered JavaScript errors it writes the top five errors to the `harvest.harvest_message` field including the number of times each error was encountered.

Note: Field level JavaScript errors will not prevent a document or source from being harvested.

Creation Criteria Scripts

Both [entity](#) and [association](#) specification objects provide a field called "creationCriteriaScript". This must be JavaScript (though you still need to set the engine and enclose in either `$SCRIPT` or `$FUNC`), and you can return one of two things from it:

- A boolean, in which case the entity object is only created if
- A string, in which case any non-null string is treated like a boolean false, and in addition the string is logged as an error that can be accessed from the "harvest.harvest_message" field of [sources](#).

 The creation criteria script is executed **before** any other scripts in the specification object.

Lookup tables in the Unstructured Analysis Handler

It is possible to add lookup tables from JSON shares that can be used in all the javascript scripts in the structured analysis handler (and also the [unstructured analysis handler](#)).

These lookup tables to provide a limited form of aliasing a harvest time - also check out the [full query-time aliasing capability](#) - in addition to many other cases where a potentially large and dynamic lookup table would be useful.

Using the lookup technology is easy:

- At the top level of the "structuredAnalysis" object, create a "caches" object that consists of the following:
 - For every lookup table, a local name you specify and then the "_id" field of a JSON share (see [share API documentation](#), or uploaded via the [File Uploader](#)). For example:

```
"structuredAnalysis": {
  "caches": {
    "myLookupTable": "4e0c7e99eb5af0fbdcfbf697"
  }
}
```

- Then within any script in the "structuredAnalysis", you can access the JSON object by the local name specified as above. For example, say the following JSON object has been uploaded:


```
{
  //...
  "US": "United States", "USA", "United States of America",
  "UK": "United Kingdom", "Great Britain", "GB",
  //...
}
```

Then the lookup table could be used as follows:

```
{
  "structuredAnalysis": {
    // (caches object specified as above)
    //...
    "entities": [
      //...
      {
        "iterateOver": "geo.countries",
        "disambiguatedName": "$SCRIPT( return myLookupTable[ _value ];)",
        "type": "Country"
      }
    ],
    //...
  }
}
```

Further Reading

- [Specifying Document Level Geographical Location](#)
- [Specifying Entities](#)
- [Specifying Associations](#)

Unstructured Analysis - Overview

The Infinite Unstructured Analysis Harvester is designed to take specified regular expressions and enrich the data via the use of regular expressions and specified text extraction modules.

The example `Source.structuredAnalysis` object below demonstrates the basic features of specifying how to enrich harvested structured data.

[There is also a reference page for the Unstructured Analysis configuration object.](#)

The Harvesting Process

Specifying a document's Header/Footer

The harvester first evaluates the strings specified for the **headerRegex** and **footerRegex**. These are each regular expression strings that specify if a document has a header or if the document has a footer. This is useful to be able to negate a lot of formatting that normally occurs in headers/footers that would not result in correctly extracted entities or events. It is important to note that the patterns are matched in DOTALL mode.

For Example:

Sample Document

Address: 123 Sample Dr. Woodbridge, VA 22191

To: John Doe

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

Sincerely,

Sample Person

Sample Person

Address

Company

Source.unstructuredAnalysis object

```
{
  "headerRegex" : "^.*\++",
  "footerRegex" : "_+.*$",
  "meta" : [ {
    "context": "Body",
    "fieldName": "addressMetadata",
    "scriptlang": "regex",
    "script": "address:\\s*(.*)",
    "groupNum": 1,
    "flags": "iD" // (the D negates the "dotall" ie stops the extraction at a
newline character)
  } ]
}
```

The above shows valid regular expressions to use to parse the sample documents header and footer, and will create the following metadata object:

Generated metadata from example above

```
{
  "metadata": {
    "addressMetadata": [
      "123 Sample Dr. Woodbridge, VA 22191"
    ]
  }
}
```

Specifying data as metadata

After checking if a header/footer is specified, the harvester then processes all Objects in the **meta** list. Meta Object's variable **fieldName** will be the 'key' term or the label in the list. **Context** is an enum variable used to specify where to check for the specified regular expression, **script** (with **scriptlang**: "regex"). **Context** has 5 possible values:

- First - If this is specified, the script/regex is applied before any text cleansing (ie on the entirety of the raw content)
- Header - If this is specified, **headerRegex** must also be specified, and then script/regex is only applied to the "header" section
- Footer - If this is specified, **footerRegex** must also be specified and then script/regex is only applied to the "footer" section
- Body - Checks the text not specified as the header or footer. If there is no body to the feed, the description of the feed will be checked.

- All - This checks for the regular expression in all areas

The regular expression used to find the data labeled by **fieldName** is placed in the **script** string. This regular expression makes use of groups, specified by **groupNum**. A group is a pair of parentheses used to group subpatterns. For example, `h(a|)t` matches hat or hit. A group also captures the matching text within the parentheses. *For example:*

```
{
  input:   abbc
  pattern: a(b*)c
}
```

causes the substring `bb` to be captured by the group `(b*)`. If the use of groups is not desired, `groupNum` should be set to the number 0 (zero), i.e. to get the entirety of the matching pattern.

In the case that the desired purpose of the regular expression is to do a replace, this replace string can be specified in `replace`.

For Example:

Source.unstructuredAnalysis.meta object

```
{
  "fieldName" : "Race",
  "context"   : "All",
  "regex"    : "C/[F|M]",
  "groupNum"  : 0,
  "replace"   : "Caucasian"
}
```

would find the instance `C/M` or `C/F` in a document and extract that it is important to note that the Race is Caucasian. The same can be done to extract `M` or `F` as a Sex meaning Male or Female.

Using Javascript to generate metadata

For power users, metadata can be generated from the content using javascript. This gives a huge amount of flexibility to apply site/source-specific knowledge to pull out metadata that can be turned into entities or associations.

By default, only one input variable is included: `"text"`, which corresponds to the `"fullText"` field of the [document JSON](#).

i If there are multiple `"meta"` objects with the same `"fieldName"`, then they form a `"pipeline"`, with each new object taking the old array, in the `"_iterator"` variable, and then overwriting the previous entry's result.

There are also a few flags that provide additional variables in the javascript:

- `"m"` to get `"_doc.metadata"`, written into the variable `"_metadata"`
 - (for example this flag can be used to copy a subset of the fields from one `fieldname` to another, before using the `"metadataFields"` field in the `"structuredAnalysis"` object to delete the larger field)
- `"d"` to get `"_doc"`, written into the variable `"_doc"`,
- `"t"` to return the full text of the document into `"text"`.
 - *If the `"flags"` field is not specified, this is returned by default. If the `"flags"` field is specified, then `"t"` must be included or the `"text"` variable is not populated.*

For example, consider the following javascript, which (like the regex example above) pulls the address out of the example letter format.

Simple javascript to be embedded in "meta" object

```
var i = text.indexOf("address:");
var j = text.indexOf("\n", i); // (starts looking after address)
var returnVal = null;
if (i >= 0 && j >= 0) {
    returnVal = text.substring(i, j).trim();
}
returnVal;
```



Note the slightly unusual way in which the object/primitive is "returned": whatever is evaluated on the final line. The easiest way of managing this is to have a single standalone line containing a previously-declared "var" at the end.

Then this would be embedded as follows in a "meta" object:

Source.unstructuredAnalysis object

```
{
  "headerRegex" : "^.*\*+",
  "footerRegex" : "_+.*$",
  "meta" : [ {
    "context": "Body",
    "fieldName": "addressMetadata",
    "scriptlang": "javascript",
    "script": "var i = text.indexOf(\"address:\");\nvar j = text.indexOf('\n',
i);\nvar returnVal = null;\nif (i >= 0 && j >= 0) {\nreturnVal = text.substring(i,
j).trim();\nreturnVal;"
  } ]
}
```

Obviously the javascript can also return more complex objects, arrays of objects, or array of primitives.

Note that using "\n"s in the embedded script is recommended, since then runtime javascript errors (reported in the "harvest.harvest_message" field of the source object) will map the line number.

[The same security restrictions as for the Structured Analysis Harvester apply.](#)

Using XPath to generate metadata

Neither regex nor javascript are well suited for extracting fields from HTML and XML (particularly since the current Javascript engine, the Java version of Rhino, does not support DOM).

As a result, Infini.t.e supports [XPath 1.0](#) (with one minor extension to allow combined XPath regex).

Consider the Following Examples:

```

<html>
<body>
  <b>Check out this really great site for News & more!</b>
  <a href="http://www.bbc.com">BBC</a>
  <i>List of my favorite topics</i>
  <ul id="favTopics">
    <li>Sport</li>
    <li>TV</li>
  </ul>
  <i>List of my not-so favorite topics</i>
  <ul class="ugly">
    <li>The Topic of Radio</li>
    <li>The Topic of News</li>
  </ul>
</body>
</html>

```

```

"meta": [{
  "context": "First",
  "fieldName": "boldText",
  "scriptlang": "xpath",
  "script": "//b[1]" //can also be specified as /html[1]/body[1]/b[1]
},
{
  "context": "First",
  "fieldName": "boldTextDecoded",
  "scriptlang": "xpath",
  "script": "//b[1]",
  "flags": "H" //will HTML-decode resulting fields
},
{
  "context": "First",
  "fieldName": "favoriteTopics",
  "scriptlang": "xpath",
  "script": "//ul[@id='favTopics']/li[*]" //The asterisk wildcard character can be used
to specify all items
},
{
  "context": "First",
  "fieldName": "notFavoriteTopics",
  "scriptlang": "xpath",
  "script": "//ul[@class='ugly']/li[*]regex[The Topic of (.*)]", //Regex can be
specified as a content filter
  "groupNum": 1 //group number of regex
}
]

```

would generate the following different outputs (note the use of "groupNum" to select which capturing group to display):

```
"metadata": {
  "boldText": [ "Check out this really great site for News & more!" ],
  "boldTextDecoded": [ "Check out this really great site for News & more!" ],
  "favoriteTopics": [ "Sport", "TV" ],
  "notFavoriteTopics": [ "Radio", "News" ],
}
```

This final example, shows how "groupNum": -1 can be used to grab the entire object instead of just the text. Note this is now deprecated, use "flags": "o" for the same effect (See below).

Consider the HTML block:

```
<html>
  <body>
    <a href="http://www.bbc.com">BBC</a>
  </body>
</html>
```

Then the following 2 XPath expressions:

```
"meta": [{
  "context": "First",
  "fieldName": "test1",
  "scriptlang": "xpath",
  "script": "//a[1]"
},
{
  "context": "First",
  "fieldName": "test2",
  // as above but with:
  "groupNum": -1
}]
```

would generate the following different outputs:

```
"metadata": {
  "test1": [ "BBC" ],
  "test2": [{
    "href": "http://www.bbc.com",
    "content": "BBC"
  }]
}
```

For reference, here is the complete set of flags for xpath:

- **'H'**: will HTML-decode resulting fields. (Eg "&#amp;" -> "&")
- **'o'**: if the XPath expression points to an HTML (/XML) object, then this object is [converted to JSON](#) and stored as an object in the corresponding metadata field array. (Can also be done via the deprecated "groupNum":-1)
- **'D'**: described above (also works for regex)

Lookup tables in the Unstructured Analysis Handler

It is possible to add lookup tables from JSON shares that can be used in all the javascript scripts in the unstructured analysis handler (and also the [structured analysis handler](#)).

These lookup tables to provide a limited form of aliasing a harvest time - also check out the [full query-time aliasing capability](#) - in addition to many other cases where a potentially large and dynamic lookup table would be useful.

Using the lookup technology is easy:

- At the top level of the "unstructuredAnalysis" object, create a "caches" object that consists of the following:
 - For every lookup table, a local name you specify and then the "_id" field of a JSON share (see [share API documentation](#), or uploaded via the [File Uploader](#)). For example:

```
"unstructuredAnalysis": {
  "caches": {
    "myLookupTable": "4e0c7e99eb5af0fbdcfbf697"
  }
}
```

- Then within any script in the "unstructuredAnalysis", you can access the JSON object by indexing "_cache" with the local name specified as above. For example, say the following JSON object has been uploaded:

```
{
  //...
  "US": "United States", "USA", "United States of America",
  "UK": "United Kingdom", "Great Britain", "GB",
  //...
}
```

Then the lookup table could be used as follows:

```
{
  "unstructuredAnalysis": {
    // (caches object specified as above)
    //...
    "meta": [
      //...
      {
        "fieldName": "disambiguatedCountryName",
        "context": "All",
        "fields": "m",
        "scriptlang": "javascript",
        "script": "_cache.myLookupTable[ _metadata.countryName[0] ];"
      }
    ],
    //...
  }
}
```

Config - Source - Add (DEPRECATED)

[/config/source/add/{sourceurl}/{sourcetitle}/{sourcedesc}/{extracttype}/{sourcetags}/{mediatype}/{communityId}](#)



Adds a new source to the system. This has been replaced by [Config - Source - Save](#)

Authentication

Required, see [Auth - Login](#)

Arguments

communityId (required)

Community ID to which you want to associate this source, can also be a regex ([community ID - regex](#)) provided that matches only a single

community.

sourceurl (required)
Address of the source

sourcetitle (required)
Name of the source

sourcedesc (required)
A description of the source documents

extracttype (required)
Type of source, currently one of "Feed", "File", "Database": at present only "Feed" can be added this way, use [Config - Source - Save](#) for others.

sourcetags (required)
comma delimited list of words describing the source

mediatype (required)
Type of source e.g. News, social, video, etc

Example

[http://infinite.ikanow.com/api/knowledge/source/add/www.newsource.com/Brand New Source/A description for brand new stuff/RSS/New,Tech,Brand/Blog/4cc0ebff97622e5914a70e83](http://infinite.ikanow.com/api/knowledge/source/add/www.newsource.com/Brand%20New%20Source/A%20description%20for%20brand%20new%20stuff/RSS/New,Tech,Brand/Blog/4cc0ebff97622e5914a70e83)



```
{
  response: {
    action: "Source"
    success: true
    message: "New source added successfully."
    time: 10
  }
}
```

Config - Source - Approve

`/config/source/approve/{source}/{communityId}`



Approves a source that has been submitted so that the source can be harvested.

Authentication

Required, see [Auth - Login](#). Approver must be a system administrator, or the owner/moderator of the community that the source is associated with.

Arguments

source (required)
Source ID or key you want to approve

communityId (required)
Community ID, or `regex`, that the source is associated with.

Example

<http://infinite.ikanow.com/api/config/source/approve/4e2eaeef99f86403e991b9ec/4e258f8a64b46403c55095ce>

Example Response




```
{
  response: {
    action: "Approve Source"
    success: true
    message: "Source approved successfully."
    time: 10
  }
}
```

Error Response



```
{
  response: {
    action: "Approve Source"
    success: false
    message: "Error approving source. You must be a community owner or
moderator to approve a source"
    time: 10
  }
}
```

Config - Source - Bad

/config/source/bad/{communityId-list}



Returns a list of sources that have sufficient harvest errors that they have been removed from the harvesting schedule (or have been turned off by an administrator).

Authentication

Required, see [Auth - Login](#)

Arguments

communityId-list (required)

Community ID, or IDs (comma-separated), or **community ID - regex** for which to retrieve sources.

Example

<http://infinite.ikanow.com/api/config/source/bad/4e258f8a64b46403c55095ce>

http://infinite.ikanow.com/api/config/source/bad/*system

Example Response



```
{
  "response": {
    "action": "Bad Sources",
    "success": true,
    "message": "Successfully returned bad sources",
    "time": 51
  },
}
```

```
"data": [
  {
    "_id": "4cbdb9eb5ed98e7bc8489270",
    "created": "Oct 19, 2010 11:31:55 AM",
    "modified": "Oct 19, 2010 11:31:55 AM",
    "url":
"http://english.aljazeera.net/Services/Rss/?PostingId=2007731105943979989",
    "title": "Al Jazeera: World News",
    "isPublic": true,
    "ownerId": "4e3706c48d26852237078005",
    "mediaType": "News",
    "key":
"http://english.aljazeera.net/services/rss/.postingid=2007731105943979989",
    "description": "Al Jazeera World News Articles",
    "tags": [
      "topic:politics",
      "World",
      "News",
      "industry:all"
    ],
    "communityIds": [
      "4c927585d591d31d7b37097a"
    ],
    "harvest": {
      "harvested": "Mar 1, 2012 1:43:32 AM",
      "harvest_status": "error",
      "harvest_message": "Read timed out",
      "doccount": 8153
    },
    "isApproved": true,
    "harvestBadSource": true,
    "extractType": "Feed"
  },
  {
    "_id": "4cbdb9ed5ed98e7b07499270",
    "created": "Oct 19, 2010 11:31:57 AM",
    "modified": "Oct 19, 2010 11:31:57 AM",
    "url": "http://www.army.mil/rss/feeds/health.xml",
    "title": "Army: Healthcare",
    "isPublic": true,
    "ownerId": "4e3706c48d26852237078005",
    "mediaType": "News",
    "key": "http://www.army.mil/rss/feeds/health.xml",
    "description": "Army Health Care Updates",
    "tags": [
      "topic:healthcare",
      "industry:healthcare",
      "Military",
      "healthcare"
    ],
    "communityIds": [
      "4c927585d591d31d7b37097a"
    ],
    "harvest": {
      "harvested": "Mar 1, 2012 9:41:23 AM",
      "harvest_status": "error",
      "harvest_message": "Connection refused",
      "doccount": 316
    },
  },
]
```

```
"isApproved": true,  
"harvestBadSource": true,  
"extractType": "Feed"  
}
```

```
]
}
```

Config - Source - Decline

`/config/source/decline/{source}/{communityId}`



Rejects a source (decline) that has been submitted for harvesting and removes the source from the sources database collection.

If the source has previously been active, then **Config - Source - Delete** must be used instead. Calling "decline" on active sources sets them to be inactive, but calling "decline" on now-inactive-but-previously-active sources returns with an error.

Authentication

Required, see [Auth - Login](#), use must be a system administrator, or the owner/moderator of the community that the source is associated with.

Arguments

source (required)
Source ID or key you want to decline

communityId (required)
Community ID, or [regex](#), that the source is associated with.

Example

<http://infinite.ikanow.com/api/config/source/decline/4e2eaee99f86403e991b9ec/4e258f8a64b46403c55095ce>

Example Response



```
{
  response: {
    action: "Deny Source"
    success: true
    message: "Source removed successfully."
    time: 10
  }
}

{
  response: {
    action: "Deny Source"
    success: true
    message: "Source set to unapproved, use config/source/delete to remove
it."
    time: 10
  }
}
```

Error Response



```


{
  response: {
    action: "Deny Source"
    success: false
    message: "Error denying source. You must be a community owner or
moderator to deny a source"
    time: 10
  }
}

{
  response: {
    action: "Deny Source"
    success: false
    message: "Source has been active, use config/source/delete to remove it"
    time: 10
  }
}

```

Config - Source - Delete

`/config/source/delete/{source-id-or-key}/{community-id}`

 Deletes the specified source and removes all documents harvested on behalf of that source. Note this may take a few minutes to run for sources with many documents.

Authentication

Required, see [Auth - Login](#). The caller must either be the source owner, or a community moderator/owner, or an administrator.

Arguments


source (required)
source ID or source key (see [Source data model](#)) to be deleted

community-id (required)
The community ID of the community containing the specified source, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

Example

<http://infinite.ikanow.com/api/config/source/delete/4acdb6eb4ed15e7baa395217/4cbdb9eb5ed98e7bdd489270>
<http://infinite.ikanow.com/api/config/source/delete/www.fema.gov.help.rss.shtm/4cbdb9eb5ed98e7bdd489270>

Example Response



```

{
  "response": {
    "action": "Delete Source",
    "success": true,
    "message": "Deleted source and all documents: <number of docs deleted>",
    "time": 120000
  }
}

```

Config - Source - Delete - Docs

`/config/source/delete/docs/{source-id-or-key}/{community-id}`

 Removes all documents harvested on behalf of a specified source. Note this may take a few minutes to run for sources with many documents. This call is normally made as part of ingest debugging (eg in conjunction with [source/test](#) and [source/save](#))

Authentication

Required, see [Auth - Login](#). The caller must either be the source owner, or a community moderator/owner, or an administrator.

Arguments

source (required)
source ID or source key (see [Source data model](#)) for which to delete all the documents.

community-id (required)
The community ID of the community containing the specified source, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

Example

<http://infinite.ikanow.com/api/config/source/delete/docs/4acdb6eb4ed15e7baa395217/4cbdb9eb5ed98e7bdd489270>
<http://infinite.ikanow.com/api/config/source/delete/docs/www.fema.gov.help.rss.shtm/4cbdb9eb5ed98e7bdd489270>

Example Response



```
{
  "response": {
    "action": "Delete Source",
    "success": true,
    "message": "Deleted source documents: <number of docs deleted>",
    "time": 120000
  }
}
```

Config - Source - Get

`/config/source/get/{source-id-or-key}`

 Retrieves a sources information.

Authentication

Required, see [Auth - Login](#)

Arguments

source (required)
source ID or source key (see [Source data model](#)) for which you want to see detailed info

Example

<http://infinite.ikanow.com/api/config/source/get/4cbdb9eb5ed98e7bdd489270>
<http://infinite.ikanow.com/api/config/source/get/www.fema.gov.help.rss.shtm>

Example Response



```

{
  "response": {
    "action": "Source Info",
    "success": true,
    "message": "Successfully retrieved source info",
    "time": 3
  },
  "data": {
    "_id": "4cbdb9eb5ed98e7bdd489270",
    "created": "Oct 19, 2010 11:31:55 AM",
    "modified": "Oct 19, 2010 11:31:55 AM",
    "url": "www.fema.gov/help/rss.shtm",
    "title": "FEMA: News Releases",
    "isPublic": true,
    "ownerId": "4e3706c48d26852237078005",
    "mediaType": "News",
    "key": "www.fema.gov.help.rss.shtm",
    "description": "Federal Emergency Management Agency Articles",
    "tags": [
      "response",
      "industry:emergency management",
      "topic:national disasters",
      "preparedness",
      "Disaster mitigation"
    ],
    "communityIds": [
      "4c927585d591d31d7b37097a"
    ],
    "harvest": {
      "harvested": "Mar 1, 2012 12:00:24 AM",
      "synced": "Mar 2, 2012 11:02:33 AM",
      "harvest_status": "error",
      "harvest_message": "Invalid document",
      "doccount": 0
    },
    "isApproved": true,
    "harvestBadSource": true,
    "extractType": "Feed"
  }
}

```

Config - Source - Good

/config/source/good/{communityId-list}



Returns a list of sources from the specified communities that are currently being harvested.

Authentication

Required, see [Auth - Login](#)

Arguments

communityId-list (required)

Community ID, or IDs (comma-separated), or **community ID - regex** for which to retrieve sources.

Example

<http://infinite.ikanow.com/api/config/source/good/4e258f8a64b46403c55095ce>

http://infinite.ikanow.com/api/config/source/good/*crime

Example Response

```
{
  "response": {
    "action": "Good Sources",
    "success": true,
    "message": "successfully returned good sources",
    "time": 14
  },
  "data": [
    {
      "_id": "4f22e491275e7a960af8cb7b",
      "created": "Apr 18, 2010 11:31:59 AM",
      "modified": "Apr 18, 2010 11:31:59 AM",
      "url": "http://eckington.wordpress.com/comments/feed/",
      "title": "Eckington: Way Better Than Spotsylvania (Comments)",
      "isPublic": true,
      "ownerId": "4e3706c48d26852237078005",
      "mediaType": "Blog",
      "key": "http.eckington.wordpress.com.comments.feeds",
      "description": "Eckington: Way Better Than Spotsylvania, Eckington
Community Blog (Comments)",
      "tags": [
        "dc",
        "washington",
        "district of columbia",
        "district",
        "blog",
        "eckington",
        "comments"
      ],
      "communityIds": [
        "4c927585d591d31d7c37097b"
      ],
      "harvest": {
        "harvested": "Mar 2, 2012 11:54:34 AM",
        "harvest_status": "success",
        "harvest_message":
"source=http://eckington.wordpress.com/comments/feed/ extracted=1 updated=0
deleted=0",
        "doccount": 17
      },
      "isApproved": true,
      "harvestBadSource": false,
      "extractType": "Feed"
    }
  ]
}
```

Config - Source - Pending

`/config/source/pending/{communityId-list}`

i Returns a list of sources waiting to be approved by the community owner.

Authentication

Required, see [Auth - Login](#)

Arguments

communityId-list (required)

Community ID, or IDs (comma-separated), or community ID - regex for which to retrieve sources.

Example

<http://infinite.ikanow.com/api/config/source/pending/4e258f8a64b46403c55095ce>

http://infinite.ikanow.com/api/config/source/pending/*

Example Response

i

```
{
  "response": {
    "action": "Pending Sources",
    "success": true,
    "message": "successfully returned pending sources",
    "time": 22
  },
  "data": [
    {
      "_id": "4cbdb9ed5ed98e7be9489270",
      "created": "Oct 19, 2010 11:31:57 AM",
      "modified": "Oct 19, 2010 11:31:57 AM",
      "url": "www.cdc.gov/nchs/Default.htm",
      "title": "Center for Disease Control and Prevention: Health
Information",
      "isPublic": true,
      "ownerId": "4e3706c48d26852237078005",
      "mediaType": "News",
      "key": "www.cdc.gov.nchs.default.htm",
      "description": "Health Information Source",
      "tags": [
        "Medical news",
        "medical",
        "topic:healthcare",
        "news",
        "industry:health care"
      ],
      "communityIds": [
        "4c927585d591d31d7b37097a"
      ],
      "harvest": {
        "harvested": "May 31, 2011 8:36:08 AM",
        "harvest_status": "error",
        "harvest_message": "Invalid XML: Error on line 323: The element
type \"li\" must be terminated by the matching end-tag \"</li>\".",
        "doccount": 0
      },
      "isApproved": false,
      "harvestBadSource": false,
      "extractType": "Feed"
    },
    {
      "_id": "4cbdb9ed5ed98e7bf0489270",
      "created": "Oct 19, 2010 11:31:57 AM",
```

```
"modified": "Oct 19, 2010 11:31:57 AM",
"url": "http://www2c.cdc.gov/podcasts/createrss.asp?t=r&c=175",
"title": "CDC: Environmental Medicine Case Studies",
"isPublic": true,
"ownerId": "4e3706c48d26852237078005",
"mediaType": "News",
"key": "http.www2c.cdc.gov.podcasts.createrss.asp.t=r.c=175",
"description": "Healthcare - Environmental Medicine articles",
"tags": [
  "environmental medicine",
  "environment",
  "topic:healthcare",
  "industry:healthcare",
  "healthcare"
],
"communityIds": [
  "4c927585d591d31d7b37097a"
],
"harvest": {
  "harvested": "May 31, 2011 8:36:17 AM",
  "harvest_status": "success",
  "doccount": 13
},
"isApproved": false,
"harvestBadSource": false,
"extractType": "Feed"
}
```

```
]
}
```

Config - Source - Save

`/config/source/save/{communityId}?json=<source-JSON>`

`/config/source/save/{communityId}` (POST ONLY)



Adds a new source, or updates an existing one, via GET or POST.

Note that it is recommended to **test sources** before "saving" (adding/updating) them.

Authentication

Required, see [Auth - Login](#)

Arguments

communityId (required)

The ID of the [community](#) with which you want to associate this source, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

json (required for GET methods, not supported for POST methods)

Source object in its JSON representation... see [this page](#) for an explanation of the source format, and [this set of pages](#) for an overview of the available source ingest functionality.

In the URL/posted JSON, note that:

- If the "key" or "_id" fields are specified, then the "save" is treated like an update, ie the source must already exist.
- The "communityIds" field is ignored, the above URL parameter is used instead.

Examples

Method.Get

`http://infinite.ikanow.com/api/config/source/save/4f4b9f8d4f36a94607bd0766?json={...}`

Method.Post

Example using curl:

```
curl -c login_cookie.txt -XPOST
'http://infinite.ikanow.com/api/config/source/save/4f4b9f8d4f36a94607bd0766' -d '{ ...
}'
```

Actionscript

See [this example](#).

Example Response



```
{
  response: {
    action: "Source"
    success: true
    message: "New source added successfully."
    time: 10
  }
}
```

```
{
  response: {
    action: "Source"
    success: true
    message: "New source added successfully. Note functionally identical sources
are also present within your communities, which may waste system resources."
    time: 10
  }
  data: {
    // source JSON, including the generated communityIds, key, and _id fields.
  }
}
```

In this case above, the exact same source (only differing by "display" parameters, owner etc) already exists. It is recommended to delete the source unless you have a specific reason not to.

```
{
  response: {
    action: "Source"
    success: true
    message: "Source updated successfully."
    time: 10
  }
}
```

Error Response



```
{
  response: {
    action: "Source"
    success: False
    message: "Unable to add new source."
    time: 10
  }
}
```


```
{
  response: {
    action: "Source"
    success: False
    message: "Unable to update source."
    time: 10
  }
}
```

Common error messages:

- **Unable to serialize Source JSON:** indicates that the JSON object POSTED (or passed via URL parameter) is invalid. Try using [JSON Lint](#) or a similar tool to debug.
- **The source ID is invalid:** For update requests, a source "_id" or "key" has been specified that does not match any in the database (this often happens with adding a source where the "_id" or "key" from a different system has been left in).
- **User does not have permissions to edit sources shared by this community:** You are not the owner of this source, nor a community moderator, not a system admin.
- **(curl/wget returns nothing:** Normally an indication that a POST has been used with no URL parameter, or a POST has been used with no content.)

Config - Source - Test

`/config/source/test?numReturn=<documents-to-return>&returnFullText=<1|0|true|false> (POST)`

 Returns documents harvested and enriched according to the POSTed [source object](#). This call can be used to test and debug source configuration prior to [saving](#) the object to be harvested (at which point it becomes harder both to debug and to fix problems, eg by [deleting documents](#)).

Authentication

Required, see [Auth - Login](#)

Arguments

numReturn (optional)

Number of processed documents to return (defaults to 10). (Note that this does not affect how many documents are harvested, just how many are enriched. Therefore it may still take a while when run on large directories/databases/fileshares with slow IO.)

returnFullText (required)

If "false" (default) or 0, does not return the full text of the document (to make the document easier to read etc). If "true" or "1" returns the populated "fullText" field (which is sometimes necessary, eg to debug the text extraction or cleansing configuration).

Examples

Method.Post

Example using curl:

```
curl -XPOST 'http://infinite.ikanow.com/api/config/source/test?numReturn=1' -d '{
  "json": {...} }'
```

Actionscript

See [this example](#).

Example Response



```
{
  response: {
    action: "Test Source"
    success: true
    message: "successfully returned 4 docs: source=<url> extracted=4 updated=0
deleted=0 urlerrors=0. <Warnings/non-fatal errors>"
    time: 10
  }
}
```

Error Response



```
{
  response: {
    action: "Test Source"
    success: true
    message: "successfully returned 0 docs: 4 file error(s).\n\n<List of errors>"
    time: 10
  }
}
```

```
{
  response: {
    action: "Test Source"
    success: false
    message: "Source error: <error message>"
    time: 10
  }
}
```

Common error messages:

- **Unable to serialize Source JSON:** indicates that the JSON object POSTED (or passed via URL parameter) is invalid. Try using [JSON Lint](#) or a similar tool to debug.
- **Source error:** A major problem occurred before enrichment started, eg authentication or path problems.
- **"successfully returned 0 docs <...>":** (0, or < the number desired) Also normally indicates an error occurred during harvesting but on a per document basis rather than for the entire source.
- **"[...] urlerrors=<more than 0>":** These errors have occurred during the enrichment process, eg the third party text/entity extractor failed, or there were errors in the structured/unstructured analysis handlers. The lines following this notification are error messages that should give some idea what went wrong.
 - (Note that errors can appear even if "urlerrors=0", eg non-fatal problems that caused an entity/association to be dropped).
- **(curl/wget returns nothing):** Normally an indication that a POST has been used with no URL parameter, or a POST has been used with no content. Alternatively if using a command-line json lint tool, it may indicate that there are characters not handled by the tool or it believes the JSON is invalid.)

Config - Source - User

`/config/source/user`



Returns a list of sources owned by a user. If the user is a system administrator `/config/source/user` returns a list of all sources that the user is a member of.

Authentication

Required, see Auth - Login

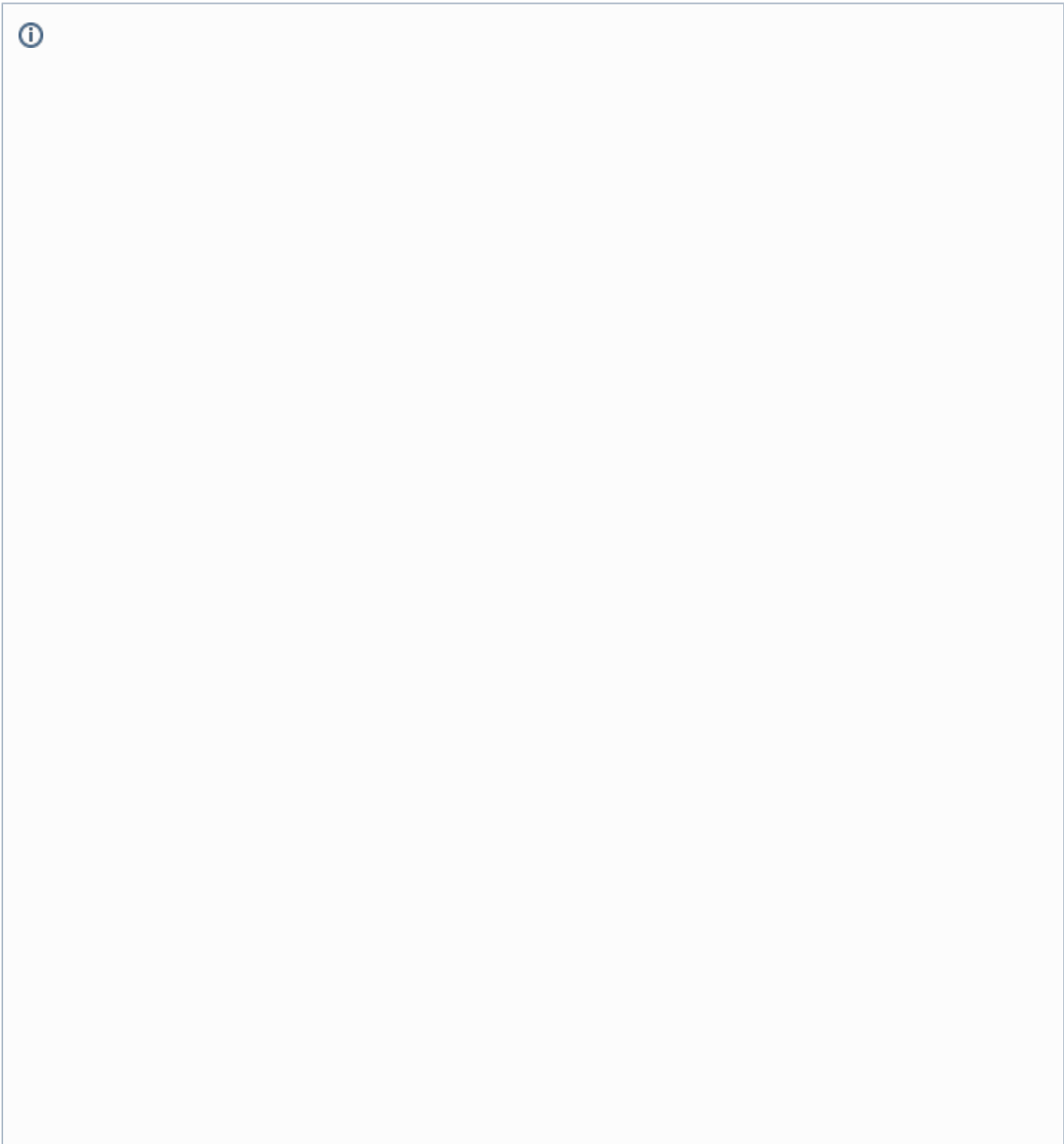
Arguments

none

Example

<http://infinite.ikanow.com/api/config/source/user>

Example Response



```

{
  "response": {
    "action": "User's Sources",
    "success": true,
    "message": "successfully returned user's sources",
    "time": 14
  },
  "data": [
    {
      "_id": "4f22e491275e7a960af8cb7b",
      "created": "Apr 18, 2010 11:31:59 AM",
      "modified": "Apr 18, 2010 11:31:59 AM",
      "url": "http://eckington.wordpress.com/comments/feed/",
      "title": "Eckington: Way Better Than Spotsylvania (Comments)",
      "isPublic": true,
      "ownerId": "4e3706c48d26852237078005",
      "mediaType": "Blog",
      "key": "http.eckington.wordpress.com.comments.feeds",
      "description": "Eckington: Way Better Than Spotsylvania, Eckington
Community Blog (Comments)",
      "tags": [
        "dc",
        "washington",
        "district of columbia",
        "district",
        "blog",
        "eckington",
        "comments"
      ],
      "communityIds": [
        "4c927585d591d31d7c37097b"
      ],
      "harvest": {
        "harvested": "Mar 2, 2012 11:54:34 AM",
        "harvest_status": "success",
        "harvest_message":
"source=http://eckington.wordpress.com/comments/feed/ extracted=1 updated=0
deleted=0",
        "doccount": 17
      },
      "isApproved": true,
      "harvestBadSource": false,
      "extractType": "Feed"
    }
  ]
}

```

Custom - Map Reduce

/custom/mapreduce/{inputcollection}/{map}/{reduce}/{query}



Runs a mongodb map reduce job immediately and only once.

Authentication

Required, see Auth - Login.

Arguments

inputcollection (required)

The input collection you want to run the map reduce job on. Can be DOC_METADATA for the documents collection or another custom map reduce results collection.

map (required)

The mongodb map to run.

reduce (required)

The mongodb reduce to run.

query (required)

The query to run on the input collection, the community ids will be added to whatever query gets submitted.

Example

```
http://infinite.ikanow.com/api/custom/mapreduce/DOC_METADATA/function(){ emit(this.source,{count:1})}/function(key,vals){var result = {count:0}; vals.forEach(function(val){result.count += val.count;}); return result;}/{mediaType:"Video"}
```

Example Response



```
{ "response": { "action": "Custom Map Reduce Run", "success": true, "message": "Map Reduce ran successfully", "time": 2413, "data": "[]" } }
```

Error Response



```
{ "response": { "action": "Custom Map Reduce Run", "success": false, "message": "error running map reduce", "time": 285 } }
```

Custom - Map Reduce - Get Jobs

/custom/mapreduce/getjobs

/custom/mapreduce/getjobs/{list-of-ids-or-titles}



Returns a list of jobs the current user has access to. Filtering can be performed by passing a comma-separated list of jobtitles or ids into the URL.

Authentication

Required, see Auth - Login.

Arguments

list-of-ids-or-titles (optional)

A comma-separated list of ids or titles to return (the list must contain either all ids or all titles, not a mixture of both)

Example

<http://infinite.ikanow.com/api/custom/mapreduce/getjobs>

<http://infinite.ikanow.com/api/custom/mapreduce/getjobs/twitterSentiment>

<http://infinite.ikanow.com/api/custom/mapreduce/getjobs/twitterSentiment,geoClustering>

<http://infinite.ikanow.com/api/custom/mapreduce/getjobs/4fe48dfe8c56e75b5fde868a,4f8d9a19f0c62dcba3281a7f>

cURL Example

Example Response



```
{
  "response": {
    "action": "Custom Map Reduce Get Jobs",
    "success": true,
    "message": "succesfully returned jobs",
    "time": 7
  },
  "data": [
    {
      "_id": "4fb245bfe4b0cc1eda8bd229",
      "jobtitle": "twitterSentiment",
      "jobdesc": "test",
      "submitterID": "4d88d0f1f9a624a4b0c8bd71",
      "communityIds": [
        "4e1f545315c1b0b2c4d4f4a3"
      ],
      "jarURL": "$infinite/share/get/4f8c7abde4b09c2c0a50189a",
      "inputCollection": "doc_metadata.metadata",
      "outputCollection": "4fb245bfe4b0cc1eda8bd229",
      "lastCompletionTime": "May 16, 2012 8:12:02 PM",
      "nextRunTime": 9223372036854776000,
      "jobidN": 0,
      "scheduleFreq": "NONE",
      "firstSchedule": "Dec 31, 1969 7:00:00 PM",
      "timesRan": 10,
      "isCustomTable": false,
      "mapper": "geographic.GenericGeographicSentiment$TokenizerMapper",
      "reducer": "geographic.GenericGeographicSentiment$IntSumReducer",
      "combiner": "geographic.GenericGeographicSentiment$IntSumReducer",
      "query": "{\"sourceKey\" :
\\\"twitter.com.statuses.public_timeline.atom\\\", \\\"docGeo\\\": {\\\"$ne\\\" : null}}",
      "outputKey": "org.apache.hadoop.io.Text",
      "outputValue": "org.apache.hadoop.io.IntWritable",
      "lastRunTime": "May 16, 2012 7:12:02 PM"
    }
  ]
}
```


Error Response



There aren't any, will just return an empty data section if user cannot access any jobs.

Custom - Map Reduce - Get Results

`/custom/mapreduce/getresults/{jobid_or_jobtitle}?limit=200`

 Returns the results of a map reduce job.

Authentication

Required, see [Auth - Login](#).

Arguments

jobid_or_jobtitle (required)

The id or title of a job you want the results of. You can find the id by writing it down when submitting a job or using [Get Jobs](#).

limit (optional)

An integer of how many results you want retrieved from the database, results will be grabbed in whatever order they have been stored in the db.

Example

<http://infinite.ikanow.com/api/custom/mapreduce/getresults/4e9c77ef17ef3523b657a890?limit=200>

<http://infinite.ikanow.com/api/custom/mapreduce/getresults/customjob1?limit=50>


cURL Example

Example Response



```
{ "response": { "action": "Custom Map Reduce Job
Results", "success": true, "message": "Map reduce job completed at:
123456789", "time": 98 }, "data": { "_id" : "a", "value" : 2 } }
```

Error Response




```
{ "response": { "action": "Custom Map Reduce Job
Results", "success": false, "message": "Job does not exist", "time": 98 } }
```

Custom - Map Reduce - Remove Job

`/custom/mapreduce/removejob/{jobidortitle}?removeJar=[true|1|false|0]`

 Removes a hadoop map reduce job and any data associated with it. This will not delete a jobs jar file if it is found in our share system.

 A job must not be running when attempting to remove it, an error will be reported if you do not wait until a job finishes.

A simple [web-based utility](#) is available for uploading JARs and managing jobs.

Authentication

Required, see [Auth - Login](#).

Arguments

jobidortitle (required)

The id or title of a job to be removed.

removeJar (optional)

If set to true or 1 will attempt to remove the jar after removing the job, will fail if the jar is used elsewhere or you aren't the owner. False by default, will not delete jar.

Example

<http://infinite.ikanow.com/api/custom/mapreduce/removejob/4e9c77ef17ef3523b657a890?removeJar=false>

cURL - Login, Get Map Reduce Jobs, Get Map Reduce Job Results, LogoutJava Example

Example Response



```
{ "response": { "action": "Remove MapReduce Job", "success": true, "message": "Job removed successfully" }
```

Error Response



```
{ "response": { "action": "Remove MapReduce Job", "success": false, "message": "Job is currently running (or not yet marked as completed). Please wait until the job completes to update it.", "time": 142 } }
```

Custom - Map Reduce - Schedule Job

`/custom/mapreduce/schedulejob/{jobtitle}/{jobdesc}/{communityIds}/{jarURL}/{timeToRun}/{frequencyToRun}/{mapperClass}/{reducerClass}/{combinerClass}/{query}/{inputcollection}/{outputKey}/{outputValue}`



Schedules a hadoop map reduce job. Returns the output collection id in the data field of the response if successfully queued to run.

A [detailed guide](#) to creating plugins.

A simple [web-based utility](#) is available for uploading JARs and managing jobs.

Authentication

Required, see [Auth - Login](#).

Arguments

jobtitle (required)

A descriptive name of the job being submitted.

jobdesc (required)

A description of what the job being submitted is attempting to do.

communityIds (required)

Community ID, or IDs (comma-separated), that the map reduce job wants to run on. These will be appended to the mongo query.

jarURL (required)

A URL to the location of the jar file to run for the job, this can be in our [Shares table](#) or hosted somewhere else on the web. Any permission errors will die silently and the job will not complete.

timeToRun (required)

The time you want a job to be run after in long form. For example if you want it to run immediately when possible you can submit 0. If you want the job to run after January 1, 2015 submit: 1420106400000.

frequencyToRun (required)

How often the job should be ran, either: NONE, HOURLY, DAILY, WEEKLY, MONTHLY. This will cause the job to get resubmitted after running, use NONE if you only want the job to run once.

mapperClass (required)

The java classpath to the jobs mapper, it should be in the form of package.file\$class

reducerClass (optional)

The java classpath to the jobs reducer, it should be in the form of package.file\$class

combinerClass (optional)

The java classpath to the jobs combiner, it should be in the form of package.file\$class (use the reducer if you have not written a combiner or submit null). If not present, then only the mapper (or combiner) is run, and records with duplicate keys will overwrite each other in an arbitrary order.

query (required)

The mongo query to use to get the jobs data. {} is a blank query or you can submit null. Also you can submit any post-processing you want by passing in an array of the form [{mongodb query},{postproc}]null, {fields} where postproc is a json object following the form:

```
{
  "limit":int,
  "sortField":"field.field.field",
  "sortDirection":-1|1,
  "limitAllData":true|false
}
```

and "fields" is the usual MongoDB "projection" object for specifying fields returned from queries.

UPDATE: the preferred method of specifying post-processing fields is now to use additional optional fields in the top-level of the query, as follows:

```
{
  // Output pre-processing
  "$output": { // (as above)
    "limit":int, // a record limit, how it is applied depends on limitAllData below
    "limitAllData":boolean, // if true, the above limit is applied to the entire
collection; if false, just to this iteration's records
    "sortField":string, // field to sort on (defaults to "_id"), supports dot notation
    "sortDirection":int, // -1 or 1 for descending or ascending
  },
  // Other control fields
  "$limit": int, // If specified then will process this number of records in one
giant split (used for debugging)
  "$fields": string, // Usual MongoDB specification of the fields to provide to the
mapper
  "$splits": int, // The maximum number of splits before the standard
MongoInputFormat class is used (which is very inefficient when a query is applied)
  "$docsPerSplit": int, // The maximum number of docs per split before the standard
MongoInputFormat class is used (which is very inefficient when a query is applied)
  //
  // The MongoDB query as before eg "sourceKey": "key.to.source", etc
  //
}
```

See the [Hadoop Plugin Guide](#) for more information.

inputcollection (required)

The mongo collection you want to use as input. You can submit DOC_METADATA to get the [documents metadata](#), DOC_CONTENT to get the [document contents](#), or grab a previous map reduce jobs results table in your communities by submitting its id or title (must be a member of that community).

outputKey (required)

The classpath for the map reduce output format key usually org.apache.hadoop.io.Text

outputValue (required)

The classpath for the map reduce output format value usually org.apache.hadoop.io.IntWritable

json (optional)

Current you can pass a json object in containing any custom arguments you want passed in to the map reduce job at runtime, these arguments will be available in the config file in your mapper/reducer
format: { "arguments": "any string you want here" }

Example

http://infinite.ikanow.com/api/custom/mapreduce/schedulejob/TestJob/Testing%20map%20reduce/4e9c77ef17ef3523b657a890/%24infinite%2Fshare%2Fget%2F4eafed58233558b98055c872/0/NONE/com.ikanow.infinite.core.mapreduce.examplejars.Test%24TokenizerMapper/com.ikanow.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/com.ikanow.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/null/doc_metadata/org.apache.hadoop.io.Text/org.apache.hadoop.io.IntWritable

cURL - Login, Get Map Reduce Jobs, Get Map Reduce Job Results, LogoutJava Example

Method.Post

Example using curl:

```
curl \-XPOST  
'http://infinite.ikanow.com/api/custom/mapreduce/schedulejob/TestJob/Testing%20map%20r  
educe/4e9c77ef17ef3523b657a890/%24infinite%2Fshare%2Fget%2F4eafed58233558b98055c872/0/  
NONE/com.ikanow.infinite.core.mapreduce.examplejars.Test%24TokenizerMapper/com.ikanow  
.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/com.ikanow.infinite.core.m  
apreduce.examplejars.Test%24IntSumReducer/null/doc_metadata/org.apache.hadoop.io.Text/  
org.apache.hadoop.io.IntWritable' \-d '{ "arguments": "Sterling Archer" }'
```

Example Response



```
{ "response": { "action": "Schedule MapReduce Job", "success": true, "message": "Job  
scheduled successfully, will run on: Wed Dec 31 19:00:00 EST  
1969", "time": 246 }, "data": "4f2007dd8196fe53a52c25a1" }
```

Error Response



```
{ "response": { "action": "Schedule MapReduce Job", "success": false, "message": "You  
are not allowed to use the given input collection.", "time": 142 } }
```

Custom - Map Reduce - Update Job

[/custom/mapreduce/updatejob/{jobidortitle}/{jobtitle}/{jobdesc}/{communityIds}/{jarURL}/{timeToRun}/{frequencyToRun}/{mapperClass}/{reducerClass}/{combinerClass}/{query}/{inputcollection}/{outputKey}/{outputValue}](#)



Updates a hadoop map reduce job. Returns the output collection id in the data field of the response if successfully queued to run. If you change the timeToRun a job can be rescheduled.



Return the word "null" for any of the update fields to not change a field, it will remain whatever the field was previously.

A detailed guide to creating plugins.

A simple web-based utility is available for uploading JARs and managing jobs.

Authentication

Required, see [Auth - Login](#).

Arguments

jobidortitle (required)

The id or the title of the job you want to update

jobtitle (required)

A descriptive name of the job being submitted.

jobdesc (required)

A description of what the job being submitted is attempting to do.

communityids (required)

Community ID, or IDs (comma-separated), that the map reduce job wants to run on. These will be appended to the mongo query.

jarURL (required)

A URL to the location of the jar file to run for the job, this can be in our [Shares table](#) or hosted somewhere else on the web. Any permission errors will die silently and the job will not complete.

timeToRun (required)

The time you want a job to be run after in long form. For example if you want it to run immediately when possible you can submit 0. If you want the job to run after January 1, 2015 submit: 1420106400000.

frequencyToRun (required)

How often the job should be ran, either: NONE, HOURLY, DAILY, WEEKLY, MONTHLY. This will cause the job to get resubmitted after running, use NONE if you only want the job to run once.

mapperClass (required)

The java classpath to the jobs mapper, it should be in the form of package.file\$class

reducerClass (required)

The java classpath to the jobs reducer, it should be in the form of package.file\$class

combinerClass (required)

The java classpath to the jobs combiner, it should be in the form of package.file\$class (use the reducer if you have not written a combiner or submit null).

query (required)

The mongo query to use to get the jobs data. {} is a blank query or you can submit null. See [Custom - Schedule Job](#) for more information about this query's format, or see the [Hadoop Plugin Guide](#) for more information.

inputcollection (required)

The mongo collection you want to use as input. You can submit DOC_METADATA to get the [documents metadata](#), DOC_CONTENT to get the [document contents](#), or grab a previous map reduce jobs results table in your communities by submitting its id or title (must be a member of that community).

outputKey (required)

The classpath for the map reduce output format key usually org.apache.hadoop.io.Text

outputValue (required)

The classpath for the map reduce output format value usually org.apache.hadoop.io.IntWritable

json (optional)

Current you can pass a json object in containing any custom arguments you want passed in to the map reduce job at runtime, these arguments will be available in the config file in your mapper/reducer

format: { "arguments": "any string you want here" }

Example

http://infinite.ikanow.com/api/custom/mapreduce/updatejob/4f2007dd8196fe53a52c25a1/TestJob/Testing%20map%20reduce/4e9c77ef17ef3523b657a890/%24infinite%2Fshare%2Fget%2F4eafed58233558b98055c872/0/NONE/com.ikanow.infinite.core.mapreduce.examplejars.Test%24TokerizerMapper/com.ikanow.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/com.ikanow.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/null/doc_metadata/org.apache.hadoop.io.Text/org.apache.hadoop.io.IntWritable - this example is the same as the schedule job, it would change a jobs fields to all these new settings (or the same settings)

<http://infinite.ikanow.com/api/custom/mapreduce/updatejob/4f2007dd8196fe53a52c25a1/null/null/null/0/null/null/null/null/null/null/null/null> - this example will just reschedule a job to run as soon as possible

[cURL - Login](#), [Get Map Reduce Jobs](#), [Get Map Reduce Job Results](#), [LogoutJava Example](#)

Method.Post

Example using curl:

```
curl \-XPOST
'http://infinite.ikanow.com/api/custom/mapreduce/updatejob/4f2007dd8196fe53a52c25a1/TestJob/Testing%20map%20reduce/4e9c77ef17ef3523b657a890/%24infinite%2Fshare%2Fget%2F4eafed58233558b98055c872/0/NONE/com.ikanow.infinite.core.mapreduce.examplejars.Test%24TokenizerMapper/com.ikanow.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/com.ikanow.infinite.core.mapreduce.examplejars.Test%24IntSumReducer/null/doc_metadata/org.apache.hadoop.io.Text/org.apache.hadoop.io.IntWritable' \-d '{ "arguments": "Sterling Archer" }'
```

Example Response



```
{"response":{"action":"Update MapReduce Job","success":true,"message":"Job updated successfully, will run on: Wed Dec 31 19:00:00 EST 1969","time":246},"data":"4f2007dd8196fe53a52c25a1"}
```

Error Response



```
{"response":{"action":"Update MapReduce Job","success":false,"message":"You are not allowed to use the given input collection.","time":142}}
```

Other Error Messages:

- **Must be owner/admin of job you are trying to update: You are not an admin or submitter of this job**
- **Bad job title/id: No jobs with this ID exist**
- **Bad parameters passed in for update or other general error: error scheduling job**
- **Bad parameter for frequencyToRun: No enum matching scheduled frequency, try NONE, DAILY, WEEKLY, MONTHLY**
- **Not a unique jobname: A job already matches that title, please choose another title**
- **Not allowed access to an input collection: You are not allowed to use the given input collection.**
- **Job is currently running: Job is currently running (or not yet marked as completed). Please wait until the job completes to update it.**

Custom - Saved Query - Schedule Job

`/custom/savedquery/schedulejob/{jobtitle}/{jobdesc}/{communitylds}/{timeToRun}/{frequencyToRun}/{query}/{inputcollection}/{outputKey}/{outputValue}`



Schedules a query job. Returns the output collection id in the data field of the response if successfully queued to run.

A simple [web-based utility](#) is available for scheduling jobs.

Authentication

Required, see [Auth - Login](#).

Arguments

jobtitle (required)

A descriptive name of the job being submitted.

jobdesc (required)

A description of what the job being submitted is attempting to do.

communityIds (required)

Community ID, or IDs (comma-separated), that the query job wants to run on. These will be appended to the mongo query.

timeToRun (required)

The time you want a job to be run after in long form. For example if you want it to run immediately when possible you can submit 0. If you want the job to run after January 1, 2015 submit: 1420106400000.

frequencyToRun (required)

How often the job should be ran, either: NONE, HOURLY, DAILY, WEEKLY, MONTHLY. This will cause the job to get resubmitted after running, use NONE if you only want the job to run once.

query (required)

The mongo query to use to get the jobs data. {} is a blank query or you can submit null. A typical place to get a query is by using the gui at infinite.ikanow.com and going to Advanced -> Save. Also you can submit any post-processing you want by passing in an array of the form [{mongodb query},{postproc}] where postproc is a json object following the form:

```
{
  "limit":int,
  "sortField":"field.field.field",
  "sortDirection":-1|1,
  "limitAllData":true|false
}
```

See the [Hadoop Plugin Guide](#) for more information.

inputcollection (required)

The mongo collection you want to use as input. DOC_METADATA is the only currently available option.

outputKey (required)

Currently set to "null" will be used in the future.

outputValue (required)

Currently set to "null" will be used in the future.

Example

http://infinite.ikanow.com/api/custom/savedquery/schedulejob/TestJob/Testing%20map%20reduce/4e9c77ef17ef3523b657a890/0/NONE/null/DOC_METADATA/null/null

Example Response

```
{"response":{"action":"Schedule MapReduce Job","success":true,"message":"Job
scheduled successfully, will run on: Wed Dec 31 19:00:00 EST
1969","time":246},"data":"4f2007dd8196fe53a52c25a1"}
```

Error Response

```
{"response":{"action":"Schedule MapReduce Job","success":false,"message":"You
are not allowed to use the given input collection.,"time":142}}
```

Custom - Saved Query - Update Job

[/custom/savedquery/updatejob/{jobidortitle}/{jobtitle}/{jobdesc}/{communityIds}/{timeToRun}/{frequencyToRun}/{query}/{inputcollection}/{outputKey}/{outputValue}](#)



Updates a saved query job. Returns the output collection id in the data field of the response if successfully queued to run. If you change the timeToRun a job can be rescheduled.



Return the word "null" for any of the update fields to not change a field, it will remain whatever the field was previously.

A simple web-based utility is available for updating jobs.

Authentication

Required, see [Auth - Login](#).

Arguments

jobidortitle (required)

The id or the title of the job you want to update

jobtitle (required)

A descriptive name of the job being submitted.

jobdesc (required)

A description of what the job being submitted is attempting to do.

communityids (required)

Community ID, or IDs (comma-separated), that the saved query job wants to run on. These will be appended to the mongo query.

timeToRun (required)

The time you want a job to be run after in long form. For example if you want it to run immediately when possible you can submit 0. If you want the job to run after January 1, 2015 submit: 1420106400000.

frequencyToRun (required)

How often the job should be ran, either: NONE, HOURLY, DAILY, WEEKLY, MONTHLY. This will cause the job to get resubmitted after running, use NONE if you only want the job to run once.

query (required)

The mongo query to use to get the jobs data. {} is a blank query or you can submit null. A typical place to get a query is by using the gui at [infinite.ikanow.com](#) and going to Advanced -> Save. Also you can submit any post-processing you want by passing in an array of the form [{mongodb query},{postproc}] where postproc is a json object following the form:

```
{
  "limit":int,
  "sortField":"field.field.field",
  "sortDirection":-1|1,
  "limitAllData":true|false
}
```

See the [Hadoop Plugin Guide](#) for more information.

inputcollection (required)

The mongo collection you want to use as input. DOC_METADATA is the only currently available option.

outputKey (required)

Currently set to "null" will be used in the future.

outputValue (required)

Currently set to "null" will be used in the future.

Example

http://infinite.ikanow.com/api/custom/savedquery/updatejob/4f2007dd8196fe53a52c25a1/TestJob/Testing%20map%20reduce/4e9c77ef17ef3523b657a890/0/NONE/null/DOC_METADATA/null/null - this example is the same as the schedule job, it would change a jobs fields to all these new settings (or the same settings)

<http://infinite.ikanow.com/api/custom/mapreduce/updatejob/4f2007dd8196fe53a52c25a1/null/null/null/0/null/null/null/null> - this example will just reschedule a job to run as soon as possible

Example Response



```
{"response":{"action":"Update MapReduce Job","success":true,"message":"Job updated successfully, will run on: Wed Dec 31 19:00:00 EST 1969","time":246},"data":"4f2007dd8196fe53a52c25a1"}
```

Error Response



```
{"response":{"action":"Update MapReduce Job","success":false,"message":"You are not allowed to use the given input collection.","time":142}}
```

Other Error Messages:

- **Must be owner/admin of job you are trying to update: You are not an admin or submitter of this job**
- **Bad job title/id: No jobs with this ID exist**
- **Bad parameters passed in for update or other general error: error scheduling job**
- **Bad parameter for frequencyToRun: No enum matching scheduled frequency, try NONE, DAILY, WEEKLY, MONTHLY**
- **Not a unique jobname: A job already matches that title, please choose another title**
- **Not allowed access to an input collection: You are not allowed to use the given input collection.**
- **Job is currently running: Job is currently running (or not yet marked as completed). Please wait until the job completes to update it.**

GUI Utilities - Plugin Manager

Overview



The Plugin Manager GUI is not currently compatible with IE. It is compatible with Chrome, Firefox, and Safari.

The plugin manager provides a simple web-based user interface for uploading new and updated map reduce plugins and saved queries to the system, and for sharing them across the different communities.

If you want to use a map reduce plugin for many jobs, use the [file uploader](#) to upload the jar once and then use this plugin Manager to schedule each job that uses the common jar.

Logging In

The plugin manager can be accessed from <ROOT_URL>/manager/pluginManager.jsp (eg <http://infinite.ikanow.com/manager/pluginManager.jsp>) . It can also be reached from the home page of the [Manager webapp](#) (itself linked from the main visualization GUI).

The following URL parameters are supported:

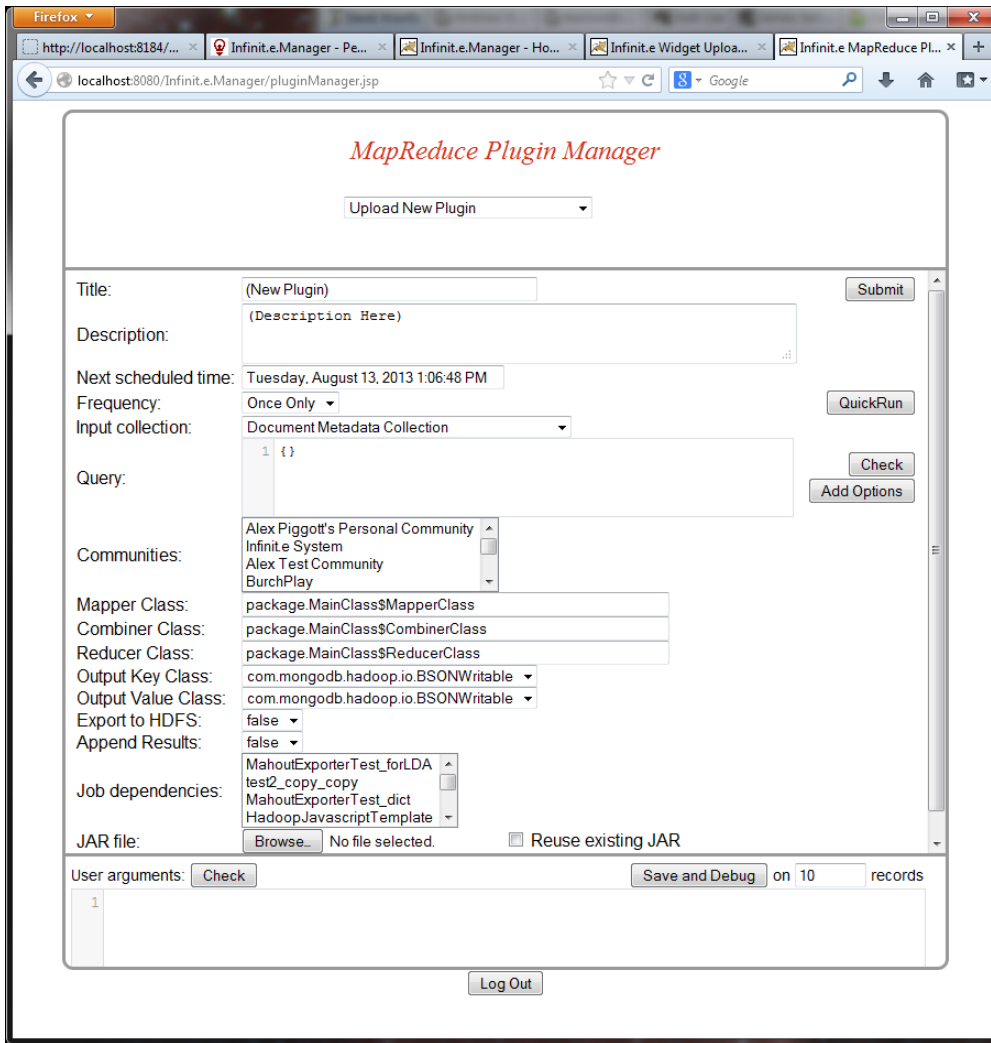
- Use **"?sudo"** to view jobs you don't own but which are shared with you (or all jobs if you are an admin), eg <http://infinite.ikanow.com/pluginManager.jsp?sudo>

This brings up username and password fields and a login button. (Unless already logged in, eg into the manager or main GUI - in which case skip to the next section).

Note that:

- The plugin manager shares its cookie with the main GUI, the file uploader, the source builder, and the person manager - logging into any of them will log into all of them.

Scheduling a new Map Reduce Plugin



The above figure shows the tool shortly after log-in.

To upload the plugin, simply fill in the fields, select the file from your hard drive and network drives, and select "Submit".

Note that you can select either a single or multiple (CTRL+click) communities. You can share plugins with any available community. If you upload to only your personal community, only you (and system administrators) will have access to the file.

The "query" field has a few noteworthy points:

- Some additional control fields can be supplied, as described [here](#).
 - The most common control fields can be specifically added with their default value by pressing the "Add Options" button
- The aforementioned query must be a MongoDB query (use the [document data model](#), or [content format](#)), Infinet.e queries are not currently supported (this functionality is coming).
- Press the "Check" button next to the "Query" field to validate the query JSON.

Note: If you set append results to false, there is no need to set an age out.

Note: If you don't want your job to depend on another jobs completion, do not select any job dependencies (you can CTRL-click to remove selected options if necessary).

Note: the "user arguments" field can be any string, it is interpreted by the code in the Hadoop JAR. For custom plugin developers: see this [tutorial](#) for a description of how to incorporate user arguments in the code (under advanced topics). Since the user arguments will normally be JSON or javascript (see info box below), a "Check" button has been provided that will validate either of those 2 formats.

i In particular, the built-in "HadoopJavascriptTemplate" template job uses the "user arguments" to hold the javascript code that gets executed in Hadoop.

Note: You can temporarily remove a combiner or reducer by putting "#" in front of it. Only the mapper is mandatory (others can be set to "none"), though normally at least the mapper and reducer are set.

Submit vs Quickrun

There are 2 options for submitting a job:

- Submit (button to the right of "Title")
- QuickRun (button to the right of "Frequency")

Submit will save the task (or update it if it already exists). If the frequency is not "Never" and the "Next Scheduled Time" is now or in the past, then job is immediately scheduled. The page refreshes immediately (unlike "QuickRun" below) and the progress can be monitored as described under "Following a job's progress" below.

"QuickRun" will set the frequency to "Once Only" and the time to "ASAP" (as soon as possible) and then will do 2 things:

- Submit as above
- It will wait for the job to complete before refreshing the page (all the "action" buttons are disabled in the meantime). You can't see the progress (see below) in the meantime, so this is best used on smaller jobs.

You can also debug tasks, this is described in the next section.

Debugging new and existing tasks

Just above "user arguments" there is a "Save and debug" button. This is very similar to pressing the "QuickRun" button described above, except:

- It will only run on the number of records specified in the text box next to the button.
- It will always run the Hadoop JAR in "local mode" (ie it won't be distributed to the Hadoop cluster, if one exists)
- Any log messages output by the Hadoop JAR (or in the javascript if running the prototype engine) are collected and output to the status message
 - (Note that if running in local mode, then "QuickRun" will log error messages - nothing will be currently logged in the typical cluster mode though, so the debug mode is necessary in this case - the alternative is running and testing in eclipse as described [here](#), which is quite involved)

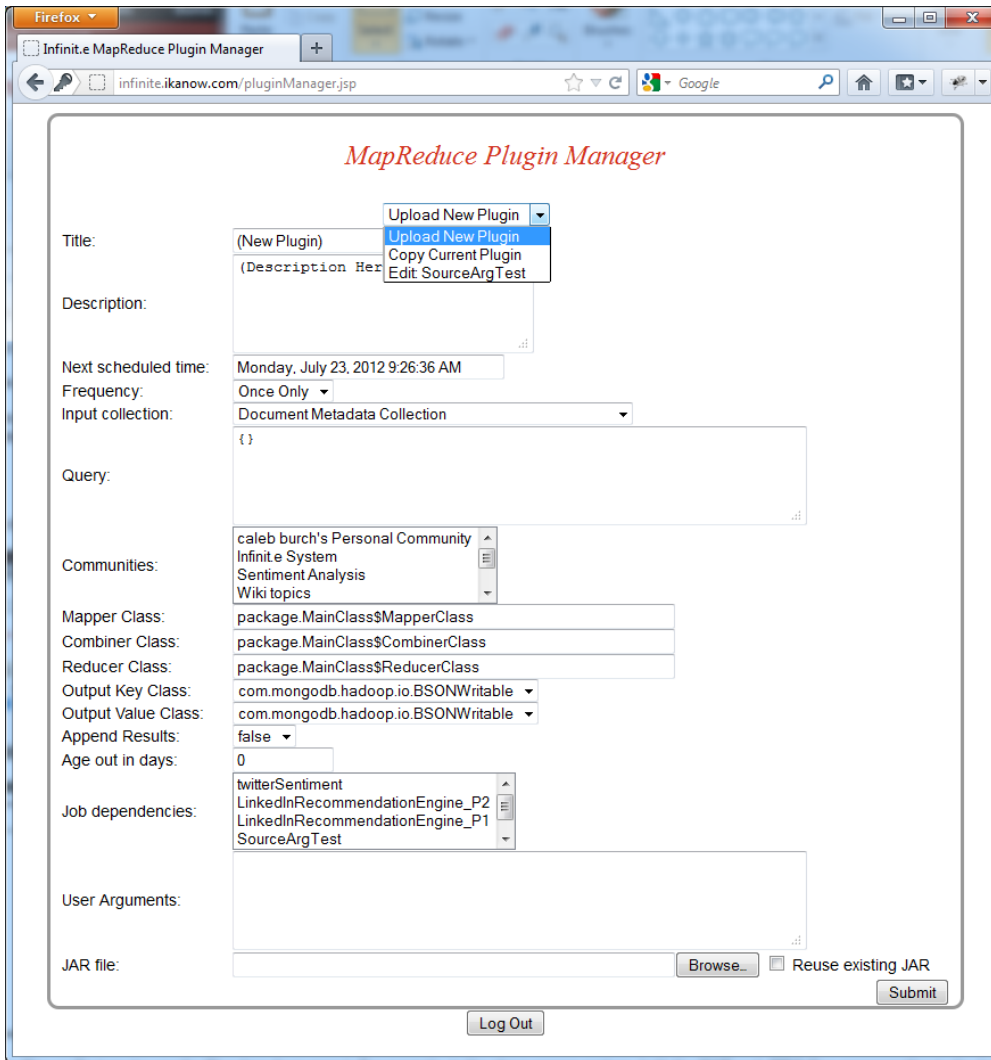
Scheduling a new Saved Query

If you want to schedule a saved query instead of a map reduce plugin you can follow the exact same process as scheduling a m/r job except for these few changes:

1. Leave JAR file blank
2. Leave Mapper, Combiner, and Reducer class blank
3. Submit a valid query for the [query](#) field, a good place to start is saving queries from the GUI under the Advanced -> Save option. These can be pasted into the query field. (Note unlike the normal map/reduce case, the query in this instance must be an Infinite query, not a MongoDB query).

Editing existing plugins/saved queries

After log-in, all plugins you own can be seen from the top drop-down menu (initially called "Upload New File"). If you are a community owner or moderator, all plugins in your communities can be seen in addition. If you an administrator, all plugins in the system can be seen.



Once a file has been chosen, it can be modified by changing the fields (and/or choosing a different file), and then selecting the "Submit" button.

Copying an existing custom task

Select the task to be copied from the top drop down menu, then select "Copy Current Plugin". You must change the title, edit other fields are required.

Deleting files

Log in and choose a file as above, then select the "Delete" button.

Running an existing job

If you want to start a job you can select for the option "Next scheduled time" to Once Only and it will be scheduled as soon as possible and only run once. If you want to schedule a job on a certain frequency you can adjust the frequency option to one of the other settings.

Following a job's progress

Once a job has been scheduled you will be able to track its progress by refreshing next to the run status, the current map and reduce completion status as well as any errors that may have occurred when running will be displayed in an informational header.

Run status:

Map Progress

Reduce Progress

Status Message:

Javascript Engine for Plugin Manager

- **Javascript Prototype Engine - basic operation**
- Javascript Prototype Engine - advanced
 - Accessing the Hadoop context object
 - Calling java from javascript
 - Date parsing
 - Logging
 - Converting JSON to strings
 - Aliasing
 - Security considerations
- Examples
 - Geo aggregation example
 - Temporal aggregation example

Javascript Prototype Engine - basic operation

The Infinite plugin manager comprises a built-in processing module, the "Javascript Prototype Engine". This module runs javascript entered into the "user arguments" text field.

To start off, select the "HadoopJavascriptTemplate", then "Copy Current Plugin" and change the title.

The basic format of the javascript is a standard map/combine/reduce template, and must be as follows:

```

function map(key, val) {
    var key = // write code to obtain a key JSON object (note: must be an object, not
just a string)
    var val = // write code to obtain a value JSON object
    emit( key, val ); // (can be called multiple times with different keys)
}

function combine(key, vals) {
    var aggregated_object = null;
    for (x in vals) {
        var val = vals[x];
        //combine val with aggregated_object
    }
    emit(key, val); // (can be called multiple times with different keys - normally just
called once with the same key though)
}

function reduce(key, vals) {
    var aggregated_object = null;
    for (x in vals) {
        var val = vals[x];
        //combine val with aggregated_object
    }
    emit(key, val); // (can be called just once, the key must match the input)
}

```

The **map function** takes the following parameters:

- **key** is the `_id` of the object described below
- **val** is an object from the input collection (eg a [document](#) if from "document metadata", an [entity](#) if from "aggregated entity", an [association](#) if from "aggregated association" etc)

The (optional) **combine function** takes the following parameters:

- **key**: the key emitted from the map functions
- **vals**: an unordered list of "val" object emitted from the map functions with the same key

The (optional) **reduce function** takes the following parameters:

- **key**: the key emitted from the map/combine functions
- **vals**: an unordered list of "val" object emitted from the map/combine functions with the same key

This code is distributed using Hadoop in a standard way, ie as follows:

- Based on the query and input collection a set of mappers are distributed across the Hadoop cluster, with each mapper being assigned a slice of the records matching the query
- The **map** function is invoked for each record, the **val** objects generated and output using the **emit** function are grouped by **key**.
- At the end of the map, the **combine** function is called on the (**key**, group-of-**val** objects) pairs ... the output from the **emit** function invocations are sent to the available reducers (normally just one)
- The reduce function is invoked for each received (**key**, group-of-**val** objects) pairs, the output from **emit** is written to the output collection.

The combine and reduce functions are normally the same, allowing the following shortcut:

```

// delete the combine function block and...
combine = reduce;

```



Note that the combine function is normally present for performance and can be discarded without loss of functionality. If the reduce function is ignored then the output of the map is written directly to the output collection (/HDFS directory) - if multiple objects with the same key are emitted from the same map function, only one is preserved. Removing either of the combine/reduce functions must be accompanied by commenting out the corresponding Combiner/Reducer class in the [plugin manager GUI](#).

Javascript Prototype Engine - advanced

Accessing the Hadoop context object

This is not currently possible.

Calling java from javascript

Note that many java functions can be called from within javascript to extend the limited built-in functionality. Examples are provided below. The following relevant JARs are available:

- [The Infini.t.e data model library](#).
- commons-logging, commons-httpclient

Date parsing

Due to the slightly non-standard date storage in MongoDB, together with the limitations of the built-in Date function in Rhino (the javascript engine built into Java), Infini.t.e/MongoDB formatted dates must be parsed in a slightly convoluted way:

```
var date = new
java.text.SimpleDateFormat('yyyy-MM-dd').parse(val.publishedDate["$date"]);
// (or val.created or val.modified, or val.associations[x].time_start, etc)
```

Logging

Messages can be logged and are stored in the status message as follows (note unless run in debug mode, only ERROR category messages are retrieved).

```
// Put this outside the map/combine/reduce functions, ie in global space
var logger = org.apache.log4j.Logger.getLogger("path.to.package"); // eg
"com.ikanow.infini.t.e.utility.hadoop.HadoopPrototypingTool"

//Within the map/combine/reduce functions it can be called like:
logger.info("Info message!"); // (discarded unless in debug mode)
logger.error("Error message!!"); // (retrieved in local or debug mode)
```

Note that even error messages cannot currently be retrieved on full Hadoop clusters - debug mode must be used in that case even for ERROR messages.

Note also that logged messages are written to the internal Hadoop log files - it is therefore recommended to have a debug flag and turn them off for operational deployments, eg:

```
// Put this outside the map/combine/reduce functions, ie in global space
var logger = org.apache.log4j.Logger.getLogger("path.to.package"); // eg
"com.ikanow.infini.t.e.utility.hadoop.HadoopPrototypingTool"
var debug = false;

//Within the map/combine/reduce functions it can be called like:
if (debug)
    logger.info("Info message!"); // (discarded unless in debug mode)
```

Converting JSON to strings

For logging, it is often useful to output an entire JSON object as a string. Unfortunately the standard "JSON.stringify" function is not available in the version of Rhino currently used. Therefore this is in general not currently possible. The "val" parameter's string representation can be accessed as the globals "_map_input_value", "_combine_input_values" or "_reduce_input_values" (and the string representations of the key objects in the mapper/reducer/combiner can be accessed via "_map_input_key", "_combine_input_key" and "_reduce_input_key").

Aliasing

Currently document/entity objects accessed are **not aliased**. (Since the data model JAR is available, in theory the [driver's](#) getAliases call is available; in practice this would be fair bit of work to integrate).

It is on the roadmap to support aliasing within the Plugin framework, at which point it will be available in the javascript engine also.

Security considerations

Note that there is currently no security layer within the javascript engine (or in the Hadoop engine in general) - therefore internal files and ports can be accessed freely by the injected code. Where strict access controls are necessary, the plugin functionality should be disabled.

Examples

Geo aggregation example

The following is a simple example of using the Hadoop JS interface to aggregate the sentiment associated with geo-tagged documents:

```
function map(key, val) {
  var label_lat = Math.round(val.docGeo.lat/10)*10;
  var label_lon = Math.round(val.docGeo.lon/10)*10;
  var label = label_lat.toString()+ ':' + label_lon.toString();
  for (ent_i in val.entities) {
    var ent = val.entities[ent_i];
    if ((null != ent.sentiment) && (ent.type == "Keyword")) {
      emit({label: label, label_lat: label_lat , label_lon: label_lon }, {sentiment:
ent.sentiment, count: 1});
    }
  }
}
function reduce(key, vals) {
  var retval = { sentiment: 0.0, count: 0 };
  for (x in vals) {
    retval.sentiment += parseFloat(vals[x].sentiment);
    retval.count += parseInt(vals[x].count);
  }
  retval.geo = {};
  retval.geo.lat = key.label_lat;
  retval.geo.lon = key.label_lon;

  emit(key,retval);
}
combine = reduce;
```

Temporal aggregation example

The following is a simple example of using the Hadoop JS interface to aggregate the sentiment across time:

```

function map(key, val) {
  // Get sentiment:
  var plusSentiment = 0.0;
  var negSentiment = 0.0;
  var sentimentCnt = 0;
  if (null != val.entities) {
    for (x in val.entities) {
      var entity = val.entities[x];
      if (null != entity.sentiment) {
        if (entity.sentiment > 0) {
          sentimentCnt++;
          plusSentiment += entity.sentiment;
        }
        if (entity.sentiment < 0) {
          sentimentCnt++;
          negSentiment += entity.sentiment;
        }
      }
    }
  }
  if (sentimentCnt > 0) {
    var date = new
    java.text.SimpleDateFormat('yyyy-MM-dd').parse(val.publishedDate["$date"]);
    emit( { 'day': date.getTime(), { plus: plusSentiment, minus: negSentiment, cnt:
    sentimentCnt } } );
  }
}

function reduce(key, vals) {
  var initVal = { plus: 0.0, minus: 0.0, cnt: 0, net: 0.0, netavg: 0.0 };
  for (x in vals) {
    initVal.plus += vals[x].plus;
    initVal.minus += vals[x].minus;
    initVal.cnt += vals[x].cnt;
  }
  initVal.net = initVal.plus + initVal.minus;
  initVal.netavg = initVal.net/initVal.cnt;
  initVal.date = new Date(key.day).toString();
  emit( key, initVal );
}
combine = reduce;

```

GUI utilities - uploading files

Overview

The file uploader provides a simple web-based user interface for uploading new files to the system, and for sharing them across the different communities.

Currently the main uses for file sharing in Infini.e are:

- Uploading widgets to the system (this is actually better done via the [widget uploader](#), which performs a number of uploads in a single step)
- Uploading JARs to the system to be used as [MapReduce plugins](#).
- Storing [sources](#) before they are published to the system (this is all managed by the source builder GUI tool).

But the [share interface](#) is completely generic, and [widgets can be developed](#) that allow communities of users to share app-specific binary and text files between them. In addition, roadmap items to both the core platform and visualization framework include query sharing and per-user/community synonym handling that will use the share interface.

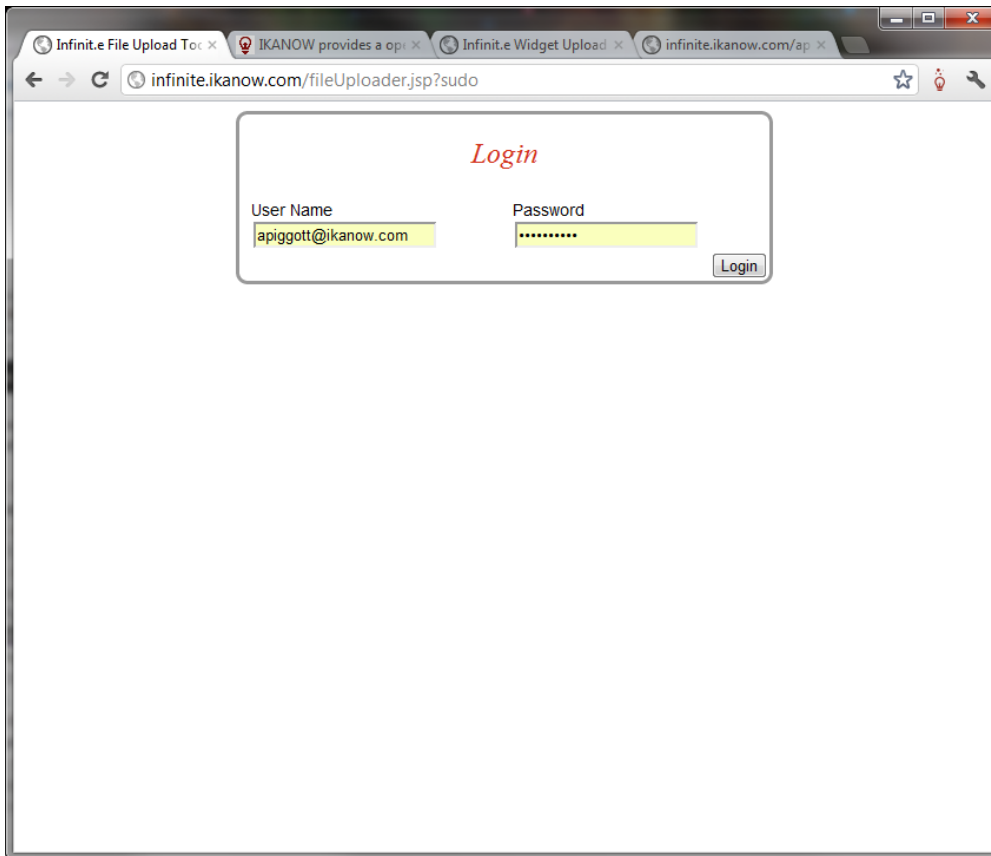
Logging In

The widget uploader can be accessed from <ROOT_URL>/manager/fileUploader.jsp (eg <http://infinite.ikanow.com/manager/fileUploader.jsp>). It can also be reached from the home page of the [Manager webapp](#) (itself linked from the main visualization GUI).

The following URL parameters are supported:

- In order to login with admin privileges (for admins only!), append "**?sudo**" or "**?sudo=true**" to the URL (eg <http://infinite.ikanow.com/manager/fileUploader.jsp?sudo>).
 - *If you are not logged-in with admin privileges you will only be able to edit your own files, though you will still be able to share files with any communities of which you are a member - this is different to both source management and widget uploading, where you must be a community moderator/owner.*
- You can filter to only certain types of file, using "**?ext=<type>**", where <type> is either "jar" or a **MIME type** (eg "pdf" for "application/pdf"). The GUI also provides a drop down for performing this from the set of available source or MIME types, so it should not normally be necessary to filter manually this way.
 - *Practical examples of filtering are described below.*

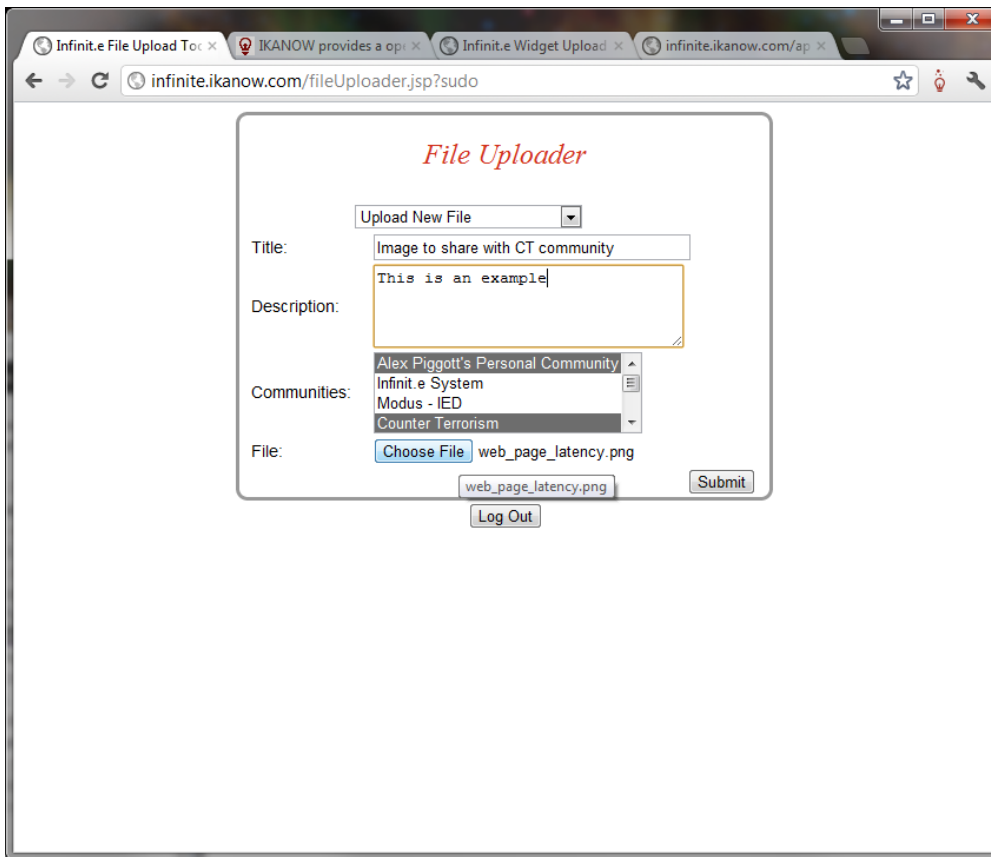
This brings up username and password fields and a login button. (Unless already logged in, eg into the manager or main GUI - in which case skip to the next section).



Note that:

- The file uploader shares its cookie with the main GUI, the widget uploader, the source builder, and the person manager - logging into any of them will log into all of them.

Uploading a new "binary" file



(Note "binary" here refers to anything except for JSON)

The above figure shows the tool shortly after log-in.

To upload the file, simply fill in the fields, select the file from your hard drive and network drives, and select "Submit". Note that at present all uploaded files are treated as **binary** not **JSON**.

Note that you can select either a single or multiple (CTRL+click) communities. You can share files with any available community. If you upload to only your personal community, only you (and system administrators) will have access to the file.

The system will automatically detect the MIME type and store this - this will be used both in filtering (see below) and for the content-type for when that file is [served from the REST API](#).

Uploading a new "JSON" file

Infinit.e handles JSON with some additional support. The process is the same as above, except:

- Select "Upload New JSON" (instead of "Upload New File")
- There is an additional field "type" that can be any string value. This type is used in a few different ways:
 - It can be filtered on from the "Filter On" drop down menu in this GUI
 - It can be used as an efficient search object in the [Social - Share - Search API call](#).
 - Shares of type "infinite-entity-alias" are used by the framework for aliasing.
 - Shares of type "source_template" are used by the [Source Editor GUI](#) as shareable templates.
 - Shares of type "widgetsave" are used the main GUI to store per-user or per-community settings
 - *(And of course any other Infinit.e application can use it for internal purposes)*
- The JSON is checked on ingest and an error thrown if it is invalid.

Sharing an existing Infinit.e (JSON) object

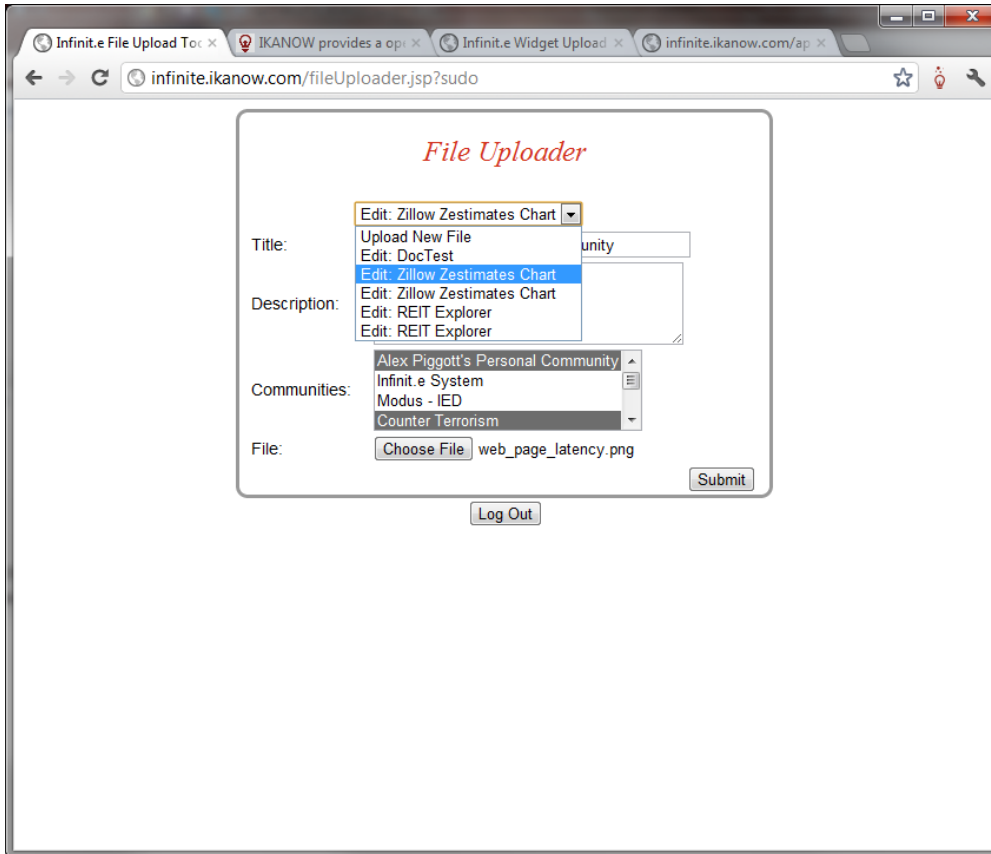
JSON objects from various internal Infinit.e databases can be shared to communities by selecting "Share existing object". This generates the following fields:

- The "type" field, which is treated identically to JSON shares as described above (note this makes sense because shared objects are always JSON)
- A "reference location" dropdown which allows the user to choose from the available databases:
 - Custom Plugin Collection: the first object from this custom job is shared
 - (eg the "_id" of the job can be see next to the "Title" field of the [plugin manager](#), or the [Custom - Get Jobs API call](#))
 - Document Metadata Collection: the [document](#) with the specified "_id" field is shared

- Aggregated Entity Collection: the **entity object** with the specified "_id" field is shared
- Aggregated Association Collection: the **association object** with the specified "_id" field is shared
- A "reference doc id" field that must contain the "_id" field of the object chosen above.
- (By contrast the "File" field is removed compared to JSON/binary shares)

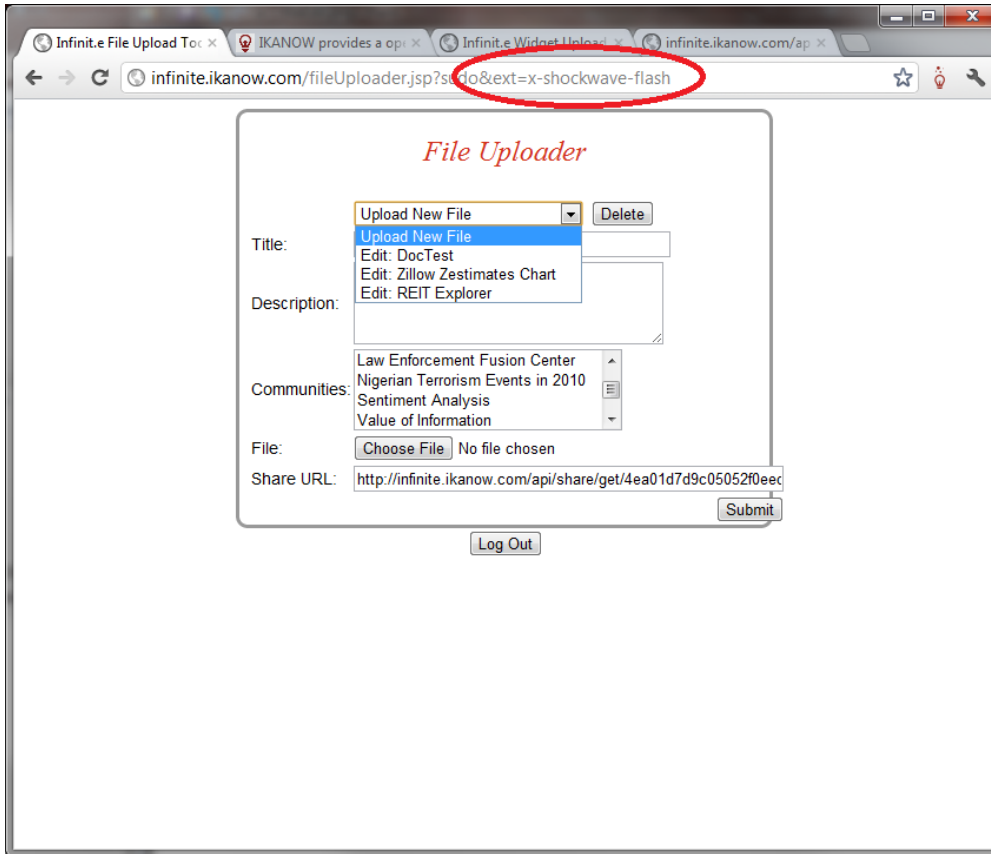
Editing existing shares

After log-in, all files you own can be seen from the top drop-down menu (initially called "Upload New File"). If you are a community owner or moderator, all files in your communities can be seen in addition. If you an administrator, all files in the system can be seen.



The "**?ext=<type>**" URL parameter can be used to filter on file type (or use the filter drop down menu). For example, in the above screen capture, both the SWF and JPEG files corresponding to the listed widgets are shown.

Compare with the following screen capture:



Here we have filtered to only show files with MIME type "x-shockwave-application", and only the SWFs are shown. Conversely, using "ext=gif" would have restricted the drop down to show only the icon files.

Currently, the most common usage will be "ext=jar" ("jar" is a special string defined for this tool), to view Hadoop plugins.

Once a file has been chosen, it can be modified by changing the fields (and/or choosing a different file), and then selecting the "Submit" button.

At present there are some restrictions on how an existing file can be updated:

- If a new file is uploaded, the entirety of the share can be modified (title, description, communities, and of course file).
- If a new file is not uploaded, ie "No file chosen" is displayed next to file, only the communities can be updated.
- *(This restriction will be removed in a future version of the tool).*

i Note that there is currently no way of explicitly [endorsing/unendorsing](#) shares from this GUI (we will add that functionality if it becomes in demand), but in the meantime you can work around that by unsharing it from the desired community, and then sharing it back again (assuming you are an administrator or community moderator).

Deleting files

Log in and choose a file as above, then select the "Delete" button.

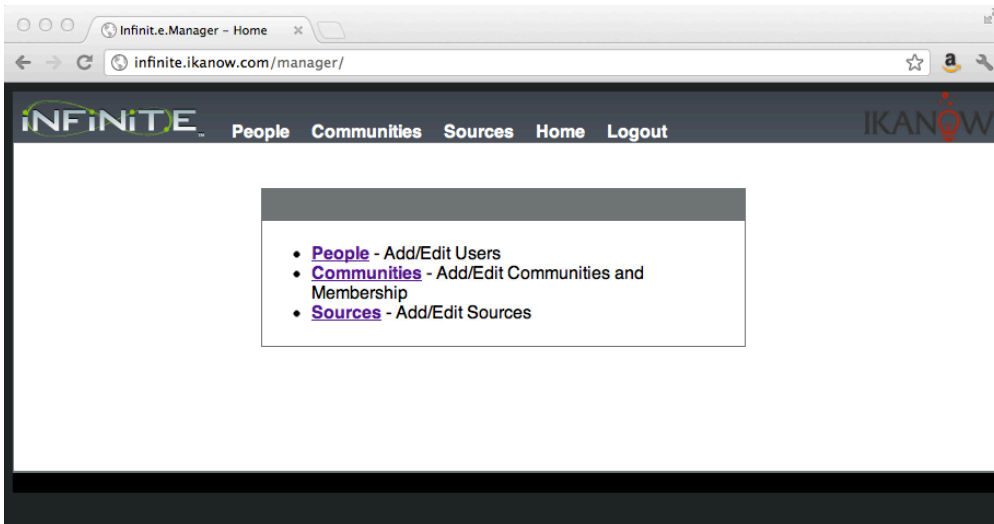
Infinite.Manager

Introduction

The Infinite.Manager application is a web based interface that allows authorized system users to manage:

- [People](#)
- [Communities](#)
- [Sources](#)
 - [Including a Chrome extension for quickly adding web pages and RSS feeds](#)
- [Plugins](#)
- [Widgets](#)

- Shared files

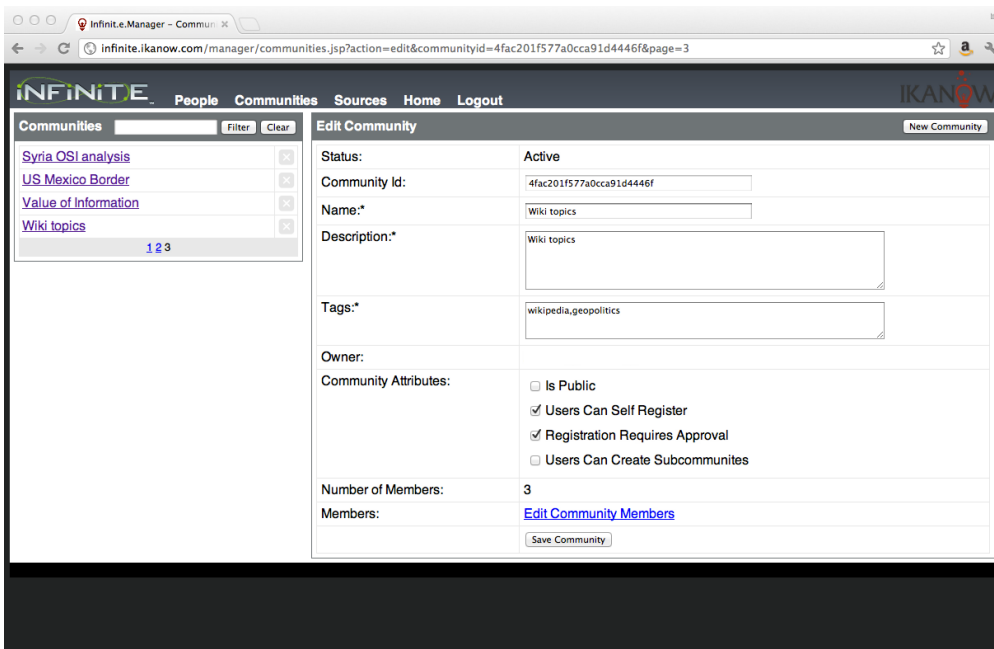


On default installations of the Infit.e platform Infit.e.Manager is accesible via web browser at: <ROOT_URL>/manager/.

Infit.e.Manager - Communities

Overview

The Infit.e.Manager Communities page allows users with the appropriate permissions to add, edit and delete communities from the system.



Note: The functionality found on the Communities page is restricted based on account type. System administrators have full access to add, edit, or delete communities while community owners and moderators have permissions to manage those communities that they have edit permissions for.

Add Community

To add a new community:

1. Click on the New Community button;
2. Fill in the required Add New Community form fields: Name, Description and Tags (a comma delimited list), and the properties (see below);
3. Click on the Create Community button.

Parameters:

- Parent Id: allows hierarchical relationships to be set up between communities. Parent/child relationships are used internally as described [here](#), in addition the information can be utilized by applications for whatever domain-specific logic is desired.
- "is Public": if true, then non-members can see this community
- "Users can self register": if true, then non-members can request joining this community (if visible); if false, then users can only be invited from within the community
- "Registration requires approval": if true, then non-members can add themselves to public communities without approval from the community owner/moderators
- "Users can create sub-communities": if false, then members without moderator/owner rights can create child communities.
- "Members are visible to other members": if false, then members without moderator/owner rights will not be able to see other members (the members field will be removed from returned communities).

The system will create the new community and refresh the page to update the form with the details of the newly created community.

Edit Community

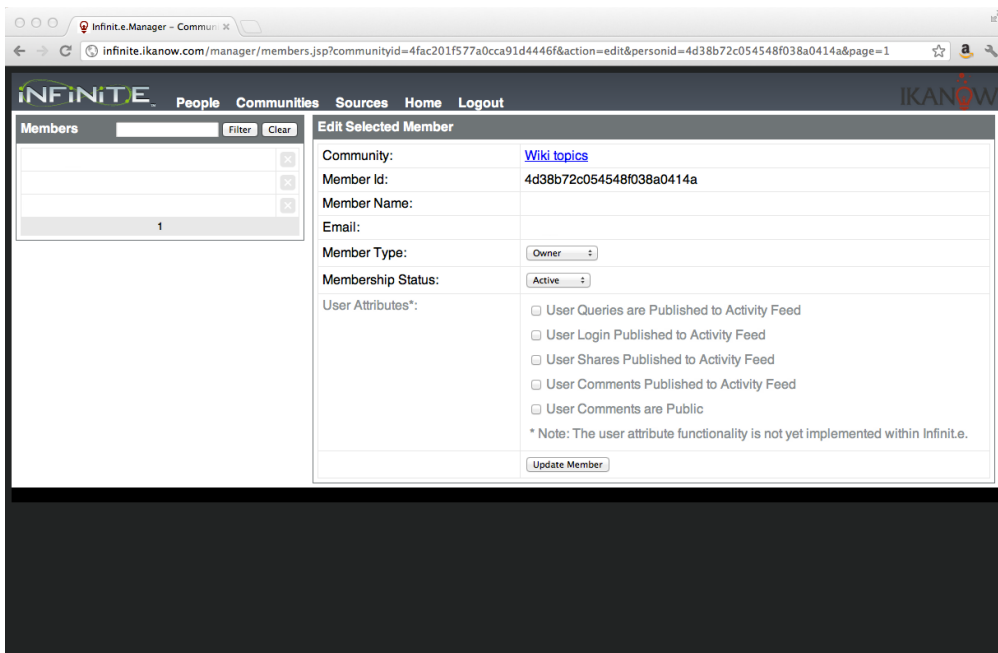
To edit a community simply click on the community's name in the in the Communities list to populate the Edit Community form. The same fields as above can be edited.

Community members can be edited by clicking on the Edit Community Members link, if visible. Community owners and moderators and system administrators can add new members from the Add Community Members link.

Edit Community Members

Clicking in the Edit Community Members link takes you to the Community Members page which allows you to edit or delete members for a given community. To edit a member simply click on their name in the Member's list (note that the names in the screenshot have been whited out for privacy reasons) and the system will display their details in the Edit Selected Member form. Currently there are only two values that can be edited:

- Member Type - Owner, Moderator, Content_Publisher, Member
- Membership Status - Active, Pending, Disabled



Members can be promoted to moderators or owners of a group, or demoted to be just a member via the Member Type drop down list. It is important to note that there can be only one community owner at any given time. If a member is promoted to owner then the current owner is demoted to moderator.

Community moderators have all the permissions of owners, except the ability to delete the community or change owner. "Content Publishers" can create sources without admin/owner/moderator approval (but otherwise have no more access than members).

New members can be approved (if a community requires registration and approval) by selecting the Active status from the Membership Status drop down list. To remove a user's ability to see the contents of a community you can select the Disabled option from the Status drop down list or you can more permanently remove the user by clicking on the "X" button next to their display name in the Members list.

Note: The User Attributes functionality is not current implemented in Infnite.e.

Add Community Members

This generates a page that shows all non-members that are visible by the user. The users can be selected by checking the checkbox next to their

name on the left, and pressing the "Add Selected People" button will add them to the community as members.

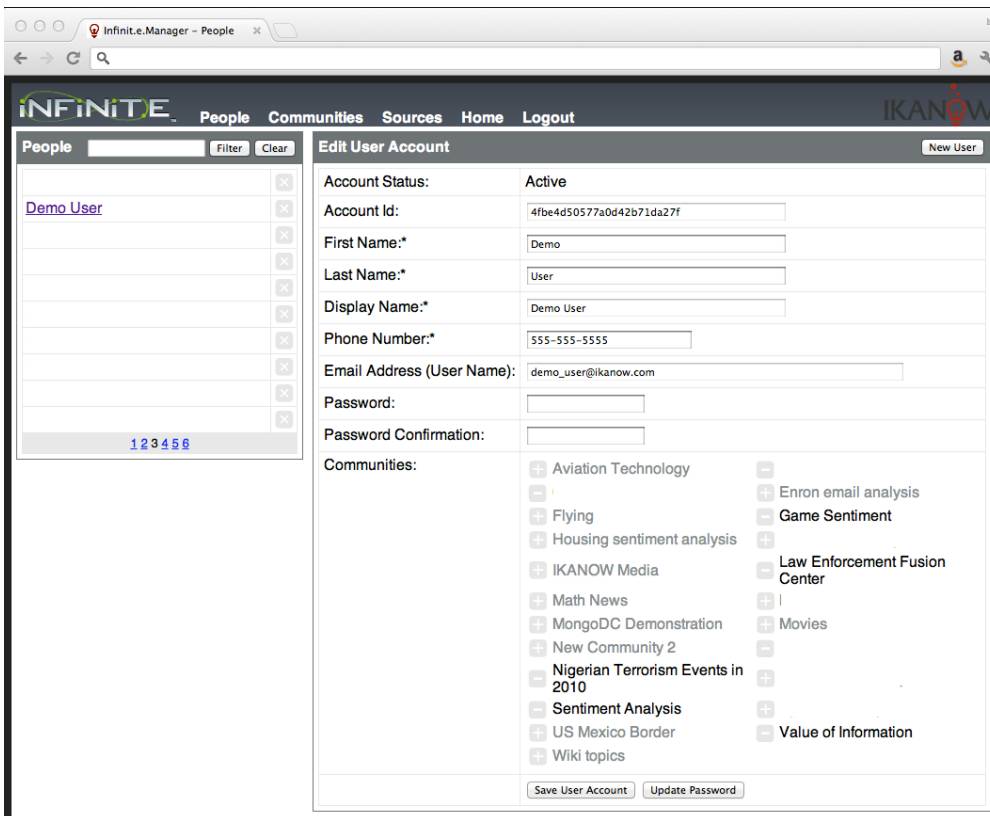
Delete Community

To delete a community click on the "X" button next to the community's name in the Communities list. Please note that the deletion process requires two steps to complete. The first time you click on the "X" button and confirm that you wish to delete the community the system removes all of the users from the community and suspends the community. To delete all shares, sources, and documents within a community you must click on the "X" button a second time. (See the [Social - Community - Remove](#) documentation for more information.)

Infini.e.Manager - People

Overview

The [Infini.e.Manager](#) People page provides a simple web-based user interface for adding new users, editing user's profiles (including passwords and which communities they belong to), and delete old users.



Note: The functionality found on the People page is restricted based on account type. System administrators have full access to add, edit, or delete users while standard users will only have access to edit their own data.

Add a New User

Click on the **New User** button in the upper right hand corner and fill out the Add New User Account form.

The following fields are required:

- First Name
- Last Name
- Account Type (admins only): defaults to "User", admins can specify a user to be another "Admin". Note this field is "write only", ie a user's current status is hidden.
 - From August 2013 there is a new account type: "admin-on-request". This is described below.
- Phone
- Email Address: The email address serves as the user name and must be unique for each user.
- Password
- API key: If specified, [Infini.e REST calls](#) can be authenticated with the URL parameter "infinite_api_key" rather than by logging in. Note that the first time an API key is added, it is not applied until a [manual login](#) occurs.

Editing a User

To edit a user click on their Hyperlinked name in the People list on the left hand side of the People page. Note: The list of accounts can be filtered by name using the text box at the top of the list and the Filter button.

When editing a user account there are three primary actions that you can perform:

- Update all user account fields (with the exception of the list of communities)
When updating a user's account information the following fields are editable: First name, last name, phone number and password. You cannot manually edit the display name field and you cannot change the email address since the email address serves as the user name. To save the changes click on the Save User Account button.
- Update the user password only
To update the user password simply type a new password into the Password and Password Confirmation text boxes and click on the Update Password button.
- Add or remove the user from communities
Underneath the user Password text boxes is a list of communities that the user can join or is already a member of. To change the user's membership status for a community simply click on the "+" or "-" button.



A user will only be able to see other users that share a community other than the system with the current logged in user, unless the logged in user is an admin.

Deleting a User

Delete a user is as simple as clicking on the "X" button to the right of the user's display name in the People list. The application will ask you to confirm that you wish to delete the account ("Do you really wish to delete the user account for: XXXXX") however it is important to note that you **cannot undo a delete**.

Users can be deleted in bulk by selecting the checkboxes next to users to be deleted and then pressing the "Deleted selected people" button.

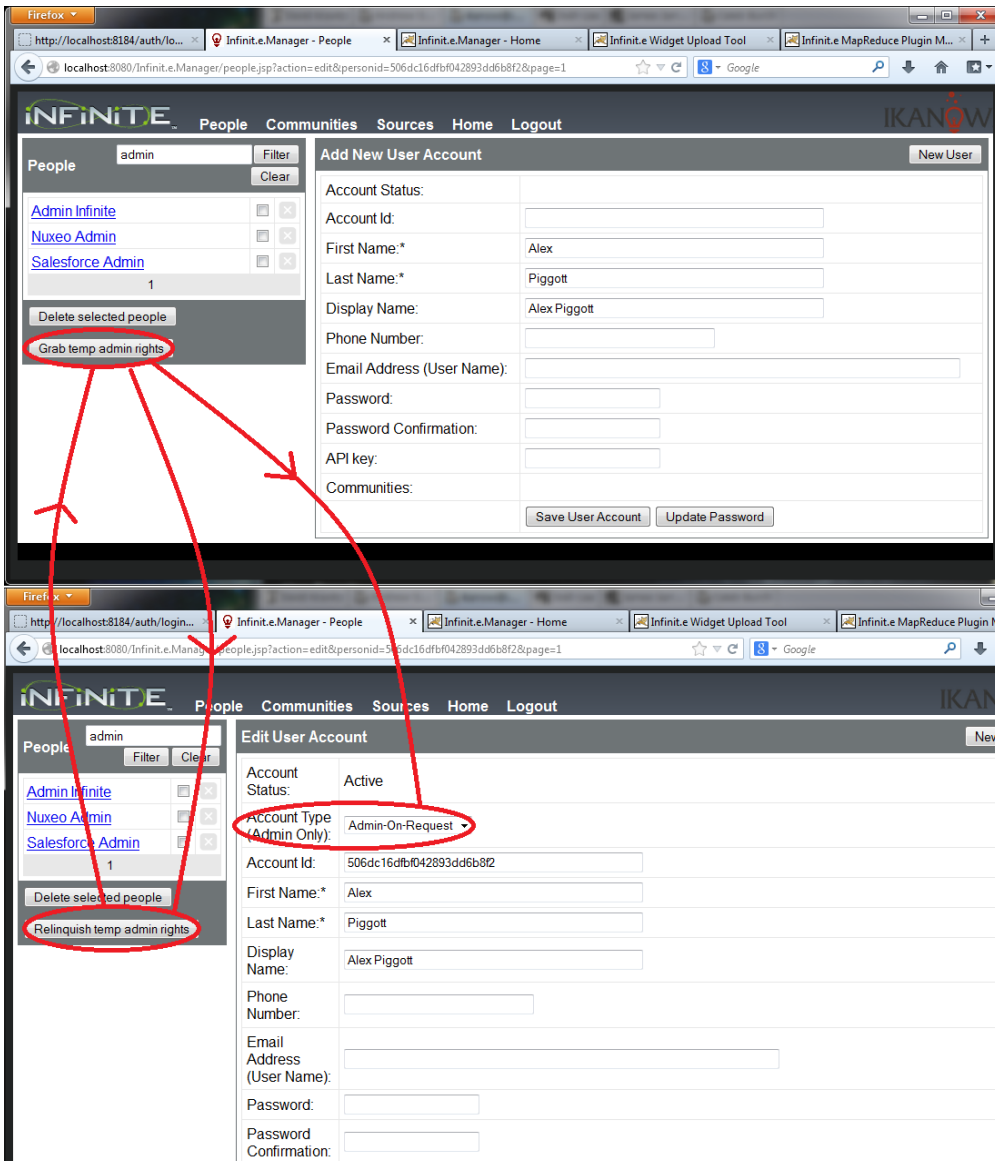
Temporary Admin Rights

From [August 2013](#) there is a new "beta" account type: "admin-on-request" (see screenshot below).

Users who have this account type are treated like normal users except:

- If they press the "Grab temp admin rights" button in the bottom left of the GUI (see below) then they are treated like full administrators for 10 minutes or until they logout or manually give up admin rights using the "Relinquish temp admin rights button" (see below)
- Also:
 - They can publish sources to any community without approval being requested
 - The harvester will always treat them like administrators (eg they can create file sources referencing "file://")

This account type is a good way of giving users admin rights only when they are needed and is recommended as the standard way of elevating privileges.



Infinit.e.Manager - Sources

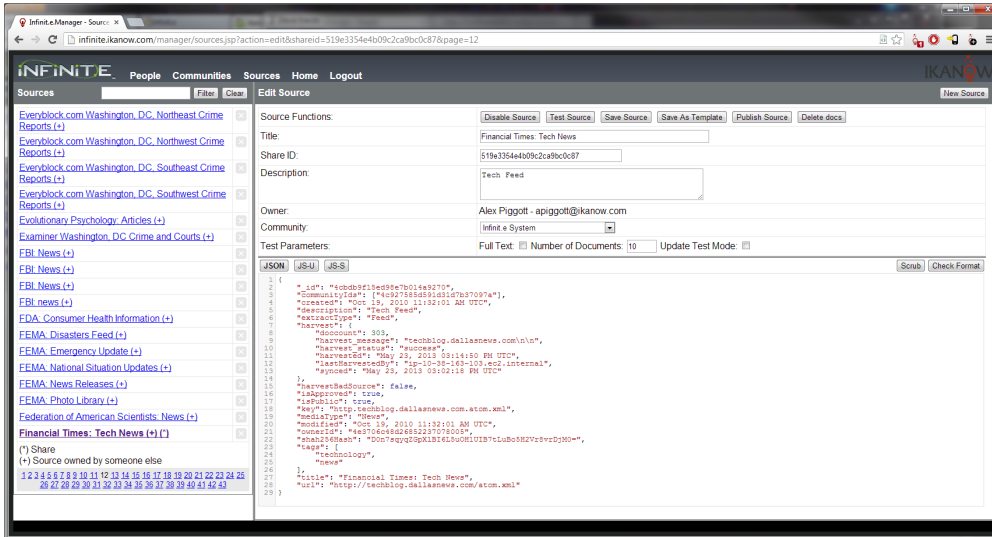
Overview



The Source Editor GUI is not currently compatible with IE. It is compatible with Chrome, Firefox, and Safari.

Source management is intrinsically a complex process (particularly when taking advantage of Infinit.e's customization engine).

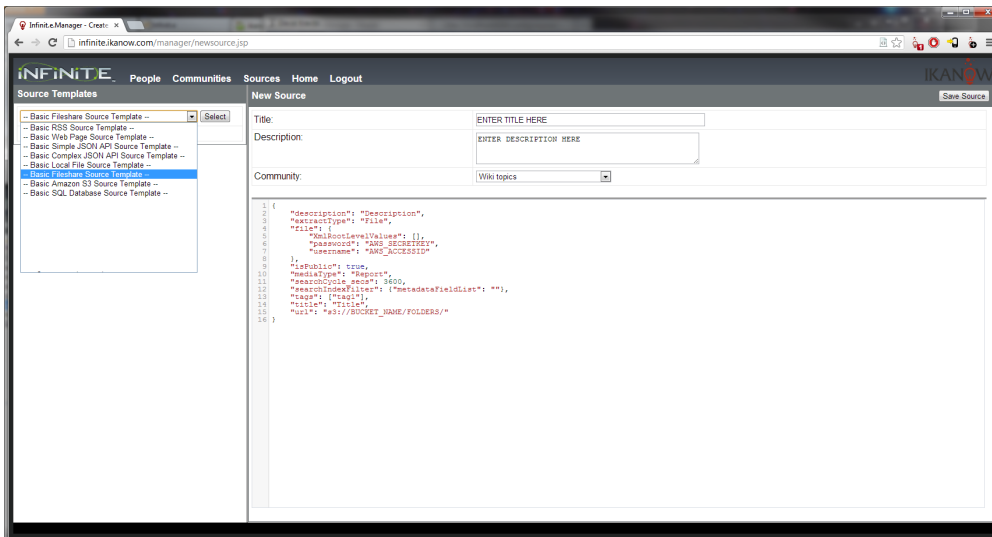
The Infinit.e.Manager Sources page provides a simple interface for adding and testing new sources, saving templates for future sources, and managing existing ones. Future iterations of the tool will provide actual support for the difficult bits of source writing, such as writing Javascript and regexes.



Note that the grey lines can be dragged to increase or decrease the size of the editor window.

Create New Source

To create a new source click on the New Source button in the upper right hand corner of the page. The Infinit.e.Manager application will forward you to the Create New Source page shown below:



i When copying an existing source into the New Source window, that existing source should be "scrubbed" first (middle right, "Scrub" button) - otherwise the presence of the "_id"/"key" fields will mean that the old source is modified instead of a new one being created.

Edit Existing Sources

To edit an existing source click on the source's name in the list of Sources found on the left hand side of the page.

Note: There are three types of documents listed in the Sources list: published sources, shares that are editable copies of published sources, and shares that have not yet been published as sources. Shares are denoted by "(*)".

i If copying the logic of an existing source, it is recommended to first "scrub" it to remove any server-added fields (particularly "_id" and "key", which can overwrite the existing source).

i Note that "private" sources ("isPublic": "false") do not have all fields displayed unless you are an admin, community moderator, or the source owner. In this case, it is likely that testing them (or using them as the basis for a new source) will fail. Contact the source owner to get a full copy.

There are 3 tabs that can be edited:

- "JSON" - this is the full source including all fields
- "JS-U" - the [Unstructured Analysis Module](#) allows content to be transformed by "scriptlets" (xpath/regex/javascript) into document metadata. This view shows only the javascript maintained in "unstructuredAnalysis.script" - all of the logic can be written in here as separate functions, and then the scriptlets can be simple calls to these functions, to maximize the maintainability of the code in the source.
- "JS-S" - the [Structured Analysis Module](#) allows content to be transformed by "scriptlets" (xpath/regex/javascript) into document metadata. This view shows only the javascript maintained in "structuredAnalysis.script" - all of the logic can be written in here as separate functions, and then the scriptlets can be simple calls to these functions, to maximize the maintainability of the code in the source.
- "JS-RSS" - (only visible if the "searchConfig" field of "rss" is specified; use "Save Source" to reset visibility if it changes during editing) the [Feed Harvester](#) can use javascript (and xpath) to create multiple documents out of a single received feed. This view shows only the javascript maintained in "rss.searchConfig.globals" - all of the logic can be written in here as separate functions, and then the scriptlets can be simple calls to these functions, to maximize the maintainability of the code in the source.

Validating the Source Format

To check the Source JSON format is valid at any time, select the "Check Format" button (middle right).

If run on the "JS-U" or "JS-S" tabs then the javascript in "structuredAnalysis.script" or "unstructuredAnalysis.script" is checked instead.

This validation is run automatically before the source is saved, tested, enabled/disabled, or published. (Or when switching between the JSON/JS tabs). Note that the automatic validation does *not* run on the javascript, only on the JSON.

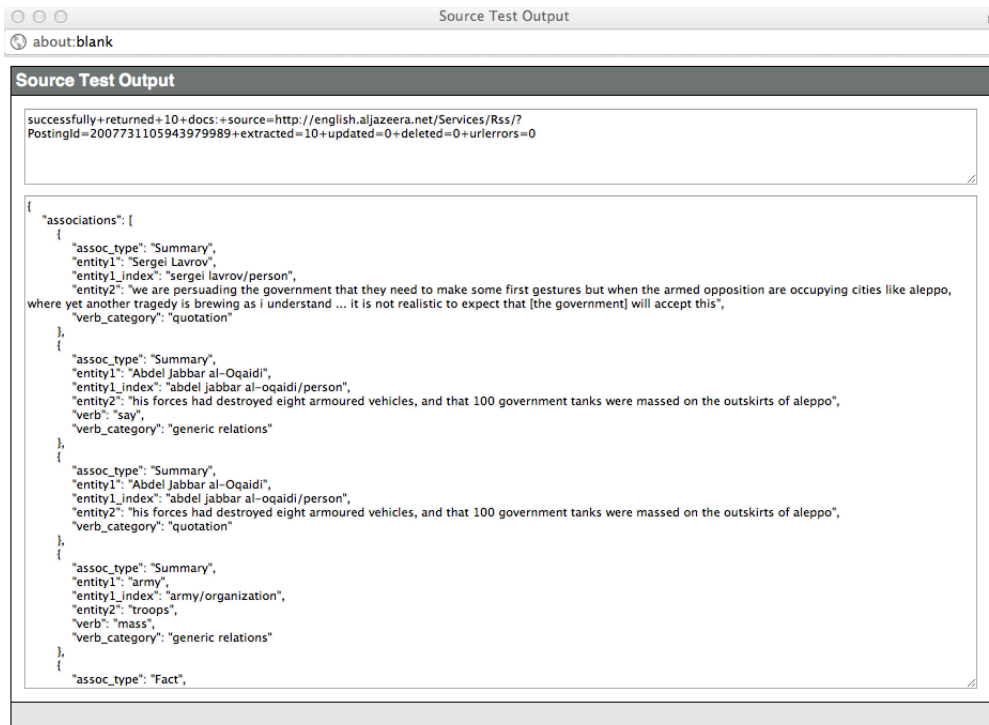
Testing a Source

Once a first draft of a source is complete it should be tested to see which documents it extracts and how it enriches the documents with additional metadata, entities, and associations, etc.

Two parameters can be set for testing the sources:

- "Full text": by default, the full text of a document is not returned (it can be quite long). For testing text extractors (eg "boilerpipe" vs "none" vs "AlchemyAPI"), or for testing "unstructured analysis" transformations, the text maybe useful or essential though; in these cases, enable this check box.
- "Number of documents": the maximum number of documents that will be enriched and returned. The smaller the number of documents, the quicker the API calls return.

Click on the Test Source button to start the testing process. Note that it can take a few minutes for the processed documents to be returned. Temporarily setting the "waitTimeOverride_ms" field of the "rss" object to be 1000 (ie 1s) can be useful during the debug stages.



```
successfully+returned+10+docs:+source=http://english.aljazeera.net/Services/Rss/?
PostinId=2007731105943979989+extracted=10+updated=0+deleted=0+urlerrors=0

{
  "associations": [
    {
      "assoc_type": "Summary",
      "entity1": "Sergei Lavrov",
      "entity1_index": "sergei lavrov/person",
      "entity2": "we are persuading the government that they need to make some first gestures but when the armed opposition are occupying cities like aleppo,
where yet another tragedy is brewing as i understand ... it is not realistic to expect that [the government] will accept this",
      "verb_category": "quotation"
    },
    {
      "assoc_type": "Summary",
      "entity1": "Abdel Jabbar al-Oqaidi",
      "entity1_index": "abdel jabbar al-oqaidi/person",
      "entity2": "his forces had destroyed eight armoured vehicles, and that 100 government tanks were massed on the outskirts of aleppo",
      "verb": "say",
      "verb_category": "generic relations"
    },
    {
      "assoc_type": "Summary",
      "entity1": "Abdel Jabbar al-Oqaidi",
      "entity1_index": "abdel jabbar al-oqaidi/person",
      "entity2": "his forces had destroyed eight armoured vehicles, and that 100 government tanks were massed on the outskirts of aleppo",
      "verb_category": "quotation"
    },
    {
      "assoc_type": "Summary",
      "entity1": "army",
      "entity1_index": "army/organization",
      "entity2": "troops",
      "verb": "mass",
      "verb_category": "generic relations"
    }
  ],
  "assoc_type": "Fact",
}
```



Note that the first time you test a source, you are likely to get an error accompanied by a request from the browser to allow/deny the window from launching pop ups. Select "Allow always" or the equivalent, refresh the browser if necessary, and press the test button again.

As can be seen from the above screen capture, the pop up contains 2 text elements:

- A status message including the number of documents returned, any errors or warnings encountered etc.
- The JSON of the extracted and enriched [document objects](#), if the test was successful.
 - Future versions of the tool will allow the documents to be viewed in widgets in the main GUI, providing a much easier interface to validate the source.

Based off the results from testing, the source can then be refined until the desired functionality is obtained.

Saving sources as templates

The Sources page allows you to save sources as templates to streamline the process creating new sources that share common attributes. To save a source as a template click on the Save Source as Template button. Note: Your new template will be available in the Source Templates drop down on the Create New Source page.

Note that templates are saved into your personal community only, but you can see any templates shared across any of the communities to which you belong. To share a template you have created with one of your communities, use the [file uploader](#).



Before turning a source into a template, that existing source should be "scrubbed" first (middle right, "Scrub" button) - otherwise the presence of the "_id"/"key" fields will mean that the old source is modified instead of a new one being created.

Publishing sources

Sources need to be "published" to the system in order for the Infinite Core Server to begin harvesting. Once you have created and tested a source, or edited and tested an existing source, you can publish the source by clicking on the Publish Source button.

If you submit (publish) a new source or to a community you do not own, then it is initially added in a "pending" state. An email is sent to the community owners and moderators, and they are given the option of allowing the source or not.

Editing sources that have previously been approved may not require further moderation, if only display fields have been modified; otherwise it is suspended pending approval as above.

Note that once a source has been published, its status can be monitored from "<ROOT URL>/InfiniteSourceMonitor.html" (eg <http://infinite.ikanow.com/InfiniteSourceMonitor.html>), provided you are logged into the main GUI or source builder.

After publishing a share, you should get an alert saying that the source has been published and the working copy "share" has been deleted. If you don't get this alert, then it is likely that an internal configuration error has occurred - contact your system administrator to get it fixed.

"Scrubbing" sources

As discussed above in a few places, this removes all fields added by the server after publishing, just retaining the actual ingest logic. It should be used before copying/templating.

If you accidentally scrub the source and then save it then you can get back to the original published source by just deleting the share and then re-selecting the source.

Enabling/disabling sources

Sources can be disabled by setting their "searchCycle_secs" to a negative number. This button just automates that process.



Note that this button only affects the un-published version of the source (ie the corresponding share). The source should be published to apply the change.

Deleting source's documents

This button will leave the source intact but will delete all of the documents harvested so far. It can only be performed on sources you own unless you are a community moderator or an admin.



Obviously, this function should be used with caution. Also for sources with many documents, this operation may take some time (eg 10 minutes for 500,000 documents).

Deleting sources or shares

To delete a source or share click on the "X" button next to the source name in the Sources list:

- **Share:** If the item you are deleting is a **Share** the system will ask you to confirm: "Do you really wish to delete the share: XXXXXXXXX"

(*)?". What happens next depends on whether or not the share has been published or not:

- If the share has been published the share is deleted but the published source is left alone and will appear in the Sources list.
 - If the share has not been published the share will simply be deleted and will disappear from the Sources list.
- **Source:** If the item you are deleting is a **Source** the system will ask you to confirm: "Do you really wish to delete the source: XXXXXXXX?". If you confirm the deletion the system will then delete the published source **and all harvested documents** associated with it.

i Note that deleting a published source will also delete all documents associated with that source. In some cases those documents will not be retrievable (eg old URLs from an RSS feed). This should therefore be used with caution. Also for sources with many documents, this operation may take some time (eg 10 minutes for 500,000 documents).

Monitoring sources

There is a graphical utility to monitor sources available from the home page (Source Monitor link). It opens in a new tab and is pictured below. It is not possible to change any source information from this GUI.

A subset of this information can also be accessed from the Source Manager dialog of the main GUI.

Stat...	Title	ID	Key	Community	Commu...	Appr...	Harv...	Last Har...	Doc Co...	Last Harvest Date	Harvest Error Mes...
Yellow	Business Wire: Te...	4cddb9f2...	feed:.origin.feed...	Infnit.e System	4:92758...	false	error		29348	Nov 1, 2011 8...	Extraction error
Yellow	Business Wire: Tr...	4cddb9f2...	http:.origin.feeds...	Infnit.e System	4:92758...	false	error		5229	Nov 9, 2011 4...	Extraction error
Green	Business Wire: Tr...	4cddb9f2...	feed:.origin.feed...	Infnit.e System	4:92758...	true	success	ip-10-38...	11357	May 23, 2013 03...	source=feed:/...
Green	Business Wire: Tr...	4cddb9f2...	feed:.origin.feed...	Infnit.e System	4:92758...	true	success	ip-10-38...	7492	May 23, 2013 03...	source=feed:/...
Green	Business Wire: W...	4cddb9f2...	http:.feeds.busine...	Infnit.e System	4:92758...	true	success	ip-10-38...	29907	May 23, 2013 03...	source=http:/...
Blue	BusinessWeek: Te...	4cddb9f0...	http:.www.busine...	Infnit.e System	4:92758...	true	in_pr...	ip-10-38...	5130	May 23, 2013 03...	source=http:/...
Green	California Medica...	4cddb9f...	http:.www.cmame...	Infnit.e System	4:92758...	true	success	ip-10-38...	0	May 23, 2013 03...	The requested ...
Green	California Veterin...	4cddb9f0...	http:.www.cvma.n...	Infnit.e System	4:92758...	true	success	ip-10-38...	80	May 23, 2013 03...	source=http:/...
Green	CalTech Today: D...	4cddb9f2...	http:.today.caltec...	Infnit.e System	4:92758...	true	success	ip-10-38...	601	May 23, 2013 03...	source=http:/...
Green	CalTech: Media R...	4cddb9f1...	http:.media.caltec...	Infnit.e System	4:92758...	true	success	ip-10-38...	161	May 23, 2013 03...	source=http:/...
Green	CancerCompass: ...	4cddb9e...	http:.www.cance...	Infnit.e System	4:92758...	true	success	ip-10-38...	0	May 23, 2013 03...	Invalid docum...
Blue	Carb Smart: Articl...	4cddb9e...	http:.www.carbs...	Infnit.e System	4:92758...	true	in_pr...	ip-10-38...	0	May 23, 2013 03...	Invalid XML: Er...
Green	Carnegie Mellon: ...	4cddb9f1...	http:.www.cmu.e...	Infnit.e System	4:92758...	true	success	ip-10-38...	459		source=http://www.cmu.edu/RSS/stories.rss source=http://www.cmu.edu/RSS/stories.rss source=http://www.cmu.edu/RSS/stories.rss
Yellow	CDU: E Coll Dub...	4cddb9e...	http:.www.zr.cdu...	Infnit.e System	4:92758...	false	success		5	May 31, 2011 8...	

The colors have the following meanings:

- Green: successfully harvested ("success")
- Blue: in progress ("in_progress")
 - (or has partially harvested, "success_iteration" - means that the most recent harvest cycle completed but not all available documents were harvested because of document/cycle limitations)
- Red: harvested with errors ("error")
- Yellow: not yet seen by a harvester, or currently unapproved.

Suspended sources retain their color status but have "[SUSPENDED]" prepended to their title.

Knowledge - Document - File - Get

`/knowledge/document/file/get/{sourceKey}/{path}`

i Returns a raw **document** harvested from a fileshare. This call preserves relative links since /s are used directly (not encoded) in the URL.

Authentication

Required, see Auth - Login

Arguments

sourceKey (required)

path (required)

The path (not URL encoded, ie "/"s are used directly) relative to the root path of the SMB (Samba/NetBIOS) fileshare of the source identified with "sourceKey". The encoding type of the response is determined by the file extension (defaulting to "application/octet-stream").


Example

<http://infinite.ikanow.com/api/knowledge/document/get/smb.fileshare.source.path/path/to/file/filename.extension>

Knowledge - Document - Get

`/knowledge/document/get/{docid}?returnFullText=[true|false|1|0]&returnRawData=[true|false|1|0]`

`/knowledge/document/get/{sourceKey}/{url}?returnFullText=[true|false|1|0]&returnRawData=[true|false|1|0]`

 Returns a **document** matching the id (or sourceKey/URL) provided.

Authentication

Required, see [Auth - Login](#)

Arguments

docid (required - unless sourceKey/URL specified)
id of the document of which to return the metadata

sourceKey (required - unless docid specified)

URL (required - unless docid specified)

Instead of specifying the document "_id", the sourceKey together with a URL-encoded URL can be specified (the 2 uniquely define any document within Infinite).

returnFullText (optional)

By default, the text of the document is not returned. Setting this parameter to "true" or "1" will return the first 50KB of the content, in the "fullText" JSON field (see below).

returnRawData (optional)

Only applies to files - if set to true (default is false) then returns the raw file directly from the fileshare. The extension of the stored filename is used to determine the reported encoding type. For non-files, this parameter is ignored.

Example

<http://infinite.ikanow.com/api/knowledge/document/get/4cc0ebff97622e5914a70e83>

<http://infinite.ikanow.com/api/knowledge/document/get/4cc0ebff97622e5914a70e83?returnFullText=1>

<http://infinite.ikanow.com/api/knowledge/document/get/rss.media.com.news/http%3a%2f%2fwww.media.com%2fcontent.html>

Example Response



```
{
  "response": {
    "action": "Doc Info",
    "success": true,
    "message": "Feed info returned successfully",
    "time": 23
  },
  "data": {
    "_id": "4e36d71880d77d6ea5b31ea5",
    "title": "vt123.kl",
    "url": "smb://modus:139/modus_input/inprocess/vt123.kl",
    "created": "Aug 1, 2011 12:40:56 PM",
    "modified": "Apr 12, 2011 3:03:33 PM",
    "publishedDate": "Sep 17, 2008 7:39:00 PM",
    "sourceKey": [
```

```
    "smb.modus.139.modus_input."
  ],
  "mediaType": [
    "Intel"
  ],
  "description": "BAGHDAD: CAR BOMBS ACROSS IRAQ, INCLUDING AT LEAST ONE
NEAR THE POLISH EMBASSY IN BAGHDAD, KILLED 20 PEOPLE AND WOUNDED DOZENS ON
MONDAY, POLICE SAID. A SUICIDE TRUCK BOMBER SLAMMED INTO A POLICE STATION
BUILDING IN DIJLAH, 130 KILOMETERS (80 MILES) NORTH OF BAGHDAD IN EARLY
AFTERNOON, KILLING NINE PEOPLE, INCLUDING THREE POLICEMEN, AND WOUNDING 26,
MOSTLY CIVILIANS. THE EXPLOSION ALSO DESTROYED OVER 20 SHOPS IN THE AREA AND
SOME 10 CARS NEARBY, DIJLAH SECURITY OFFICIAL ABDUL WAHAB AHMED SAID. IN
BAGHDAD, A 2 P.M. (11:00 GMT) BLAST NEAR THE POLISH EMBASSY IN THE DOWNTOWN
SHIITE-CONTROLLED KARRADAH DISTRICT KILLED TWO PERSONS AND WOUNDED FIVE. ALL THE
VICTIMS WERE IRAQIS, THE POLICE SAID. THE POLISH CHARGE D'AFFAIRES WALDEMAR
FIGAJ TOLD THE ASSOCIATED PRESS THAT THERE HAD BEEN A SERIES OF EXPLOSIONS
AROUND THE EMBASSY MONDAY BUT THAT THE CLOSEST WAS ABOUT 200 METERS (219 YARDS),
AND THAT THE EMBASSY HAD NO REASON TO BELIEVE IT \"WAS TARGETED IN ANY WAY.\"
POLISH AMBASSADOR GEN. EDWARD PIETRZYK, WAS WOUNDED WHEN HIS CONVOY WAS AMBUSHED
WITH ROADSIDE BOMBS LAST WEEK IN KARRADAH, IN AN ATTACK THAT ALSO KILLED A
POLISH SECURITY GUARD AND TWO OTHERS. FIGAJ SAID THE AMBASSADOR WAS STILL BEING
KEPT IN AN ARTIFICIAL COMA IN A HOSPITAL IN POLAND, A PROCEDURE HE SAID WAS
INDUCED TO ALLEVIATE PAIN BECAUSE PIETRZYK HAD SUSTAINED BURNS OVER 20 PERCENT
OF HIS BODY. ALSO MONDAY, A CAR BOMB WENT OFF AT A SMALL POPULAR MARKET NEAR
BAGHDAD UNIVERSITY'S TECHNOLOGY DEPARTMENT, KILLING FIVE CIVILIANS AND WOUNDING
15, POLICE SAID. THAT BOMB EXPLODED AT 12:30 A.M. (9:30 GMT) IN SINAA STREET, IN
DOWNTOWN BAGHDAD. THE WOUNDED INCLUDED FOUR WOMEN AND THREE CHILDREN. THE
CASUALTIES WERE TAKEN TO A NEARBY HOSPITAL.POLICE ALSO SAID AT LEAST SIX OTHER
CARS PARKED IN THE AREA WERE DAMAGED AS WELL BY THE BLAST. HALF AN HOUR
EARLIER, ANOTHER PARKED CAR BOMB WENT OFF NEAR A POLICE CHECKPOINT IN THE
PREDOMINANTLY SHIITE NEIGHBORHOOD OF KAMALIYAH IN EASTERN BAGHDAD, WOUNDING SIX
CIVILIANS AND DAMAGING AT LEAST THREE CARS, POLICE SAID. AP TELEVISION NEWS
FOOTAGE FROM THE MARKET BOMBING SHOWED THE MANGLED WRECKAGE OF THE CAR BOMB AND
DAMAGED CARS SURROUNDING IT. AMERICAN SOLDIERS WERE ALSO SEEN AT THE SCENE AND
U.S. HELICOPTERS FLEW OVER THE AREA. A SUICIDE CAR BOMBER ALSO STRUCK A POLICE
CHECKPOINT IN SADDAM HUSSEIN'S HOMETOWN OF TIKRIT, 130 KILOMETERS (80 MILES)
NORTH OF BAGHDAD, KILLING FOUR, INCLUDING THREE POLICEMEN, AND WOUNDING 10 OTHER
PEOPLE, POLICE SAID.",
  "entities": [
    {
      "disambiguated_name": "car",
      "index": "car/automobile",
      "actual_name": "CAR",
      "type": "automobile",
      "relevance": 0,
      "frequency": 8,
      "totalfrequency": 152,
      "doccount": 83,
      "dimension": "What"
    },
    {
      "disambiguated_name": "200",
      "index": "200/automobile",
      "actual_name": "200",
      "type": "automobile",
      "relevance": 0,
      "frequency": 1,
      "totalfrequency": 45,
      "doccount": 37,
    }
  ]
}
```

```
    "dimension": "What"
  },
  {
    "disambiguated_name": "police",
    "index": "police/organization",
    "actual_name": "POLICE",
    "type": "organization",
    "relevance": 0,
    "frequency": 8,
    "totalfrequency": 277,
    "doccount": 107,
    "dimension": "Who"
  },
  {
    "disambiguated_name": "victim",
    "index": "victim/person",
    "actual_name": "VICTIMS",
    "type": "person",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 24,
    "doccount": 13,
    "dimension": "Who"
  },
  {
    "disambiguated_name": "police",
    "index": "police/person",
    "actual_name": "POLICE",
    "type": "person",
    "relevance": 0,
    "frequency": 8,
    "totalfrequency": 248,
    "doccount": 107,
    "dimension": "Who"
  },
  {
    "disambiguated_name": "m",
    "index": "m/automobile",
    "actual_name": "M",
    "type": "automobile",
    "relevance": 0,
    "frequency": 2,
    "totalfrequency": 39,
    "doccount": 31,
    "dimension": "What"
  },
  {
    "disambiguated_name": "official",
    "index": "official/person",
    "actual_name": "OFFICIAL",
    "type": "person",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 88,
    "doccount": 46,
    "dimension": "Who"
  },
  {
    "disambiguated_name": "person",
```

```
"index": "person/person",
"actual_name": "PERSONS",
"type": "person",
"relevance": 0,
"frequency": 1,
"totalfrequency": 7,
"doccount": 5,
"dimension": "Who"
},
{
  "disambiguated_name": "people",
  "index": "people/person",
  "actual_name": "PEOPLE",
  "type": "person",
  "relevance": 0,
  "frequency": 3,
  "totalfrequency": 137,
  "doccount": 77,
  "dimension": "Who"
},
{
  "disambiguated_name": "80",
  "index": "80/automobile",
  "actual_name": "80",
  "type": "automobile",
  "relevance": 0,
  "frequency": 2,
  "totalfrequency": 8,
  "doccount": 7,
  "dimension": "What"
},
{
  "disambiguated_name": "civilian",
  "index": "civilian/person",
  "actual_name": "CIVILIANS",
  "type": "person",
  "relevance": 0,
  "frequency": 3,
  "totalfrequency": 60,
  "doccount": 39,
  "dimension": "Who"
},
{
  "disambiguated_name": "policemen",
  "index": "policemen/person",
  "actual_name": "POLICEMEN",
  "type": "person",
  "relevance": 0,
  "frequency": 2,
  "totalfrequency": 35,
  "doccount": 24,
  "dimension": "Who"
},
{
  "disambiguated_name": "truck",
  "index": "truck/automobile",
  "actual_name": "TRUCK",
  "type": "automobile",
  "relevance": 0,
```

```
"frequency": 1,
"totalfrequency": 45,
"doccount": 29,
"dimension": "What"
},
{
  "disambiguated_name": "suicide truck bomber",
  "index": "suicide truck bomber/person",
  "actual_name": "SUICIDE TRUCK BOMBER",
  "type": "person",
  "relevance": 0,
  "frequency": 1,
  "totalfrequency": 3,
  "doccount": 3,
  "dimension": "Who"
},
{
  "disambiguated_name": "bomber",
  "index": "bomber/person",
  "actual_name": "BOMBER",
  "type": "person",
  "relevance": 0,
  "frequency": 2,
  "totalfrequency": 42,
  "doccount": 30,
  "dimension": "Who"
},
{
  "disambiguated_name": "went",
  "index": "went/drug",
  "actual_name": "WENT",
  "type": "drug",
  "relevance": 0,
  "frequency": 2,
  "totalfrequency": 37,
  "doccount": 31,
  "dimension": "What"
},
{
  "disambiguated_name": "woman",
  "index": "woman/person",
  "actual_name": "WOMEN",
  "type": "person",
  "relevance": 0,
  "frequency": 1,
  "totalfrequency": 29,
  "doccount": 15,
  "dimension": "Who"
},
{
  "disambiguated_name": "child",
  "index": "child/person",
  "actual_name": "CHILDREN",
  "type": "person",
  "relevance": 0,
  "frequency": 1,
  "totalfrequency": 22,
  "doccount": 15,
  "dimension": "Who"
}
```

```
    },
    {
      "disambiguated_name": "he",
      "index": "he/person",
      "actual_name": "HE",
      "type": "person",
      "relevance": 0,
      "frequency": 1,
      "totalfrequency": 147,
      "doccount": 83,
      "dimension": "Who"
    },
    {
      "disambiguated_name": "suicide car bomber",
      "index": "suicide car bomber/person",
      "actual_name": "SUICIDE CAR BOMBER",
      "type": "person",
      "relevance": 0,
      "frequency": 1,
      "totalfrequency": 6,
      "doccount": 6,
      "dimension": "Who"
    },
    {
      "disambiguated_name": "soldier",
      "index": "soldier/person",
      "actual_name": "SOLDIERS",
      "type": "person",
      "relevance": 0,
      "frequency": 1,
      "totalfrequency": 82,
      "doccount": 44,
      "dimension": "Who"
    },
    {
      "disambiguated_name": "baghdad, iraq",
      "index": "baghdad, iraq/city",
      "actual_name": "BAGHDAD",
      "type": "city",
      "relevance": 0,
      "frequency": 8,
      "totalfrequency": 406,
      "doccount": 159,
      "geotag": {
        "lat": 33.3386111,
        "lon": 44.3938889
      },
      "dimension": "Where",
      "ontology_type": "city"
    },
    {
      "disambiguated_name": "38smb50",
      "index": "38smb50/geographicfeature",
      "actual_name": "38SMB5083493081",
      "type": "geographicfeature",
      "relevance": 0,
      "frequency": 1,
      "totalfrequency": 8,
      "doccount": 8,
    }
  ],
  {
    "disambiguated_name": "38smb50",
    "index": "38smb50/geographicfeature",
    "actual_name": "38SMB5083493081",
    "type": "geographicfeature",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 8,
    "doccount": 8,
  }
]
```

```

    "geotag": {
      "lat": 33.37586144623292,
      "lon": 44.47143713123417
    },
    "dimension": "Where",
    "ontology_type": "point"
  },
  {
    "disambiguated_name": "police station building in",
    "index": "police station building in/facility",
    "actual_name": "POLICE STATION BUILDING IN",
    "type": "facility",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 1,
    "doccount": 1,
    "dimension": "What"
  },
  {
    "disambiguated_name": "hospital in poland",
    "index": "hospital in poland/facility",
    "actual_name": "HOSPITAL IN POLAND",
    "type": "facility",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 1,
    "doccount": 1,
    "dimension": "What"
  },
  {
    "disambiguated_name": "shops in the",
    "index": "shops in the/facility",
    "actual_name": "SHOPS IN THE",
    "type": "facility",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 2,
    "doccount": 2,
    "dimension": "What"
  },
  {
    "disambiguated_name": "tikrit, iraq",
    "index": "tikrit, iraq/city",
    "actual_name": "TIKRIT",
    "type": "city",
    "relevance": 0,
    "frequency": 1,
    "totalfrequency": 1,
    "doccount": 1,
    "geotag": {
      "lat": 34.5966667,
      "lon": 43.6769444
    },
    "dimension": "Where",
    "ontology_type": "city"
  },
  {
    "disambiguated_name": "baghdad",
    "index": "baghdad/facility",

```

```
"actual_name": "MARKET NEAR BAGHDAD UNIVERSITY",
"type": "facility",
"relevance": 0,
"frequency": 1,
"totalfrequency": 2,
"doccount": 2,
"dimension": "What"
},
{
  "disambiguated_name": "hospital",
  "index": "hospital/facility",
  "actual_name": "NEARBY HOSPITAL",
  "type": "facility",
  "relevance": 0,
  "frequency": 1,
  "totalfrequency": 7,
  "doccount": 7,
  "dimension": "What"
},
{
  "disambiguated_name": "cars parked in the",
  "index": "cars parked in the/facility",
  "actual_name": "CARS PARKED IN THE",
  "type": "facility",
  "relevance": 0,
  "frequency": 1,
  "totalfrequency": 1,
  "doccount": 1,
  "dimension": "What"
},
{
  "disambiguated_name": "cars",
  "index": "cars/automobile",
  "actual_name": "CARS",
  "type": "automobile",
  "relevance": 0,
  "frequency": 2,
  "totalfrequency": 22,
  "doccount": 21,
  "dimension": "What"
},
{
  "disambiguated_name": "bmw",
  "index": "bmw/automobile",
  "actual_name": "M",
  "type": "automobile",
  "relevance": 0,
  "frequency": 2,
  "totalfrequency": 37,
  "doccount": 30,
  "dimension": "What"
},
{
  "disambiguated_name": "audi",
  "index": "audi/automobile",
  "actual_name": "200",
  "type": "automobile",
  "relevance": 0,
  "frequency": 3,
```



```
        "totalfrequency": 61,  
        "doccount": 48,  
        "dimension": "What"  
    }  
],  
"tags": [  
    "Modus",  
    "IED",  
    "Intel",  
    "extraction",  
    "Iraq"  
],  
"communityId": [  
    "4db5c05fb246d25364aceca0"  
],  
"associations": [],  
"docGeo": {  
    "lat": 33.37586144623292,  
    "lon": 44.47143713123417  
}
```

```
}  
}
```

Knowledge - Document - Query

/knowledge/document/query/{community-ids}?<parameters, see below> (GET)

/knowledge/document/query/{community-ids} (POST)



This is the most important API call Infnit.e provides. It takes in a complex query object that controls what input sources are used, what information is desired, how scoring should be performed, and what form the returned information should take.

Authentication

Required, see [Auth - Login](#). There are various exceptions for [RSS output](#)

Arguments

community-ids (required)

Community ID, or IDs (comma-separated), or [community ID - regex](#) to control which sources the query will use. Obviously the logged-in user must have access rights to those communities. A user's communities can be viewed using the "[person/get](#)" API call

Parameters/Basic Usage

Because of the complexity of this API call, it will be documented in a number of different pages, covering:

- [Query basics](#) (this page)
- [Available query terms](#):
 - [Free text and exact text](#)
 - [Entity queries](#)
 - [Entity aliasing](#)
 - [Geospatial queries](#)
 - [Temporal queries](#)
 - [Event/association queries](#)
 - [Combining query terms](#)
 - [The "raw" query term](#)
- [Scoring parameters](#)
- [Input options](#) (within the sources available based on the community list)
- [Output options](#)
 - [General options and format](#)
 - [Document format options](#)
 - [Aggregation options](#)
 - [Filtering options](#)

The remainder of this section provides some top level documentation.

The query request is represented by the following (top level) JSON object:

```
{
  "qt": [ { ... } ],
  "expandAlias": false, // See under "entity aliasing" above
  "logic": string, // see query object above
  "raw": { ... },

  "input": { ... },
  "score": { ... },
  "output": { ... },

  "explain": boolean // (optional, if present and true then each document has an
  "explain" field added, see below)
}
```

There are 3 ways to send this JSON object to the REST endpoint (see examples below):

1. "POST" the JSON object to the URL described above (note that the "contentType" and "Accept" headers in the request field may need to be set to "application/JSON" - eg to use the AS3 HTTPService)
2. Perform a "GET" to the URL described above, with the "json=" parameter set to a URL-encoded single-line representation of the JSON object
3. Perform a "GET" to the URL described above, with the different non-null JSON terms in the object above turned into individual URL parameters using the dot notation, eg "&qt[0].etext="exact text"&output.docs.enable=true" (see examples below).

Note that the Widget API also allows visualization widgets to read, modify, and send these query objects to the server. In this case only the JSON object is required, the transport is handled by the framework and the community set cannot be changed.

The optional "explain" boolean is (primarily) a diagnostic aid that adds the [elasticsearch explain object](#) to each returned document (embedded as JSON inside a field called "explain").

Examples

Queries can be made from the command-line or a URL browser in a few different ways:

Using 'curl' to POST a query

```
#bash> curl -XPOST
'http://infinite.ikanow.com/api/knowledge/document/query/4e0c7e99eb5af0fbdcfbf697' -d
'{
  "qt": [
    {
      "etext": "BBC "
    }
  ],
  "score": {
    "numAnalyze": 1000,
    "sigWeight": 67,
    "relWeight": 33
  },
  "logic": "1",
  "output": {
    "format": "json",
    "docs": {
      "facts": true,
      "eventsTimeline": false,
      "enable": true,
      "geo": true,
      "ents": true,
      "numReturn": 100,
      "summaries": true,
      "events": true,
      "skip": 0
    },
    "aggregation": {
      "factsNumReturn": 100,
      "sourceMetadata": 20,
      "geoNumReturn": 100,
      "eventsNumReturn": 100,
      "entsNumReturn": 100,
      "timesInterval": "1w",
      "sources": 20
    }
  }
}'
```

Using URL parameters to query via GET

```
#bash> curl -XPOST
'http://infinite.ikanow.com/api/knowledge/document/query/4e0c7e99eb5af0fbdcfbf697?qt[0
].etext="BBC"&input.tags="news"&score.sigWeight=33&output.aggregation.factsNumReturn=1
00'
```

Using a URL-encoded JSON object to query via GET

```
#bash> curl -XPOST
'http://infinite.ikanow.com/api/knowledge/document/query/4e0c7e99eb5af0fbdcfbf697?json=
{"qt":{"etext":"BBC
"}]}%2C"logic":"1"%2C"output":{"aggregation":{"factsNumReturn":"100"%2C"sourceMet
adata":{"geoNumReturn":"100"%2C"eventsNumReturn":"100"%2C"entsNumReturn":"100"%2
C"timesInterval":"1w"%2C"sources":{"format":"rss"}}}'
```

Note that with these last 2, the URL could also be pasted into the URL bar of a browser (assuming a tab in that browser had acquired a cookie via a log-in call).

Accessing the query from within `ActionScript` is also a common operation (although note that the `Infinite` main GUI and the module plugins provide an encapsulated version of this service already):

Invoking the query method from within ActionScript

```
private function buildQueryRequest(communityIdList:String, queryObj:Object):void
{
    var service:mx.rpc.http.mxml.HTTPService = new mx.rpc.http.mxml.HTTPService();
    service.method = "POST";
    var header:Object=new Object();
    header["Accept"] = "application/json";
    service.contentType = "application/json";
    service.headers = header;
    service.url = BASE_URL + "knowledge/document/query/" + communityIdList;
    service.addEventListener(ResultEvent.RESULT, onQuerySuccess);
    service.addEventListener(FaultEvent.FAULT, onQueryFailure);
    service.send(JsonEncoder.encode(queryObj));
}
private function onQuerySuccess(event:ResultEvent):void
{
    // Handle a successful query
}
private function onQueryFailure(event:ResultEvent):void
{
    // Handle a query failure, see below
}
```

Perhaps more useful is this code fragment showing how to access the query API call from Javascript, eg for lighter weight uses of `Infinite`. For the purposes of the example below illustrates how to use `jQuery` to call the `Infinite` API using both a 'GET' and 'POST'. For more information on create data visualizations using `AJAX`, `jQuery` and `Protovis` or `d3.js` check out our [blog](#).

The example illustrates how to use `jQuery.ajax()` to invoke a `AJAX` call using a 'GET' request. Upon success you can then process the response.

Invoking the query method using a 'GET' request from within JavaScript using jQuery

```
$.ajax( {
    type: 'GET',

    url: 'http://infinite.ikanow.com/api/knowledge/document/query/4e0c7e99eb5af0fbdcfbf697'
    ,

    data: 'qt[0]="BBC"&input.tags="news"&score.sigWeight=33&output.aggregation.factsNumReturn=100'

    dataType: 'json',
    success: function(response) {
        // Logic Processing
    }
})
```

The example illustrates how to use `jQuery.post()` to invoke a AJAX call using a 'POST' request. Upon success you can then process the response.

Invoking the query method using a 'POST' request from within Javascript using jQuery

```
$.ajax({
    type: 'POST',
    url:
'http://infinite.ikanow.com/api/knowledge/document/query/4e0c7e99eb5af0fbdcfbf697',
    data: '{
        "qt":[
            {
                "etext": "BBC"
            }
        ],
        "score": {
            "numAnalyze": 1000,
            "sigWeight": 67,
            "relWeight": 33
        },
        "qtOptions": null,
        "logic": "1",
        "output": {
            "format": "json",
            "docs": {
                "facts": true,
                "eventsTimeline": false,
                "enable": true,
                "geo": true,
                "ents": true,
                "numReturn": 100,
                "summaries": true,
                "events": true,
                "skip": 0
            },
            "aggregation": {
                "factsNumReturn": 100,
                "sourceMetadata": 20,
                "geoNumReturn": 100,
                "eventsNumReturn": 100,
                "entsNumReturn": 100,
                "timesInterval": "1w",
                "sources": 20
            }
        }
    }',
    dataType: 'json',
    success:function(response) {
        // Logic Processing
    }
});
```

Error Response

Currently the error responses are fairly basic:

- Authentication error (normally means that you are not logged in):

```
{
  "response": {
    "action": "Cookie Lookup",
    "success": false,
    "message": "Cookie session expired or never existed, please login first",
    "time": 0
  }
}
```

- All other errors:

```
{
  "response": {
    "action": "Query"
    "success": false
    "message": "Unknown query error"
    "time": 0
  }
}
```

Typical candidates for "all other errors" include:

- Incorrect or unauthorized community ID strings
- Syntax errors in the JSON or "logic" field


Note that future releases will provide better error reporting.

Knowledge - Document - Query - Manual Aliases

Manual Aliases

The Infini.t.e query engine provides support for the following activities:

- Aliasing several different entities into a single entity
 - This applies both to queries, ie adding the aliases to queries involving the master entity, and also to query results, ie replacing "aliased entities" with the "master entity"
- Discarding unwanted entities

 It should be noted that setting aliases does not affect the stored documents, the transformation occurs in the API before documents are returned.

This means that alias changes are quick-to-apply, easily reversible, and different users can have different aliases; the downside is that the process is not perfect - the statistics are slightly less accurate, there will be occasional duplicate entities/associations (though entity aggregations are never duplicated) etc.

Unlike other query engine controls, aliasing is not configured from either the URL/POST parameters or from the static configuration. Instead [JSON share](#) are used. This makes it easy to maintain large numbers of aliases and also to manage which users have which alias configuration enabled. The following sections describe how to write, maintain, and share these JSON configurations.

Share format

The format of the alias configuration object is as follows:


```

{
  "<alias-of-master-entity>": { // ie index in the format disambiguatedName/type
    "index": string, // OPTIONAL, just used for display, if present should be the same
as the index "key"
    "disambiguatedName": string, // The disambiguated name corresponding to the index
"key"
    "type": string, // The type corresponding to the index "key"
    "alias": [ string ], // A list of indexes of entities that should be aliased to the
master entity
    "linkdata": [ string ] // A list of "etext" terms that are added to any query terms
involving the master entity
  },

  // Other master entities, in the same format

  // Optionally:
  "DISCARD": {
    "alias": [ string ] // A list of indexes of entities to be discarded
  }
}

```

Developers who use the [Java API/Java driver](#) will notice that this format is simply a map of [EntityFeaturePojo](#) objects, which makes its use in Java clients easy.

Here is an example of an alias file:

```

{
  "brooklyn, ny/location": {
    disambiguated_name: "Brooklyn, NY",
    type: "Location",
    dimension: "Where",
    index: "brooklyn, ny/location",
    alias: ["brooklyn/keyword"]
  },
  "DISCARD": {
    disambiguated_name: "DISCARD",
    index: "DISCARD",
    type: "SPECIAL",
    alias: [
      "sandy/keyword",
      "rt/keyword",
      "amp/keyword"
    ]
  },
  "new york city, ny/location": {
    disambiguated_name: "New York City, NY",
    type: "Location",
    dimension: "Where",
    index: "new york city, ny/location",
    alias: [
      "nyc/keyword",
      "downtown nyc/keyword",
      "new york city/keyword",
      "new york city/location"
    ]
  }
}

```

Alias management

When a query is performed, all JSON share objects **belonging to the queried communities, and with type "infinite-entity-alias"**, are read from the database (or from cache if unchanged).

Where multiple entities are aliased differently across the files, it is undefined which setting will be used, **EXCEPT THAT** the aliases from any shares solely in the user's personal share will always override community-wide aliases.



The reason for this priority is to allow users to utilize their personal communities as staging areas to test new aliases. That way there is no risk of damaging other users' queries (and conversely the aliases "under test" will always be applied, even if they conflict with operationally deployed aliases).

Once an alias file has been created it should be uploaded either using the [Share - Add - JSON API call](#) or from the [file uploader](#).

Files uploaded via the API can then be shared with other communities using the [Share - Add - Community](#) or [Share - Remove - Community](#) calls or again from the [file uploader](#).

Aliasing can be manually disabled by setting the top-level query field "expandAlias" to false (not to be mistaken for the false-by-default "expandAlias" fields that are present in the [query terms](#), and which apply to the "automatic aliases" generated from entity extraction, ie the "actual names" corresponding to the entity's disambiguated names - this nomenclature will be tidied up in a forthcoming version of the platform).

Aliasing is a complicated topic, and the following additional functional items are planned on the roadmap:

- The ability to apply aliases at harvest time (unlike the current aliasing this will not be dynamic, but it may be desirable for very well established aliases - in the meantime, "caches" in the [unstructured](#) and [structured](#) harvesters can be used)
- Currently aliases are not passed to [Hadoop plugins](#). This will be added in a later version.
- Currently aliases are not applied to documents returned from [Knowledge - Document - Get](#). This will be added in a later version.
- As mentioned above, the nomenclature between the old (not-very-useful-in-practice) "automatic aliases" and the new manual aliases

described here needs to be tidied up.

- Currently there is no way for community moderators or owners to authorize shares - any member of a community can share their JSON alias set and it will be automatically applied to any appropriate queries.
 - (It is easy to see which alias sets are being applied however, either from the [file uploader](#) or the [Share - Search API call](#), searching on type="infinite-entity-alias")

Knowledge - Query - Input Options

Overview

There are two ways that a user can control the sources over which the query is applied:

- By selecting which communities from those available are used. This is part of the URL as described in the [query overview documentation](#)
- By creating an "input" object at the query top level. This is described in the remainder of this page.

Input specification

Input JSON specification

```
{
  "input": {
    // One of the following:
    "tags": [ string ],
    "typesAndTags": [ { "type": string, "tags": [ string ] } ],
    "sources": [ string ],
    "srcInclude": boolean // (only modifies "sources" field)
  }
}
```

As can be seen in the above specification, there are three different ways of specifying the source set over which the query runs:

- If any "tags" are specified then only documents with any (case insensitive/exact) matching tags are included in the query
- "typesAndTags" is similar, except you can restrict a set of tags to documents of a given "type"
- "sources" is the simplest, specify a set of strings, each of which are compared against documents "sourceKey" fields (taken from [sources' "key" fields](#)), and then:
 - if "srcInclude": true (default), then only matching documents are included in the query
 - if "srcInclude": false, then only non-matching documents are included in the query
- If more than one of the above fields is specified, then the results are OR'd together

Source information (such as the source key, and tags and types) can be obtained via the ["person/get" API call](#).

Examples

Example 1:

```
{
  "input": {
    "tags": [ "topic:cyber", "topic:defense" ]
  }
}
```

Example 2: as above, but restricted to blogs

```
{
  "input": {
    "typesAndTags": [ {
      "type": "Blog",
      "tags": [ "topic:cyber", "topic:defense" ]
    } ]
  }
}
```

Example 3: search YouTube only

```
{
  "input": {
    "sources": [
      "http.gdata.youtube.com.feeds.base.standardfeeds.most_recent.client=ytapi-youtube-browser.alt=rss" ]
  }
}
```

Example 4: exclude YouTube and Flickr from the search

```
{
  "input": {
    "sources": [

      "http.gdata.youtube.com.feeds.base.standardfeeds.most_recent.client=ytapi-youtube-browser.alt=rss",
      "http.api.flickr.com.services.feeds.photos_public.gne"
    ],
    srcInclude: false
  }
}
```

Knowledge - Query - Output options

Overview

The "output" object allows the configuration of 3 different logical functions:

- The **output format**: JSON or XML (functionally equivalent), RSS (behaves differently in terms of both functionality and authentication), and KML is coming soon.
- The **document format**: how many documents to return from a query
- **Aggregations**: these are often the most important outputs from Infinit.e - aggregations over all matching documents of various fields. Understanding how to configure them and the format of the resulting objects is important to taking most advantage of the platform. The aggregations section includes the powerful "moments" aggregation.
- **Filtering**: this allows callers to restrict retrieved **documents, entities and associations** based on either **entity type** or **association verb category**.

The sections below show the configuration of each of these different functions. They can obviously be combined into a single object, eg:

Output options

```
{
  "output": {
    "format": string,
    "docs": { ... },
    "aggregation": {...},
    "filter": {...}
  }
}
```

Output format

Output format

```
{
  "output": {
    "format": string, // "json" (default), "xml", or "rss"
  }
}
```

The only complication is RSS. Integrating REST authentication with RSS readers is a known problem, and Infinite currently provides a few different options:

- If the query is made from a browser that is already logged in (eg the "RSS" button from the GUI) or with a cookie obtained from [loginthen](#) it works as normal.
 - *(The cookies lifetime is only 30 minutes (server-side configurable), so this is not a viable long term option eg for use in RSS readers.)*
- In addition, and different to "json"/"xml", it is also possible to use "user=" and "pword=" URL parameters. The password can either be clear text (not recommended, obviously), or [SHA-256](#) and Base-64 encoded ([this site can be used for testing](#)).
 - This enables users to generate arbitrary queries and store them in RSS readers, at the expense of showing a password that can at best be used by others in any other REST function.
 - *(A future version of the tool will force these queries to be over SSL, to mitigate the risk this presents somewhat.)*
- Finally, a key for a specific query can be generated from the GUI, and then this query can be used without any authentication at all.
 - Obviously this final authentication bypass is terrible for a few reasons (eg the key is only protected by "security through obscurity!"), and it is only a temporary solution.
 - *(A future version will provide a "license key" that will be usable only for RSS, to replace this.)*



When requesting RSS via a key, you will need to supply the users communityId as the first id in the communityIds json call e.g. {"communityIds":["USER_ID","ALL_OTHER_COMMS"]}

This is to ensure a user has access to the communities you supply since the api call is being made unauthenticated.

Examples of using RSS with various authentication methods:

- Username and clear text password (type into the URL): <http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?input.tags=topic:news&output.format=rss&user=email@domain.com&pword=mlckeym0use>
- Username and encrypted password (type into the URL): <http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?input.tags=topic:news&output.format=rss&user=email@domain.com&pword=u6JP3GVV5PSwvU4/1eFUj+kUPsAWhKa0eOqCFOrDyNQ=>
- Query-specific key:

```
curl -XPOST
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?key=GcC6
1AotZObZYxTgdKW9pbFTWdpCk8EkSIzQlCM3XDM' -d'{
"qt": [ { "ftext": "*" } ],
"input": { "tags": [ "topic:news" ] }, "output": { "format": "rss" } }'
```

Note finally that RSS (unlike XML and JSON) only provides the document URLs, none of the metadata.

Document formats

Document formats

```
{
  "output": {
    "docs": {
      // Whether to return documents/how many:
      "enable": boolean, // (defaults to true)
      "numReturn": integer, // (defaults to 100, maximum is 10K - not advised unless all
scoring is turned off)
      "skip": integer, // (defaults to 0)
      // Alternative/complement to documents:
      "eventsTimeline": boolean, // (defaults to false)
      "numEventsTimelineReturn": integer, // (defaults to 1000)
      // Which sub-objects to return per document:
      "ents": boolean, // (all of these default to true)
      "geo": boolean,
      "events": boolean,
      "facts": boolean,
      "summaries": boolean,
      "metadata": boolean
    }
  }
}
```

Controlling the number of documents returned

As described in the section on [scoring](#), documents are sorted according to a scoring algorithm, and are retrieved in order. The "numReturn" field dictates how many are returned to the user, and the "skip" field allows a primitive concept of paging (eg "?output.docs.numReturn=10&skip=0", "?output.docs.numReturn=10&skip=10", "?output.docs.numReturn=10&skip=20", etc). There are two reasons to limit the number of documents (we think 100 works quite well within general visualization GUIs):

- For performance reasons
- The point of the Infinite tool is to extract "knowledge" for a corpus of documents, therefore cluttering the display with a large number of documents could be viewed as counter productive.

If "&output.docs.enable=false" then no documents are returned.

Controlling the format of documents returned

The [document format](#) is described here. As can be seen there, documents have a number of sub-objects: entities, events (which are then sub-divided into "Events", "Facts", and "Summaries"), and source-specific metadata, see (TODO link to structured and unstructured analysis modules)).

The "ents", "geo", "events", "facts", "summaries", and "metadata" fields are simply booleans that control whether these sub-objects are included. The main reason for not including them is just to avoid cluttering up and slowing down requests where they are not needed.

Note that "geo" controls whether entities with (lat,long)s are included - eg for geospatial apps it maybe that most entities are not of interest, but geotagged ones are: in this case the pairing "&ents=false&geo=true" would be used.

Events timeline

The most significant of these parameters is "output.docs.eventsTimeline". This generates a new output array, consisting of the "event" sub-objects (for "Events", "Facts", and "Summaries").

Events are taken from the top "score.numAnalyze" matching documents - the "top" specified documents are returned, where "top" is based on the Pythagorean sum of the documents containing each event. Note that "score.numAnalyze" currently controls two other important output components:

- How many documents are analyzed and scored (using a combination of [Lucene/Significance](#)) to determine which ones to return as part of the [document output](#).
- How many entities are analyzed are scored ([Significance](#) only) to determine which ones to return as part of [entity aggregations](#).

The fields "output.docs.events", "output.docs.facts", "output.docs.summaries" control which of "Events", "Facts" and "Summaries" are included in the timeline.

Example query/output ([link to JSON format specification](#)):

Geo-spatial aggregation example

```
//curl -XGET
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?qt[0].etext="
*&input.tags="topic:technology"&output.docs.enable=false&output.docs.eventsTimeline=t
rue'
{
  response: { ... }
  stats: { ... }
  eventsTimeline: [
    {
      entity1: "ev solar carport"
      entity1_index: "ev solar carport/facility"
      verb: "deliver"
      verb_category: "generic relations"
      entity2: "125 mw hours"
      event_type: "Summary"
      time_start: "2011-05-26"
      assoc_sig: 126.6919059017276,
      doccount: 1
    },
    {
      entity1: "aol advertising.com group"
      entity1_index: "aol advertising.com group/company"
      verb: "include"
      verb_category: "generic relations"
      entity2: "advertising.com"
      entity2_index: "advertising.com, inc./company"
      event_type: "Fact"
      time_start: "2011-06-01"
      assoc_sig: 126.6919059017276,
      doccount: 4
      time_end: "2011-06-27"
    },
    //etc
  ]
}
```

"Events" and "Summaries" with the same time range (ie "time_start", "time_end" pair) are aggregated, with "doccount" used to store the sum. For "Facts", the "time_start" and "time_end" are set to the newest and oldest dates in which the "Fact" occurs (ie this may give some useful time range over which it is being discussed), with "doccount" counting all instances of the "Fact" within that time range.

Aggregation formats

Aggregation formats

```
{
  "output": {
    "aggregation": {
      // Geo-spatial/temporal aggregations (all of these default to 0):
      "geoNumReturn": integer,
      "timesInterval": string,
      // Entities, events, facts:
      "entsNumReturn": integer,
      "eventsNumReturn": integer,
      "factsNumReturn": integer,
      // "Moments", temporal aggregation for entities:
      moments: { ... }
      // Source information:
      "sources": integer,
      "sourceMetadata": integer // (includes both tags and types)
      // Raw Elasticsearch "facets":
      "raw": string // (see below)
    }
  }
}
```

The configuration of aggregation outputs is relatively simple, but this section also covers the different output formats:

- [Geo-spatial](#)
- [Temporal](#)
- [Entities](#)
- [Events and facts](#)
- [Moments](#)
- [Sources and source metadata](#)
- [Raw access to Elasticsearch "facets"](#)

Note that a (near-) future release will provide a more generic and powerful aggregation interface, allowing various document, entity, event, and metadata properties to be aggregated over time, space, and frequency.

Geo-spatial aggregation

Geo-spatial aggregation configuration

```
{
  "output": {
    "aggregation": {
      "geoNumReturn": integer,
    }
  }
}
```

All of the "[type]NumReturn" fields simply configure the number of entries returned (in the case of "geo", (lat,long) pairs), in order of frequency in the query-matching dataset.

The output format and an example is show below:

Geo-spatial aggregation output format

```
{
  //...
  "geo": [
    {
      "type": string, // the ontology type - see below
      "lat": number,
      "lon": number,
      "count": integer // the number of occurrences
    }
  ],
  "maxGeoCount": number
  //...
}
```

Geo-spatial aggregation example

```
//curl -XGET
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?qt[0].etext=%
22*%22&input.tags=%22topic:technology%22&output.aggregation.geoNumReturn=100&output.do
cs.enable=false'
{
  response: {
    action: "Query"
    success: true
    message: "((*))"
    time: 296
  },
  stats: {
    found: 53314
    start: 0
    maxScore: 0
    avgScore: 0
  },
  geo: [
    {
      type: "city"
      lat: 37.77499996125698
      lon: -122.4183003231883
      count: 1494
    },
    {
      type: "point"
      lat: 47.60639989748597
      lon: -122.3308002948761
      count: 442
    },
    {
      type: "geographicalregion"
      lat: 37.441899944096804
      lon: -122.14190002530813
      count: 206
    },
    //(etc)
  ],
  maxGeoCount: 9910
}
```

The "maxGeoCount" field in the top-level response is simply the highest count that occurs in the list (which is ordered by the underlying [geohash](#) used to store the lat/long). This can be used to calculate scaling factors without first having to traverse the return array.

The "type" field (ontological type) is discussed under the [Geo JSON format](#).

A typical use of the "geo" aggregation is to show heatmaps: in this case a "geoNumReturn" value of at least 1000 is recommended for large datasets.

Temporal aggregation

Temporal aggregation configuration

```
{
  "output": {
    "aggregation": {
      "timesInterval": string,
    }
  }
}
```

Temporal aggregation has a different configuration parameter to the others. Instead of specifying a number of entries to return, a string specifies the interval over which a document count is to be summed. This is in the standard format: "N[hdwmy]" ie an integer followed by h (hour), d (day), w (week), m (month), y (year).

- (Note that if "m" for month is the interval unit, then the aggregation is always performed over 1 month intervals, regardless of the "N")

The output format is very simple:

Temporal aggregation output format

```
{
  //...
  "times": [
    {
      "time": long,
      "count": integer // the number of occurrences
    }
  ],
  "timeInterval": long,
  //...
}
```

The "time" field in the "times" array is the start time of the interval in "ms" Unix time (milliseconds after 1970). The top-level "timeInterval" is the duration of that interval in ms, ie each interval can be expressed as ["times.time", "times.time"+"timeInterval"].

Even though the document counts are sorted by time rather than by "count", unlike geo-spatial aggregation no maximum count is provided. This is part oversight, part because it is not so (performance) critical to know the scaling factors in advance, but it will probably be corrected in a future release.

Temporal aggregation example

```
//curl -XGET
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?qt[0].etext=%
22*%22&input.tags=%22topic:technology%22&output.aggregation.timesInterval="1w"&output.
docs.enable=false'
{
  response: { ... },
  stats: { ... },
  times: [
    {
      time: 977702400000
      count: 2
    },
    {
      time: 993427200000
      count: 1
    },
    {
      time: 1041206400000
      count: 3
    },
    //etc
  ],
  timeInterval: 604800000
}
```

See also the more granular per-entity temporal aggregation available using "moments".

Entity aggregation

Entity aggregation preserves the format of the "entities" sub-objects of the [document](#), but across all documents in the query-matching dataset.

- (In fact due to implementation limitations, currently only the top "score.numAnalyze" eg 1000 documents are used to generate the entity aggregations)

Entity aggregation configuration

```
{
  "output": {
    "aggregation": {
      "entsNumReturn": integer,
    }
  }
}
```

It is worth noting that entities are returning in descending significance order (the other sorted aggregation types such as "geo" and "events" are ranked by frequency and are actually generated from the entire matching dataset, rather than a subset). A future release will try to standardize use of frequency vs significance, and also remove the use of subsets where possible.

The entity output format is identical to the [entity sub-object described here](#), except that the per-document fields ("significance" and "frequency") are replaced with the maximum per-document values in the matching sub-set, and some other fields ("actual_name", "relevance", "sentiment" are not present).

Entity aggregation example

```
//curl -XGET
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?qt[0].etext=%
22*%22&input.tags=%22topic:technology%22&output.aggregation.entsNumReturn=10&output.do
cs.enable=false'
{
  response: { ... },
  stats: { ... },
  entities: [
    {
      dimension: "Who"
      disambiguated_name: "LulzSec"
      doccount: 35
      frequency: 26
      index: "lulzsec/organization"
      totalfrequency: 348
      type: "Organization"
      significance: 6.874384562114902
      datasetSignificance: 6.001782525850459
      queryCoverage: 0.03835649052289408
      averageFreq: 0.054
    },
    {
      dimension: "Who"
      disambiguated_name: "Oracle Corporation"
      doccount: 827
      frequency: 20
      index: "oracle corporation/company"
      linkdata: [
        http://d.opencalais.com/er/company/ralg-trlr/eab9bfaa-47f1-368a-a9b7-a87bb345cf30
      ]
      totalfrequency: 2541
      type: "Company"
      significance: 9.882910950748887
      datasetSignificance: 5.5199530144758615
      queryCoverage: 0.8277825954323541
      averageFreq: 0.076
    },
    //etc
  ]
}
```

Good "entsNumReturn" values vary with application. For a document set that will not be filtered, 100 is a good value. For "recommendation" displays (eg "Other entities you may be interested in"), as few as 5-10 works fine. For larger datasets where the user will filter down from the initial return set then 1000+ is recommended.

See also the more granular per-entity temporal aggregation available using "moments".

Event and fact aggregation

As described under their [format specification](#), events are split into 3 categories:

- "Events": link multiple entities (via "entity1_index", "entity2_index", "geo_index") and represent a transient activity (eg travel)
- "Facts": link multiple entities like "Events" but represent (transient or permanent) relationships (eg being president)
- "Summaries": generally link 1 entity to a free text (eg a quotation: "Obama says...").

Summaries cannot currently be aggregated (except manually or via the "output.docs.eventsTimeline" function), because of (surmountable but non-trivial) implementation issues, combined with its perceived low priority. It is unclear whether it will get added in the future.

The configuration format is straightforward. As described under [entities](#), events and facts are ranked by frequency not significance (but this is likely to be an option in the future).

Event/fact aggregation configuration

```
{
  "output": {
    "aggregation": {
      "eventsNumReturn": integer,
      "factsNumReturn": integer,
    }
  }
}
```

Similar to [entities](#), the event/fact output format is essentially the same as the [Documents and their sub-objects \(entities, associations, user metadata, aggregations\)#Eventdocument](#) sub-object format, although fewer fields are populated:

Event/fact aggregation output format

```
{
  //...
  "events": [
    {
      "event_type": "Event",
      "entity1_index": string,
      "verb_category": string,
      "entity2_index": string,
      "geo_index": string,
      "assoc_sig": number, // A significance score for the association object (see below)
      "entity1_sig": number, // A significance score for entity1, if present
      "entity2_sig": number, // A significance score for entity2, if present
      "geo_sig": number, // A significance score for the geo, if present
      "doccount": integer // the number of occurrences
    }
  ],
  "facts": [
    {
      "event_type": "Fact",
      "entity1_index": string,
      "verb_category": string,
      "entity2_index": string,
      "geo_index": string,
      "assoc_sig": number, // A significance score for the association object (see below)
      "entity1_sig": number, // A significance score for entity1, if present
      "entity2_sig": number, // A significance score for entity2, if present
      "geo_sig": number, // A significance score for the geo, if present
      "doccount": integer // the number of occurrences
    }
  ],
  //...
}
```

Event/fact aggregation example


```
//curl -XGET
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?qt[0].etext=%
22*%22&input.tags=%22topic:technology%22&output.aggregation.eventsNumReturn=2&output.a
ggregation.factsNumReturn=2&output.docs.enable=false'
{
  response: { ... },
  stats: { ... },
  events: [
    {
      event_type: "Event"
      entity1_index: "microsoft corporation/company"
      verb_category: "acquisition"
      entity2_index: "skype technologies s.a./company"
      assoc_sig: 13.454
      entity1_sig: 12.5
      entity2_sig: 14.5
      doccount: 69
    },
    {
      event_type: "Event"
      entity1_index: "google inc./company"
      verb_category: "product release"
      entity2_index: "google+/product"
      assoc_sig: 35.123324
      entity1_sig: 10.5123
      entity2_sig: 94.235435
      doccount: 14
    }
  ],
  facts: [
    {
      event_type: "Fact"
      entity1_index: "google inc./company"
      verb_category: "company product"
      entity2_index: "android/product"
      assoc_sig: 23.34546
      entity1_sig: 10.5123
      entity2_sig: 54.235576
      doccount: 244
    },
    {
      event_type: "Fact"
      entity1_index: "tom XXX/person"
      verb_category: "person email address"
      entity2_index: "tXXX5@bloomberg.net/emailaddress"
      assoc_sig: 43.234324
      entity1_sig: 43.234324
      entity2_sig: 0
      doccount: 72
    }
  ]
}
```

Note that "time_start" and "time_end" are aggregated out of the object, ie all time information is lost. To aggregate events over time, use "output.d

[ocs.eventsTimeline](#)". It is likely at some point that the two formats will be combined somehow. At present, "output.aggregation.eventsNumReturn" and "output.aggregation.factsNumReturn" is best used with "link analysis" style applications, and "output.docs.eventsTimeline" is best used for timeline style applications.

Moments: per entity temporal aggregation

The "moments" function allows the entity aggregation to be combined with the "times" aggregation, generating a list of time periods in which named entities were mentioned, together with counts of the mentions for each time period.

 Coming versions of the platform will enhance this capability further, eg providing aggregated sentiment for the named entities.

Moments: per entity temporal aggregation

```
{
//...
"output": {
//...
"aggregation": {
//...
"moments": {
"timesInterval": string, // the time period over which the values are aggregated -
same format as "output.aggregation.timesInterval"
"entityList": [ string ] // A list of entity indexes, eg "barack obama/person"
}
//...
}
//...
}
//...
}
```

Note that if no "output.aggregation.moments.timesInterval" is set, then the time will be taken from "output.aggregation.timesInterval" if available, and defaulted to 1 month otherwise.

Note that the entityList respects [aliases](#).

Example moments request/reply

REQUEST (fragment)

```
{
  "moments": {
    "timesInterval": "1m",
    "entityList": [ "barack obama/person", "mitt romney/person" ]
  }
}
```

REPLY (fragment)

```
{
  "moments": {
    "barack obama/person": [
      {
        "time": 1346457600000,
        "count": 1
      },
      {
        "time": 1351728000000,
        "count": 2
      },
      {
        "time": 1354320000000,
        "count": 1
      },
      {
        "time": 1356998400000,
        "count": 9
      }
    ],
    "mitt romney/person": [
      {
        "time": 1346457600000,
        "count": 1
      },
      {
        "time": 1351728000000,
        "count": 1
      }
    ]
  }
}
```

Sources and source metadata aggregation

It can often be useful to understand what sources/source categories documents are being returned from. Infnite allows the following aggregations:

- Individual sources (using the "sourceKey" field of the [document object](#), ie the "key" field of the "source" object)
- Source types (using the "mediaType" field of the [document object](#), ie the "mediaType" field of the "source" object)
- Source tags (using the "tags" field of the [document object](#), ie the "tags" field of the "source" object)
 - There is currently no concept of "per document" tags (eg auto-generated from the document content), though there may be in the future.
 - Note that the Infnite "system collection" contains two "top level" tags, "topic:<tag>" and "industry:<tag>" (its general, or "content" tags tend to be quite low level and difficult to use)

The first of these is configured by "output.aggregation.sources", the second two by "output.aggregation.sourceMetadata":

Source/source metadata aggregation configuration

```
{
  "output": {
    "aggregation": {
      "sources": integer,
      "sourceMetadata": integer,
    }
  }
}
```

The output format is very simple, arrays "sources", "sourceMetaTags", and "sourceMetaTypes", with fields "term" (string) and "count" (integer). Examples:

Event/fact aggregation example

```
//curl -XGET
'http://infinite.ikanow.com/api/knowledge/query/4c927585d591d31d7b37097a?qt[0].etext=%
22*%22&input.tags=%22topic:technology%22&output.aggregation.sources=5&output.aggregati
on.sourceMetadata=5&output.docs.enable=false'
{
  "response": {
    "action": "Query",
    "success": true,
    "message": "((*)",
    "time": 140
  },
  "stats": {
    "found": 53560,
    "start": 0,
    "maxScore": 0,
    "avgScore": 0
  },
  "sources": [
    {
      "term": "feed:..origin.feeds.pheedo.com.bw.technology_news-rss",
      "count": 11182
    },
    {
      "term": "http.gizmodo.com.index.xml",
      "count": 3102
    },
    {
      "term": "http.www.reddit.com.r.technology..rss",
      "count": 2391
    },
    {
      "term": "feed:..origin.feeds.pheedo.com.bw.energy_news-rss",
      "count": 2390
    },
    {
      "term": "http.www.engadget.com.rss.xml",
      "count": 1687
    }
  ],
}
```

```
"sourceMetaTags": [
  {
    "term": "topic:technology",
    "count": 53560
  },
  {
    "term": "news",
    "count": 44708
  },
  {
    "term": "industry:technology",
    "count": 38355
  },
  {
    "term": "technology",
    "count": 30871
  },
  {
    "term": "industry:all",
    "count": 15205
  }
],
"sourceMetaTypes": [
  {
    "term": "News",
    "count": 53281
  },
  {
    "term": "Video",
    "count": 279
  }
]
```

```
}
]
}
```

Raw access to Elasticsearch "facets"

Finally, like for [queries](#), there is the option simply to pass an arbitrary "facet" (ie aggregation) through to [ElasticSearch](#), using its raw API. Like for queries, this functionality should be considered an absolute last resort.

Unlike for queries, where the raw Elasticsearch query is specified as a JSON object, raw facets are specified as a string conversion of the JSON object (this may change in a future release), example:

Raw aggregation configuration example

```
{
  "output": {
    "aggregation": {
      "raw": "{\"sources\":{\"terms\":{\"field\":\"sourceKey\",\"size\":5}}}"
    }
  }
}
```

The above example is functionally equivalent to specifying `&output.aggregation.sources=5`, except that the output would be in the array `"facets.sources"` instead of the top-level sources, eg:

Raw aggregation configuration output example

```
{
  "response": {
    "action": "Query",
    "success": true,
    "message": "((*)",
    "time": 140
  },
  "stats": {
    "found": 53560,
    "start": 0,
    "maxScore": 0,
    "avgScore": 0
  },
  "facets": {
    "sources": [
      {
        "term": "feed:..origin.feeds.phedo.com.bw.technology_news-rss",
        "count": 11182
      },
      {
        "term": "http.gizmodo.com.index.xml",
        "count": 3102
      },
      {
        "term": "http.www.reddit.com.r.technology..rss",
        "count": 2391
      },
      {
        "term": "feed:..origin.feeds.phedo.com.bw.energy_news-rss",
        "count": 2390
      },
      {
        "term": "http.www.engadget.com.rss.xml",
        "count": 1687
      }
    ],
    // +Any other "raw" facets specified
  }
}
```

Note finally that, like for queries, specifying any facets overrides any Infinit.e aggregations (except currently for [entities](#), though this may change).

Filtering

Filter format

```
{
  "output": {
    "filter": {
      "entityTypes": [ string ], // A list of (case sensitive) entity types - if specified
      non-matching entities and associations and documents will be discarded
      "assocVerbs": [ string ] // A list of (case sensitive) verb categories - if
      specified non-matching associations and documents will be discarded
    }
  }
}
```



Either of the above filters can be made "negative" by inserting a "-" in front of the *first* entry in the array. Negative filtering simply removes all entities or associations that match the filter from the document (and also their score). Note that queries can still match on negatively filter entities and associations.

Examples:

Filter format - examples

```

//
// Twitter example: only pull back hashtags and twitter handles from tweets (and
// discard documents and associations not containing either)
//
{
  "output": {
    "filter": {
      "entityTypes": [ "HashTag", "TwitterHandle" ]
    }
  }
}
//
// Negative twitter example: remove all keywords and locations extracted for the tweet
//
{
  "output": {
    "filter": {
      "entityTypes": [ "-Keyword", "Location" ]
    }
  }
}

//
// Twitter example: pull back all entities, but only tweets that are retweets (and
// only retweet associations)
//
{
  "output": {
    "filter": {
      "assocVerbs": [ "retweets" ]
    }
  }
}
//
// Twitter example: this will discard all associations (because "retweets" are always
// associations between TwitterHandle types)
//
{
  "output": {
    "filter": {
      "entityTypes": [ "Keyword" ],
      "assocVerbs": [ "retweets" ]
    }
  }
}
//
// Business acquisition example
//
{
  "output": {
    "filter": {
      "entityTypes": [ "Company", "Organization" ],
      "assocVerbs": [ "acquires" ]
    }
  }
}

```

There is one implementation issue that is noteworthy: the association verb category is stored in such a way that subsets of the phrase will match. For example, "generic" or "relations" will match on "generic relations", and "generic relations" will match on "generic relations (special case)".

Knowledge - Query - Query Terms

Overview of querying

Within the top level query JSON object there is a query field "qt" that is an array of query term objects. Query term objects are described below and allow the following query types:

- Exact text
- Free text
- Entities
 - Entity Aliasing
- Geospatial
- Temporal
- Entity associations
- Raw Elasticsearch queries

These terms can then be combined in an arbitrary boolean expression (with the operators AND, OR, NOT and parantheses) using the (case insensitive) "logic" field of the top level object, where the different terms are denoted by their index in the array (counting from 1). For example:

Example top level query

```
{
  "qt": [ { term1 }, { term2 }, { term3 }, { term4 } ],
  "logic": "1 AND (2 OR 3) AND NOT 4"
}
```

In the above example each of term1-term4 is one of the objects described below.

If the logic term is set to null or not present, it defaults to ANDing all the terms together.

In the "dot notation" used to represent query objects as URL parameters in "GET" requests, the different "qt" terms are represented as "qt[0]", "qt[1]", etc (ie indexed from 0, unlike the "logic" string).

Finally, note that the combination of "qt" and "logic" can be replaced by the "raw" object described at the bottom of this page, which gives the user access to the raw [ElasticSearch query API](#). If both "qt"/"logic" and "raw" are present, the "qt"/"logic" fields are ignored.

Exact text

The exact text query term object has the following format:

Free text format

```
{
  "etext": string
}
```

The "etext" string is a phrase that must match exactly somewhere in the document (in any of the text fields). There is one exception: if "*" is the "etext" field then it matches all documents.

For example (using dot notation), qt[0].etext="barack obama" will match on documents containing "barack obama" but not documents containing only (eg) "president obama", "barack 'barry' obama" etc.

Free text

The free text query term object has the following format:

Free text format

```
{
  "ftext": string
}
```

The "ftext" field represents an arbitrary Lucene query ([Lucene syntax](#), including [elasticsearch extensions and modifications](#)). By default, all text fields in the [document \(including its entities and events; link to the document format\)](#) are included in the query, though the standard "field:text" syntax can be used.

For example, { "qt": ["ftext": "barack obama"] } will match on any documents containing either "barack" or "obama", with documents containing both [scored more highly](#). { "qt": ["ftext": "+barack +obama"] } requires both be present (but not necessarily in the same phrase), and { "qt": ["ftext": "barack obama"] } is equivalent to the "etext" query described above.

Other examples:

- qt[0].ftext="+obama -palin": documents containing the word "Obama" but not containing the word "Palin"
- qt[0].ftext="title:\\"palin\\" ": documents with the word "Palin" in the title.

Note when using dot notation and typing queries directly into the URL bar that characters like '+' must be double-URL-encoded, eg to %25%32%4B (ie via %2B from +).

Entities

The entity query term object has the following 2 possible formats:

Entity format

```
{
  // EITHER
  "entity": string,
  //OR
  "entityValue": string, // (entityValue is mandatory, entityType is optional)
  "entityType": string,
  // AND OPTIONALLY
  "entityOpt": { // (optional, see below for demos)
    "expandAlias": boolean, // (optional, defaults to false if not present)
    "rawText": boolean // (optional, defaults to false if not present)
  },
  "sentiment": { // (optional, specify one or both of min/max, see below)
    "min": number,
    "max": number
  }
}
```

In the first instance the "entity" string is in the format "entityValue/entityType" (this is its "index" form, eg "index" in the [Entity JSON object](#)).

In the second, decomposed, instance either of "entityValue" or "entityType" can be left out (in the first case this would match on all entities of a given type; in the second case, it would match on all entity names regardless of the type).

The optional "entityOpt.expandAlias" boolean term will allow matching not just on the entity but also on common, automatically extracted, "aliases". This will tend to have the effect of matching on more documents, some of which will be false positives however. This query type is also slower. Note this is different to manual entity aliasing, described [here](#).

The optional "entityOpt.rawText" boolean term adds the entity's disambiguated name as an exact text query - this can be useful when some sources have low quality entity extraction (eg are in foreign languages, or are in list format etc), since any instance of the name appearing in a page will result in that page's selection.

Some examples:

- qt[0].entity="facebook/company": *will match on documents containing references to the company Facebook, but not the*

technology.

- `qt[0].entityValue="facebook"&qt[0].entityType="company"`: *equivalent to the above*
- `qt[0].entityValue="facebook"`: *will match on both uses of the term Facebook*
- `{ "qt": [{ "entity": "barack obama/president", "entityOpt": { "expandAlias": true } }] }`: *will match on documents containing references to Barack Obama, but also other common text strings such as "Barry Obama", "President Obama" etc.*



Manual entity aliasing is supported and is described [here](#).

Sentiment

Entity queries can be combined with sentiment, using the "sentiment" json described above. If the "entity" or "entityValue" field is specified, then only documents containing that entity with a sentiment field that exists and is in the specified range. If neither of the text fields are specified then only documents containing 1+ entities with sentiment are selected.

Geospatial

The geospatial query term has the following possible formats:

Geospatial format

```
{
  "geo": {
    "centerll": string,
    "dist": string,
    "ontology_type": string // optional, see below
  }
}
//or
{
  "geo": {
    "minll": string,
    "maxll": string
    "ontology_type": string // optional, see below
  }
}
```

In the first case, the user is specifying the center latitude ("centerll") and longitude pair and radius ("dist") of a circle.

In the second case, the user is specifying a bounding box via the "minll" (lowest lat and long values ie the "bottom left") and "maxll" (highest lat and long values, ie the "top right").

In all cases the lat/long values are represented as strings either as "<lat>,<long>" or "<lat>,<long>" (ie the same but without parantheses).

The "dist" string is a distance in the format "<distance><unit>" where <distance> is an integer or floating point number, and unit is one of "m" (miles), "km" (kilometers), "nm" (nautical miles).

In both cases, an optional "ontology_type" can be specified. If it is specified, then entities with an higher "ontology_type" and ignored: see [geo](#) discussion for more details.

Examples:

- `qt[0].geo.centerll="40.12,-71.34"&qt[0].geo.dist="100km"`: *within 100km of the specified lat/long.*
- `{ "qt": [{ "geo": { "centerll": "40.12,-71.34", "dist": "100" } }] }`: *uses the default unit (km), ie is the same query as above.*
- `qt[0].geo.minll="(4.1,-171.34)"&qt[0].geo.maxll="40.12,-71.34"`: *bounding box showing lat/long format with and without parantheses.*

Temporal

The temporal query term has the following format:

Temporal format

```
{
  "time": {
    "min": string,
    "max": string
  }
}
```

One of "min" and "max" must be specified. If one of them is not specified, time is not bounded in that direction (eg if "min" is not specified then it means "all times before max"; if "max" is not specified then it means "all times after min").

The date fields are both strings and support a number of different formats:

- "now" which always resolves to the current time,
- any Unix time (ie **milliseconds** after "Jan 1 00:00:00 1970"),
- and the following date/date-time formats: "yyyy'-DDD", "yyyy'-M'-dd", "yyyyMMdd", "dd MMM yyyy", "dd MMM yy", "MM/dd/yy", "MM/dd/yyyy", "MM.dd.yy", "MM.dd.yyyy", "dd MMM yyyy hh:mm:ss", "yyyy-MM-dd" (ISO Date), "yyyy-MM-ddZZ" (ISO Date-Timezone), "yyyy-MM-dd'T'HH:mm:ssZZ" (ISO DateTime-Timezone), "EEE, dd MMM yyyy HH:mm:ss Z" (SMTP DateTime).

Examples:

- { "qt": [{ "time": { "min": "1284666757164", "max": "now" } }] } : *from 16 Sep 2010 until now.*
- qt[0].time.min="now": *any time in the future.*
- qt[0].time.max="20100201": *any time before 1 Feb 2010.*
- { "qt": [{ "time": { "min": "02/10/2000", "max": "10 Feb 2001 13:00:00" } }] } : *from 10 Feb 2000 until 10 Feb 2001 at 1pm.*

Associations

The association query format is slightly more complex than the others. It is also slightly more limited.

The association format is as follows:

Event format

```
"assoc": {
  "entity1": { ... }, // the "subject"; can be ftext, etext, or
entity/entityValue/entityType query terms
  "entity2": { ... }, // the "object"; can be ftext, etext, or
entity/entityValue/entityType query terms

  "verb": string,

  "geo": { ... }, // geo query term
  "time": { ... }, // time query term

  "type": string // "Event", "Fact", or "Summary"
},
"sentiment": { // (optional, specify one or both of min/max, see below)
  "min": number,
  "max": number
}
```

As can be seen from the above code block, the association query term is a composite of other query term types ([free text](#), [exact text](#) and [entity terms](#) for "entity1" and "entity2"; also [temporal](#) and [geospatial](#)).

Some things to note while performing entity queries:

- The "entity1" field is processed as follows:
 - "ftext" and "etext" terms are applied across both the "entity1" and "entity1_index" fields within the [entity object](#).

- entity/entityValue/entityType terms are only applied to the "entity1_index" field
- The "entity2" field is processed analogously
- The "verb" string is applied as an exact text query to the "verb_category" field and a free text query to the "verb" field within the *entity object*
- For events with a time range ("time_start" and "time_end" fields), any part of the event time range can match the "time" term.
- The difference between "Events", "Facts" or "Summaries" is described [here](#).
- If multiple terms are specified then these are ANDed together. There is currently no way of performing more complex boolean equations on individual events (obviously multiple event query terms can be specified and match across all events within a document).
- If sentiment is specified then only documents containing associations with a sentiment field (this is somewhat rare) that exists and is in the selected range are selected.
- Event queries with multiple terms can be a bit slower than other queries (due to its implementation in Elasticsearch).

Example event queries

```
// Any fact in which Barack Obama is the subject:
{
  "assoc": {
    "entity1": {
      "entity": "barack obama/person"
    },
    "type": "Fact"
  }
}
// Travel associations involving Sarah Palin:
{
  "assoc": {
    "entity1": {
      "entityValue": "sarah palin",
      "entityType": "person"
    },
    "verb": "travel",
  }
}
// Events in the future:
{
  "assoc": {
    "time": {
      "min": "now"
    },
    "type": "Event"
  }
}
```

Combining query terms

Multiple query terms can be combined in 2 ways:

- Using the "logic" field as described [above](#) under "Overview of querying". This is the standard way of combining separate queries.
- In addition, within a single query term multiple elements of different types can be merged into a single object - this has the effect of ANDing them together. For example:
 - { "qt": [{ "entity": "barack obama/person", "time": { "min": "1284666757164", "max": "now" } }] } : *documents containing the entity Barack Obama, from 16 Sep 2010 until now.*
 - qt[0].etext="apple"&qt[0].ftext="pair" : *this is equivalent to qt[0].etext="apple"&qt[1].ftext="pair" &logic="1 and 2"*

Raw Elasticsearch queries

At present [ElasticSearch](#) is used as the front end of the search engine.

The system provides a "passthrough" interface for full (or "raw") Elasticsearch queries. Obviously this capability is for advanced use only, and should be avoided where possible.

The Elasticsearch API and Query DSL is described in detail on their web-site, and it is beyond the scope of this documentation to go into any more details. Only the "search" call and the "facet" call (accessed from the "aggregation" object) are available.

The search call can be placed inside the "raw" object from the top level as follows:

ElasticSearch passthrough syntax and example

```
// Syntax:
{
  "raw": {
    // Put fields and objects from the top level Elasticsearch "query" object here
  }
}
// Example:
{
  "raw": {
    "match_all": {}
  }
}
```

Some things to be aware of when making raw queries:

- "raw" queries overrides other query terms - so only the "raw" query is performed.
- All other aspects of the query are still take from the URL/JSON Object, ie:
 - Only documents from the specified communities and the specified inputs are processed
 - Once documents matching the "raw" query have been retrieved by the server, they are ranked according to the "score" object
 - The format and number of documents returned to the client are determined by the "output" object.
- It is not currently possible to specify both "raw" queries and "raw" aggregations (facets) within a single query.

Knowledge - Query - Scoring Parameters

Overview

In order to understand the scoring parameters presented by the Infinit.e API, it is necessary to have a basic understanding of the query and scoring process:

- The user query is turned into an Elasticsearch query and applied across the cluster.
- The number of documents returned from Elasticsearch is capped at a "large" number (default 1000, eg 10x the documents to return). The documents are ordered by their [Lucene score](#) (or optionally just by descending date).
- Each returned document is then assigned a [Significance score](#) as described below.
- The significance and relevance scores are then normalized against each other based on a relative importance specified by the user (default 2:1 in favor of significance) and combined, with the mean score set to 100 (like the "+" stats in baseball, eg 120 is 20% higher than average).
 - (All three scores are attached to the documents, as "queryRelevance", "aggregateSignif", and "score" respectively)
- The top scoring [documents](#) or [entities](#) are returned to the client.

Significance

It is beyond the scope of this documentation to go into much detail about significance, but this section provides a brief description (a 1-line summary is also provided below!):

- Each [document](#) has a set of [entities](#)
- For each entity,document pair, a [TF-IDF](#) score is generated, the [entity's "significance"](#). This score is adjusted in a number of ways:
 - Entities with low document counts have their significance suppressed by 33% (well below a dynamically calculated "noise floor") or 66% (just below/at the "noise floor")
 - When only a subset of the matching documents are returned (eg > 1000 documents), the significance is adjusted to estimate the TF-IDF across the entire matching dataset, not just the returned subset.
- The [document significance](#) ("[aggSignificance](#)") is the sum of the entity,document significances. This score can be adjusted in a number of ways:
 - Temporally, using the document's "publishedDate" field and a standard "[decay algorithm](#)"
 - Geo-spatially, similarly to the above but based on distance using the closest entity to the decay origin (lat,long).
- Entities are also assigned a "[datasetSignificance](#)", which is just the average of the significances across all documents in which it appears.
 - Note that neither entity scores are currently adjusted for time or geo-spatial decay, though this will be added as an option in a future release.

Separate documentation will describe the scoring algorithms in more detail.

In summary: **relevance** measures how well a *document* matches the user's query, **significance** measures how well an *entity* matches the user's query (and **document significance** is simply the sum of the entity significances).

Scoring parameters

All scoring parameters are maintained under a "score" object under the top level query. The remainder of this section describes the "score" object's fields.

Scoring parameters - "numAnalyze"

```
{
  "score": {
    "numAnalyze": integer, // (default: 1000)
    "scoreEnts": boolean, // (default: true)
    // See following sections for other parameters
  }
}
```

The "numAnalyze" parameter dictates the maximum number of documents to be returned from the Lucene (/ElasticSearch) query and analyzed according to the significance algorithm described above. The larger the number, the more accurate the results but the slower the query.

Empirically, the default of 1000, which takes 0.5-1 second has produced good results.

Note that this parameter is also currently used to determine how many documents are used to generate the ["event timeline"](#).

If "scoreEnts" is set to false (defaults to true if not present), then entities do not have significance scores generated.

- *This can be useful in cases where documents are not being scored using significance either (see "sigWeight" field below) - in this case rather than retrieving "numAnalyze" documents, only "output.docs.numReturn" can be retrieved, which is much faster.*

Scoring parameters - significance/relevance weighting

```
{
  "score": {
    // See preceding sections for other parameters
    "sigWeight": number, // (default: 0.67)
    "relWeight": number, // (default: 0.33)
    "adjustAggregateSig": boolean, // (default: auto-decide, see below)
    // See following sections for other parameters
  }
}
```

These two floating point numbers represent the relative weight of significance vs relevance (as described above). If they don't sum to 1, they are just divided by their sum.

Increasing the "sigWeight" field tends to return documents that are longer and don't necessarily strongly relate to the user's query; instead they will tend to return documents that discuss concepts particular to the query.

Increasing the "relWeight" field tends to return documents that are shorter and very strongly relates to the user's query.

- *(Eg for a query on "american politics", The most significant documents would contain discussion of Obama, Palin etc; the most relevant documents would contain the words "american" and "politics" with high frequency compared to other words)*

If one of the two weights is set to 0 then its score is neither calculated nor used.



If both weights are set to 0 then documents are ranked in descending date order and no scoring is performed.

If "adjustAggregateSig" is set to true (the default is described below), then aggregation significances (ie in entities and associations) are adjusted for the average relevances of the documents containing them. This is useful in cases where free text queries are used, eg "ftext": "bob smith" will match on documents only containing "bob" or only containing "smith", but with lower relevance scores - so entities contained in such documents should be weighted down accordingly.

By default, this is enabled automatically if "ftext" terms are used, and otherwise disabled. The reason for this is that in other cases when relevance may vary widely, eg chains of "etext" queries linked with ORs, it is not normally desirable to adjust the entities (in the example given, it is likely to be a list of aliases: scoring up documents that contain many aliases via the relevance term is fine, but there is no reason to score up the individual entities contained in the documents.)

Scoring parameters - manual significance weighting

```
{
  "score": {
    // See preceding sections for other parameters
    sourceWeights: { string: double },
    typeWeights: { string: double },
    tagWeights: { string: double },
    // See following sections for other parameters
  }
}
```

The 3 fields listed above allow users to adjust document scores manually. This can be useful when, for example, a source that is known to contain relatively low intelligence documents scores high because each document has many common entities, particularly when those entities are unique to the source (hence queries involving that source will tend to give those entities a higher significance).

The weights are in formats like this:

```
{
//..
  "sourceWeights": {
    "washingtondc.IncidentReport": 0.5, // halve scores from the source with this key
    (source.key)
    "arstechnica.com.tech-policy.2012.10.last-android-vendor-f.147.4.": 1.5 // increase
    documents from this source by 50%
  }
  "typeWeights": {
    "Record": 2.0, // Any documents of source.type "Record" that don't match a source
    weight
    "Report": 0.75, //(etc)
  }
  "tagWeights": {
    "database": 1.5, // If a document has this tag, multiply its total score by 1.5
    "largeReport": 0.25 // (averaged across all matching tags)
  }
//...
}
```

The weights are applied as follows:

- First the source weights are applied.
- If no source weight matches the document, then the type weights are applied.
- If no type weight matches the document, then a tag weight is generated by averaging all matching entries from the "tagWeights" map.
- If no weight matches the document, then its total score is preserved.

Scoring parameters - time proximity

```
{
  "score": {
    // See preceding sections for other parameters
    "timeProx": {
      "time": string,
      "decay": string
    },
    // See following sections for other parameters
  }
}
```

"time" is the center point around which to decay. It has the same format as the "min" and "max" fields of the ["time" query term](#), ie "now", Unix time in ms, or one of a standard set of date/time formats (ISO, SMTP, etc).

"decay" is the "half life" of the decay (ie the duration from "time" at which the score is halved). It is in the format "N[dmwy]" where N is an integer and d,m,w,y denote "day", "month", "week" or "year" (eg "1w", "1m"; note currently if "m" is used, then the duration is always 1 month).

Scoring parameters - geospatial proximity

```
{
  "score": {
    // See preceding sections for other parameters
    "geoProx": {
      "ll": string,
      "decay": string
    }
  }
}
```

"ll" is the lat/long of the center point around which to decay. It has the same format as the "centerll"/"minll"/"maxll" fields of the [geospatial query term](#), ie "lat, long" or "(lat, long)".

"decay" is the "half life" of the decay (ie the distance from "ll" at which the score is halved). It is in the same format as the "dist" field of the [geospatial query term](#), ie in the format "<distance><unit>" where <distance> is an integer or floating point number, and unit is one of "m" (miles), "km" (kilometers), "nm" (nautical miles).

Example "score" object

```
{
  "score": {
    "numAnalyze": 1000,
    "sigWeight": 0.67,
    "relWeight": 0.33,
    "timeProx": {
      "time": "now",
      "duration": "1m"
    }
  }
}
```


`/knowledge/feature/aliasSuggest/{field}/{term}/{comma-separated-list-of-community-ids}`



Returns a list of alias suggestions (ie alternative names for an entity) that have been found in other documents, based on your query term.

Authentication

Required, see [Auth - Login](#)

Arguments

field (required)

The entity field on which to search, ie "index", "disambiguated_name", or "actual_name"

term (required)

The value of the entity field on which to search for aliases. Note this needs to be URL-escaped, eg "/" in the "index" would be represented by "%2F"

community id list (required)

The list of community IDs or [community ID - regex](#) in the URL controls which sources the query will use. Obviously the logged-in user must have access rights to those communities. A user's communities can be viewed using the ["person/get"](#) API call.

Example

<http://infinite.ikanow.com/api/knowledge/feature/aliasSuggest/index/barack>

[obama%2Fperson/4c927585d591d31d7b37097a,4db5c05fb246d25364aceca0](http://infinite.ikanow.com/api/knowledge/feature/aliasSuggest/disambiguated_name/barack)

http://infinite.ikanow.com/api/knowledge/feature/aliasSuggest/disambiguated_name/barack

[obama/4c927585d591d31d7b37097a,4db5c05fb246d25364aceca0](http://infinite.ikanow.com/api/knowledge/feature/aliasSuggest/disambiguated_name/barack)

http://infinite.ikanow.com/api/knowledge/feature/aliasSuggest/actual_name/Barry%20Obama/4c927585d591d31d7b37097a,4db5c05fb246d25364aceca0

Example Response



```
{
  "response": {
    "action": "aliasSuggest",
    "success": true,
    "message": "Successfully returned aliases",
    "time": 0
  },
  "data": [
    "BARACK OBAMA",
    "Barack Obama"
  ]
}
```

Knowledge - Feature - Association Suggest

`/knowledge/feature/assocSuggest/{entity1}/{verb}/{entity2}/{searchfield}/{communityids}`



Returns a list of entity-association suggestions based the sentence/query fragment passed in (eg what the user has typed so far into a query box). Only the specified term (subject/object/verb) from the matching associations is returned, ordered by document count across the specified communities.

Authentication

Required, see [Auth - Login](#)

Arguments

entity1 (required)

The first term (subject) of the event you are searching for, send "null" to match on all subject entities

verb (required)

The 2nd term (verb) of the event you are searching for, send "null" to match on all possible verbs

entity2 (required)

The 3rd term (object) of the event you are searching for, send "null" to match on all object entities

searchfield (required)

The field that you want to get suggestions for, can be "entity1", "verb", or "entity2". Note that currently only "indexed" entities (eg "disambiguated_name/type") are returned, and only the verb categories corresponding to verbs.

communityids (required)

A comma delimited list of community IDs or [community ID - regex](#) in the URL controls which sources the query will use. Obviously the logged-in user must have access rights to those communities. A user's communities can be viewed using the "person/get" API call.

Example

<http://infinite.ikanow.com/api/knowledge/feature/assocSuggest/obam/null/null/entity1/4c927585d591d31d7b37097a>

<http://infinite.ikanow.com/api/knowledge/feature/assocSuggest/obama/travel/null/entity1/4c927585d591d31d7b37097a>

<http://infinite.ikanow.com/api/knowledge/feature/assocSuggest/obam/traveled/can/entity1/4c927585d591d31d7b37097a>

<http://infinite.ikanow.com/api/knowledge/feature/assocSuggest/obam/traveled/can/entity1/4c927585d591d31d7b37097a,4c927585d561d31e7b37096b>

Example Response

```
{
  "response": {
    "action": "Association Suggestions",
    "success": true,
    "message": "obam",
    "time": 0
  },
  "data": [
    "mike del checcolo/person",
    "center on budget and policy priorities/organization",
    "trent gavazzi/person",
    "obama/person",
    "john nase/person",
    "mehmet oz/person",
    "michelle obama/person",
    "lopez howell/person",
    "dave booze/person",
    "earl perkins/person",
    "mark blyth/person",
    "kathleen o'loughlin/person",
    "larry lanzillotta/person",
    "barack obama/person",
    "obama administration/organization"
  ]
}
```

Knowledge - Feature - Entity Suggest

[/knowledge/feature/entitySuggest/{fragment}/{comma-separated-list-of-community-ids}?geo=\[true|1|false|0\]&linkdata=\[true|1|false|0\]](#)



Returns a list of entity suggestions based the sentence/query fragment passed in (eg what the user has typed so far into a query box).

Authentication

Required, see [Auth - Login](#)

Arguments

fragment (required)

One or more words (the last word can also be a fragment), which are "fuzzily" compared against known entities stored in Infnit.e

community ID list (required)

The list of community IDs or [community ID - regex](#) in the URL controls which sources the query will use. Obviously the logged-in user must have access rights to those communities. A user's communities can be viewed using the ["person/get" API call](#).

geo (optional)

By default, where ("Where") entities have lat/longs associated with them, they are returned by this REST call. Specify this parameter as "false" or "0" to discard them.

linkdata (optional)

If specified as "true" or "1", then any "linkdata" fields for the entity (ie references to Web services that form part of the [Linked Open Data Project](#)) are returned. By default they are not.

Example

<http://infinite.ikanow.com/api/knowledge/feature/entitySuggest/obam/4c927585d591d31d7b37097a>

<http://infinite.ikanow.com/api/knowledge/feature/entitySuggest/c/4c927585d591d31d7b37097a?geo=true&linkdata=true>

<http://infinite.ikanow.com/api/knowledge/feature/entitySuggest/hillary cli/4c927585d591d31d7b37097a,4c927585d561d31e7b37096b>

Example Response



```
{
  "response": {
    "action": "Suggestions",
    "success": true,
    "message": "c",
    "time": 0
  },
  "data": {
    "dimensions": [
      {
        "dimension": "Where",
        "value": "New York City,New York,United States",
        "type": "City",
        "geotag": {
          "lat": 40.7142,
          "lon": -74.0064
        },
        "ontology_type": "city",
        "linkdata": [
          "http://d.opencalais.com/er/geo/city/ralg-geo1/b3719a18-c511-51a8-b3f9-f8480b3b6e48"
        ]
      },
      {
        "dimension": "Where",
        "value": "California,United States",
        "type": "ProvinceOrState",
        "geotag": {
          "lat": 36.4885198674,
          "lon": -119.701379437
        },
        "ontology_type": "countrysubsidiary",
        "linkdata": [
```

```
"http://d.opencalais.com/er/geo/provinceorstate/ralg-geol/fd9e1e90-96a9-04ff-1397-445c4e93208b"
```

```
]
},
{
  "dimension": "Who",
  "value": "CEO",
  "type": "Position"
},
{
  "dimension": "What",
  "value": "cancer",
  "type": "MedicalCondition"
},
{
  "dimension": "Where",
  "value": "China",
  "type": "Country",
  "geotag": {
    "lat": 32.9042932784,
    "lon": 110.467708512
  },
  "ontology_type": "country",
  "linkdata": [
```

```
"http://d.opencalais.com/er/geo/country/ralg-geol/8a7d7ba2-88ca-0f0e-a1ec-f975b026e8e1"
```

```
]
},
{
  "dimension": "Where",
  "value": "Canada",
  "type": "Country",
  "geotag": {
    "lat": 56.757746527,
    "lon": -86.4195625771
  },
  "ontology_type": "country",
  "linkdata": [
```

```
"http://d.opencalais.com/er/geo/country/ralg-geol/c2aadac2-ca30-ca8a-adfb-e331ae180d28"
```

```
]
},
{
  "dimension": "Who",
  "value": "Securities and Exchange Commission",
  "type": "Organization"
},
{
  "dimension": "Where",
  "value": "Los Angeles,California,United States",
  "type": "City",
  "geotag": {
    "lat": 34.0522,
    "lon": -118.2428
  },
  "ontology_type": "city",
```

```

        "linkdata": [
"http://d.opencalais.com/er/geo/city/ralg-geol/9c147196-160d-8555-537a-06b783fe5
4af"
        ]
    },
    {
        "dimension": "Where",
        "value": "San Francisco,California,United States",
        "type": "City",
        "geotag": {
            "lat": 37.775,
            "lon": -122.4183
        },
        "ontology_type": "city",
        "linkdata": [
"http://d.opencalais.com/er/geo/city/ralg-geol/f56118da-0256-d097-ff73-1f5d09380
31b"
        ]
    },
    {
        "dimension": "Who",
        "value": "Congress",
        "type": "Organization"
    },
    {
        "dimension": "What",
        "value": "Jude Children?s Research Hospital",
        "type": "Facility"
    },
    {
        "dimension": "Where",
        "value": "Chicago,Illinois,United States",
        "type": "City",
        "geotag": {
            "lat": 41.85,
            "lon": -87.65
        },
        "ontology_type": "city",
        "linkdata": [
"http://d.opencalais.com/er/geo/city/ralg-geol/b0dfbe09-9b57-0fa7-e789-ccfc8a6f6
50a"
        ]
    },
    {
        "dimension": "Who",
        "value": "president and CEO",
        "type": "Position"
    },
    {
        "dimension": "Who",
        "value": "Microsoft Corporation",
        "type": "Company",
        "linkdata": [
"http://d.opencalais.com/er/company/ralg-tr1r/4eel3d20-b44f-3bce-98bc-49a303e72d
b5"
        ]
    }

```

```

    ],
    {
      "dimension": "Who",
      "value": "St. Jude Children?s Research Hospital",
      "type": "Organization"
    },
    {
      "dimension": "Who",
      "value": "Chairman",
      "type": "Position"
    },
    {
      "dimension": "Where",
      "value": "North Carolina,United States",
      "type": "ProvinceOrState",
      "geotag": {
        "lat": 35.4833648675,
        "lon": -79.4002284439
      },
      "ontology_type": "countrysubsidiary",
      "linkdata": [
        "http://d.opencalais.com/er/geo/provinceorstate/ralg-geol/56e11c06-6ad9-7041-8d6
        e-70a5eff06454"
      ]
    },
    {
      "dimension": "What",
      "value": "childhood cancer",
      "type": "MedicalCondition"
    },
    {
      "dimension": "Who",
      "value": "Apple Inc.",
      "type": "Company",
      "linkdata": [
        "http://d.opencalais.com/er/company/ralg-tr1r/23d07771-c50b-315b-8050-3cdaf47ac0
        d0"
      ]
    },
    {
      "dimension": "Who",
      "value": "National Cancer Institute-designated Comprehensive
      Cancer Center",
      "type": "Organization"
    }
  ]

```

```
}  
}
```

Knowledge - Feature - Geo Suggest - (NOT YET IMPLEMENTED)

`/knowledge/feature/geoSuggest/{fragment}/{comma-separated-list-of-community-ids}`

 Returns a list of geo suggestions based the sentence/query fragment passed in (eg what the user has typed so far into a query box).

Authentication

Required, see [Auth - Login](#)

Arguments

fragment (required)

One or more words (the last word can also be a fragment), which are "fuzzily" compared against known geotags stored in Infit.e

community ID list (required)

The list of community IDs or **community ID - regex** in the URL controls which sources the query will use. Obviously the logged-in user must have access rights to those communities. A user's communities can be viewed using the "[person/get](#)" API call.

Example

<http://infinite.ikanow.com/api/knowledge/feature/geoSuggest/madagas/4c927585d591d31d7b37097a>

Example Response



Social - Community - Add

`/social/community/add/{name}/{description}/{tags}`

`/social/community/add/{name}/{description}/{tags}/{parentid}`

 Add a new community (the created object is returned if successful - see [data model](#)).

Authentication

Required, see [Auth - Login](#)

Arguments

name (required)

Name of the new community. **Note:** community name must be unique.

description (required)

Text description of the the community.

tags (required)

Comma separated text values (i.e. 'news,sports,movies') - these are currently not used, but may be searchable in the future.

parentid (optional)

ID of the new community's parent community (typically the Infit.e System community) - *note this is currently not used and should be left blank.*

Examples

[http://infinite.ikanow.com/api/community/add/New Community/Text description for the new community/news,entertainment,fun](http://infinite.ikanow.com/api/community/add/New%20Community/Text%20description%20for%20the%20new%20community/news,entertainment,fun)
[http://infinite.ikanow.com/api/community/add/New Community/Text description for the new community/news,entertainment,fun/4c927585d591d31d7b37097a](http://infinite.ikanow.com/api/community/add/New%20Community/Text%20description%20for%20the%20new%20community/news,entertainment,fun/4c927585d591d31d7b37097a)

Example Response



```
{
  response:
  {
    action: "Add Community"
    success: true
    message: "The New Community community has been added."
    time: 83
  }
  data:
  {
    _id: "4e1c5d7c20be6403ca94c3bb"
    name: "New Community"
    description: "Text description for the new community"
    isSystemCommunity: false
    parentId: "4c927585d591d31d7b37097a"
    parentName: "Infinite System"
    isPersonalCommunity: false
    tags: ["news","entertainment","fun"]
    communityAttributes: {
      usersCanCreateSubCommunities: {type: "Boolean",value: "false"}
      registrationRequiresApproval: {type: "Boolean",value: "false"}
      isPublic: {type: "Boolean",value: "true"}
      usersCanSelfRegister: {type: "Boolean",value: "true"}
    }
    userAttributes: {
      publishCommentsPublicly: {type: "Boolean",defaultValue:
"false",allowOverride: false}
      publishQueriesToActivityFeed: {type: "Boolean",defaultValue:
"true",allowOverride: false}
      publishLoginToActivityFeed: {type: "Boolean",defaultValue:
"true",allowOverride: false}
      publishSharingToActivityFeed: {type: "Boolean",defaultValue:
"true",allowOverride: false}
      publishCommentsToActivityFeed: {type: "Boolean",defaultValue:
"true",allowOverride: false}
    }
    ownerId: "4d88d0f1f9a624a4b0c8bd71"
    communityStatus: "active"
    ownerDisplayName: "Jane Doe"
    numberOfMembers: 0
  }
}
```

Social - Community - Get

/social/community/get/{communityid}



Retrieves community information for the specified community.



Community Member Filtering

If a community has set the communityAttribute "publishMemberOverride" to false, then the members field will be removed before returning the community object. Administrators and community moderators are always allowed to see all other members as an exception to this rule.

Authentication

Required, see [Auth - Login](#)

Arguments

communityid (required)

Community ID to search on, can also be a regex (community ID - regex) provided that matches only a single community.

Example

<http://infinite.ikanow.com/api/community/get/4c927585d591d31d7b37097a>

Example Response



```
{
  response:
  {
    action: "Community Info"
    success: true
    message: "Community info returned successfully"
    time: 27
  }
  data:
  {
    _id: "4c927585d591d31d7b37097a"
    created: "Apr 12, 2011 6:56:47 PM"
    modified: "Apr 12, 2011 6:56:47 PM"
    name: "Infinite System"
    description: ""
    isSystemCommunity: true
    parentName: ""
    isPersonalCommunity: false
    tags: ["Infinite", "Community", "Share"]
    communityAttributes: {
      isPublic: {type: "boolean", value: "true"}
      usersCanSelfRegister: { type: "boolean",value: "false"}
      registrationRequiresApproval: {type: "boolean",value: "false"}
      usersCanCreateSubCommunities: {type: "boolean",value: "false"}
    }
    userAttributes: {
      publishLoginToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
      publishCommentsToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
      publishSharingToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
      publishQueriesToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
      publishCommentsPublicly: {type: "boolean",defaultValue:
"false",allowOverride: false}
    }
  }
}
```

```
}
ownerId: "4ca4a7c6b94b6296f6469d36"
communityStatus: "active"
ownerDisplayName: "Jim Moore"
numberOfMembers: 85
members: [
  {
    _id: "4cf77c709889a84922900249"
    email: "jill@ikanow.com"
    displayName: "Jilll Smith"
    userType: "member"
    userStatus: "active"
    userAttributes: [
      {type: "publishCommentsToActivityFeed",value: "true"}
      {type: "publishCommentsPublicly",value: "false"}
      {type: "publishQueriesToActivityFeed",value: "true"}
      {type: "publishLoginToActivityFeed",value: "true"}
      {type: "publishSharingToActivityFeed",value: "true"}]
    contacts: [
      {type: "mobile",value: "555.444.5555"}
    ]
  }
]
```

```
}  
}
```

Social - Community - Get All

/social/community/getall



Retrieves community information for all communities (public or private). Users who aren't admins can only see public communities and (private) communities to which they belong (or own).



Community Member Filtering

If a community has set the communityAttribute "publishMemberOverride" to false, then the members field will be removed before returning the community object. Administrators and community moderators are always allowed to see all other members as an exception to this rule.

Authentication

Required, see [Auth - Login](#), as above the functionality is limited for non-admins.

Arguments

N/A

Example

<http://infinite.ikanow.com/api/social/community/getall>

Example Response



```
{  
  response:  
  {  
    action: "Community Info"  
    success: true  
    message: "Community info returned successfully"  
    time: 27  
  }  
  data:  
  {  
    _id: "4c927585d591d31d7b37097a"  
    created: "Apr 12, 2011 6:56:47 PM"  
    modified: "Apr 12, 2011 6:56:47 PM"  
    name: "Infinite System"  
    description: ""  
    isSystemCommunity: true  
    parentName: ""  
    isPersonalCommunity: false  
    tags: ["Infinite", "Community", "Share"]  
    communityAttributes: {  
      isPublic: {type: "boolean", value: "true"}  
      usersCanSelfRegister: { type: "boolean",value: "false"}  
      registrationRequiresApproval: {type: "boolean",value: "false"}  
      usersCanCreateSubCommunities: {type: "boolean",value: "false"}  
    }  
    userAttributes: {
```

```
    publishLoginToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishCommentsToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishSharingToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishQueriesToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishCommentsPublicly: {type: "boolean",defaultValue:
"false",allowOverride: false}
  }
  ownerId: "4ca4a7c6b94b6296f6469d36"
  communityStatus: "active"
  ownerDisplayName: "Jim Moore"
  numberOfMembers: 85
  members: [
  {
    _id: "4cf77c709889a84922900249"
    email: "jill@ikanow.com"
    displayName: "Jill1 Smith"
    userType: "member"
    userStatus: "active"
    userAttributes: [
      {type: "publishCommentsToActivityFeed",value: "true"}
      {type: "publishCommentsPublicly",value: "false"}
      {type: "publishQueriesToActivityFeed",value: "true"}
      {type: "publishLoginToActivityFeed",value: "true"}
      {type: "publishSharingToActivityFeed",value: "true"}]
    contacts: [
      {type: "mobile",value: "555.444.5555"}
    ]
  }
]
```

```
}  
}
```

Social - Community - Get Private

/social/community/getprivate



Retrieves community information for all private (non-public) communities. Users who aren't admins can only see communities they own.



Community Member Filtering

If a community has set the communityAttribute "publishMemberOverride" to false, then the members field will be removed before returning the community object. Administrators and community moderators are always allowed to see all other members as an exception to this rule.

Authentication

Required, see [Auth - Login](#), as above the functionality is limited for non-admins.

Arguments

N/A

Example

<http://infinite.ikanow.com/api/community/getprivate>

Example Response



```
{  
  response:  
  {  
    action: "Community Info"  
    success: true  
    message: "Community info returned successfully"  
    time: 27  
  }  
  data:  
  {  
    _id: "4c927585d591d31d7b37097a"  
    created: "Apr 12, 2011 6:56:47 PM"  
    modified: "Apr 12, 2011 6:56:47 PM"  
    name: "Infinite System"  
    description: ""  
    isSystemCommunity: true  
    parentName: ""  
    isPersonalCommunity: false  
    tags: ["Infinite", "Community", "Share"]  
    communityAttributes: {  
      isPublic: {type: "boolean", value: "true"}  
      usersCanSelfRegister: { type: "boolean",value: "false"}  
      registrationRequiresApproval: {type: "boolean",value: "false"}  
      usersCanCreateSubCommunities: {type: "boolean",value: "false"}  
    }  
    userAttributes: {
```

```
    publishLoginToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishCommentsToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishSharingToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishQueriesToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
    publishCommentsPublicly: {type: "boolean",defaultValue:
"false",allowOverride: false}
  }
  ownerId: "4ca4a7c6b94b6296f6469d36"
  communityStatus: "active"
  ownerDisplayName: "Jim Moore"
  numberOfMembers: 85
  members: [
  {
    _id: "4cf77c709889a84922900249"
    email: "jill@ikanow.com"
    displayName: "Jill1 Smith"
    userType: "member"
    userStatus: "active"
    userAttributes: [
      {type: "publishCommentsToActivityFeed",value: "true"}
      {type: "publishCommentsPublicly",value: "false"}
      {type: "publishQueriesToActivityFeed",value: "true"}
      {type: "publishLoginToActivityFeed",value: "true"}
      {type: "publishSharingToActivityFeed",value: "true"}]
    contacts: [
      {type: "mobile",value: "555.444.5555"}
    ]
  }
]
```

```
}  
}
```

Social - Community - Get Public

/social/community/getpublic



Retrieves community information for all public communities.



Community Member Filtering

If a community has set the communityAttribute "publishMemberOverride" to false, then the members field will be removed before returning the community object. Administrators and community moderators are always allowed to see all other members as an exception to this rule.

Authentication

Required, see [Auth - Login](#)

Arguments

N/A

Example

<http://infinite.ikanow.com/api/community/getpublic>

Example Response



```
{  
  response:  
  {  
    action: "Community Info"  
    success: true  
    message: "Community info returned successfully"  
    time: 27  
  }  
  data:  
  {  
    _id: "4c927585d591d31d7b37097a"  
    created: "Apr 12, 2011 6:56:47 PM"  
    modified: "Apr 12, 2011 6:56:47 PM"  
    name: "Infinite System"  
    description: ""  
    isSystemCommunity: true  
    parentName: ""  
    isPersonalCommunity: false  
    tags: ["Infinite", "Community", "Share"]  
    communityAttributes: {  
      isPublic: {type: "boolean", value: "true"}  
      usersCanSelfRegister: {type: "boolean", value: "false"}  
      registrationRequiresApproval: {type: "boolean", value: "false"}  
      usersCanCreateSubCommunities: {type: "boolean", value: "false"}  
    }  
    userAttributes: {  
      publishLoginToActivityFeed: {type: "boolean", defaultValue:
```


```
"true",allowOverride: false}
  publishCommentsToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
  publishSharingToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
  publishQueriesToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
  publishCommentsPublicly: {type: "boolean",defaultValue:
"false",allowOverride: false}
}
ownerId: "4ca4a7c6b94b6296f6469d36"
communityStatus: "active"
ownerDisplayName: "Jim Moore"
numberOfMembers: 85
members: [
{
  _id: "4cf77c709889a84922900249"
  email: "jill@ikanow.com"
  displayName: "Jilll Smith"
  userType: "member"
  userStatus: "active"
  userAttributes: [
{type: "publishCommentsToActivityFeed",value: "true"}
{type: "publishCommentsPublicly",value: "false"}
{type: "publishQueriesToActivityFeed",value: "true"}
{type: "publishLoginToActivityFeed",value: "true"}
{type: "publishSharingToActivityFeed",value: "true"}]
  contacts: [
{type: "mobile",value: "555.444.5555"}
]
]
```




```
}  
}
```

Social - Community - Get System

/social/community/getsystem

 Retrieves community information for the system community.

 **Community Member Filtering**
If a community has set the communityAttribute "publishMemberOverride" to false, then the members field will be removed before returning the community object. Administrators and community moderators are always allowed to see all other members as an exception to this rule.

Authentication

Required, see [Auth - Login](#)

Arguments

N/A

Example

<http://infinite.ikanow.com/api/community/getsystem>

Example Response



```
{  
  response:  
  {  
    action: "Community Info"  
    success: true  
    message: "Community info returned successfully"  
    time: 27  
  }  
  data:  
  {  
    _id: "4c927585d591d31d7b37097a"  
    created: "Apr 12, 2011 6:56:47 PM"  
    modified: "Apr 12, 2011 6:56:47 PM"  
    name: "Infinite System"  
    description: ""  
    isSystemCommunity: true  
    parentName: ""  
    isPersonalCommunity: false  
    tags: ["Infinite", "Community", "Share"]  
    communityAttributes: {  
      isPublic: {type: "boolean", value: "true"}  
      usersCanSelfRegister: {type: "boolean", value: "false"}  
      registrationRequiresApproval: {type: "boolean", value: "false"}  
      usersCanCreateSubCommunities: {type: "boolean", value: "false"}  
    }  
    userAttributes: {  
      publishLoginToActivityFeed: {type: "boolean", defaultValue:
```

```
"true",allowOverride: false}
  publishCommentsToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
  publishSharingToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
  publishQueriesToActivityFeed: {type: "boolean",defaultValue:
"true",allowOverride: false}
  publishCommentsPublicly: {type: "boolean",defaultValue:
"false",allowOverride: false}
}
ownerId: "4ca4a7c6b94b6296f6469d36"
communityStatus: "active"
ownerDisplayName: "Jim Moore"
numberOfMembers: 85
members: [
{
  _id: "4cf77c709889a84922900249"
  email: "jill@ikanow.com"
  displayName: "Jilll Smith"
  userType: "member"
  userStatus: "active"
  userAttributes: [
{type: "publishCommentsToActivityFeed",value: "true"}
{type: "publishCommentsPublicly",value: "false"}
{type: "publishQueriesToActivityFeed",value: "true"}
{type: "publishLoginToActivityFeed",value: "true"}
{type: "publishSharingToActivityFeed",value: "true"}]
  contacts: [
{type: "mobile",value: "555.444.5555"}
]
]
```

```
}  
}
```

Social - Community - Member - Invite

/social/community/member/invite/{communityid}/{personid}

i Attempts to invite a user to a community. You must be the community owner to send invites. The person will receive an email allowing a user to accept or decline your membership offer. The person invited will respond to the invite using this API call: [Request Response](#)

Authentication

Required, see [Auth - Login](#). The caller must either be the community moderator/owner, or an administrator.

Arguments

communityid (required)

Community ID to invite a user to, can also be a regex (community ID - regex) provided that matches only a single community.

personid (required)

Person id to invite to community - this will normally be the person's username (normally email address) but can also be their database "_id".

skipinvitation (optional)

Allows administrators to automatically add a user to a community without having to send an invitation to the new user.

Example

<http://infinite.ikanow.com/api/community/member/invite/4cc942489889a84938870102/user@organization.com>

<http://infinite.ikanow.com/api/community/member/invite/4cc942489889a84938870102/user@organization.com?skipinvitation=true>

<http://infinite.ikanow.com/api/community/member/invite/4cc942489889a84938870102/4de942481532a84938870204>

http://infinite.ikanow.com/api/community/member/invite/*community%20name/4de942481532a84938870204

http://infinite.ikanow.com/api/community/member/invite/*community%20name/user@organization.com

Example Response

i

```
{"response":{"action":"Invite Community","success":true,"message":"Invited user  
to community successfully","time":868}}
```

Social - Community - Member - Join

/social/community/member/join/{communityid}

i Attempts to join a community. If the community accepts open membership, you will be added as a member immediately. If the community requires owner approval, an email will be sent to him and you will not be a member until the owner accepts you.

If owner approval is required the owner of the community will be asked to respond in email to this api call: [Request Response](#)
If the owner accepts the request, you will be silently added to the community.
If the owner denies the request, you will not be notified.

Authentication

Required, see [Auth - Login](#)

Arguments

communityid (required)

Community ID to attempt to join, can also be a regex (**community ID - regex**) provided that matches only a single community.

Example

<http://infinite.ikanow.com/api/community/member/join/4f5e189ada7f0000000510f>

Example Response



- Requires owner approval

```
{
  response:
  {
    action: "Join Community"
    success: true
    message: "Joined community successfully, awaiting owner approval"
    time: 45
  }
}
```

- Allows open enrollment

```
{
  response:
  {
    action: "Join Community"
    success: true
    message: "Joined community successfully"
    time: 45
  }
}
```

Social - Community - Member - Leave

/social/community/member/leave/{communityid}



Removes yourself as a member from a community.

Authentication

Required, see [Auth - Login](#)

Arguments

communityid (required)

Community ID to attempt to join, can also be a regex (**community ID - regex**) provided that matches only a single community.

Example

<http://infinite.ikanow.com/api/social/community/member/leave/4f5e189ada7f0000000510f>

Example Response




```
{
  response:
  {
    action: "Leave Community"
    success: true
    message: "Left community successfully"
    time: 45
  }
}
```

Error Response

Social - Community - Member - Update - Status

/social/community/member/update/status/{communityid}/{personid}/{userstatus}

 Changes a community member's status in the community.

Authentication

Required, see [Auth - Login](#). The caller must either be the community moderator/owner, or an administrator.

Arguments

communityid (required)

Community ID to change a members status in, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

personid (required)

Community member to change the status of - this will normally be the person's username (normally email address) but can also be their database "_id".

userstatus (required)

Status to change the member to (available statuses: "active", "disabled", "pending", "remove", the last of these removes the user from the community altogether)

Example

<http://infinite.ikanow.com/api/community/member/update/status/4f5e189ada7f0000000510f/4f5e1ed6da7f00000005110/active>

<http://infinite.ikanow.com/api/community/member/update/status/4f5e189ada7f0000000510f/user@organization.com/active>

Example Response



```
{
  response:
  {
    action: "Update member status"
    success: true
    message: "Updated member status successfully"
    time: 45
  }
}
```

Social - Community - Member - Update - Type

`/social/community/member/update/type/{communityid}/{personid}/{usertype}`

 Changes a community member's type in the community.

Authentication

Required, see [Auth - Login](#). Must be a system admin, the owner, or a moderator (except when changing ownership, in which case moderation rights are not sufficient).

Arguments

communityid (required)

Community ID to change a members status in, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

personid (required)

Community member to change the status of - this will normally be the person's username (normally email address) but can also be their database "_id".

usertype (required)

Type to change the member to (available types: "owner", "content_publisher", "moderator", "member" - note "content_publisher"s are members who can also publish sources)

Example

<http://infinite.ikanow.com/api/community/member/update/type/4f5e189ada7f00000000510f/4f5e1ed6da7f000000005110/user>

<http://infinite.ikanow.com/api/community/member/update/type/4f5e189ada7f00000000510f/user@organization.com/owner>

Example Response



```
{
  response:
  {
    action: "Update member type"
    success: true
    message: "Updated member type successfully"
    time: 45
  }
}
```

Social - Community - Remove

`/social/community/remove/{id}`

 Remove (delete) a community.

- The first time this is called, it just removes all users from and suspends the community (currently it must be manually re-activated, but no shares or sources or documents are deleted)
- The second time this is called, the community together with all its shares, sources, and documents are all irrevocably removed from the datastore

Authentication

Required, see [Auth - Login](#)

Arguments

id (required)

ID of the community to remove, can also be a regex (`community ID - regex`) provided that matches only a single community.

Example

<http://infinite.ikanow.com/api/community/remove/4e1c5d7c20be6403ca94c3bb>

Example Response

 First time:

```
{
  response: {
    action: "Delete community"
    success: true
    message: "Community disabled successfully - call delete again to remove for
good, including all sources, shares, and documents"
    time: 12
  }
}
```

Second time:

```
{
  response: {
    action: "Delete community"
    success: true
    message: "Community deleted forever."
    time: 12
  }
}
```

Social - Community - Request Response

`/social/community/requestresponse/{requestid}/{response}`

 Responds to a request, either issued by `community/member/join` or `community/member/invite`. The response can either be true (Accept) or false (Deny).

Authentication

Required, see [Auth - Login](#)

Arguments

requestid (required)
ID of the request you are responding to.

response
Your response to the request, either true or false

Example

<http://infinite.ikanow.com/api/community/requestresponse/4f5e189ada7f00000000510f>true>


Example Response



```
{
  response:
  {
    action: "Request Response"
    success: true
    message: "Request answered successfully!"
    time: 45
  }
}
```

Social - Community - Update

`/social/community/update/{communityid}`

 Update a community's information. **Note:** Can be called via post or get.

This method will overwrite the entire community object so it is to be used in the following format:

1. Get a community via `/social/community/get`
2. Edit that community, in a text editor, etc
3. Update the community with this call (overwriting the previous community)

Authentication

Required, see [Auth - Login](#)

Arguments

communityid (required)

ID of the community to update, can also be a regex (`community ID - regex`) provided that matches only a single community.

json (required)

JSON object

Example

Method.Get

`http://infinite.ikanow.com/api/community/update/4e1c5d7c20be6403ca94c3bb?json={...}`

Method.Post

Example using curl:

```
curl -XPOST
'http://infinite.ikanow.com/api/community/update/4e1c5d7c20be6403ca94c3bb/' -d '{
"field": "value" }'
```

Example Response




```
{
  response: {
    action: "Update community"
    success: true
    message: "Community updated successfully"
    time: 12
  }
}
```

Social - GUI - Modules - Delete

/social/gui/modules/delete/{moduleid}

Note - the "widget uploader" (`widgetUploader.jsp`) should be used in most cases, rather than this lower level call.

This call deletes widget metadata previously uploaded by the "modules/install" API call.

Authentication

Required, see [Auth - Login](#). In addition, only the system administrator or the owner can delete widget metadata.

Arguments

- The "_id" of the widget metadata, returned from the "modules/install" API call.

Example

<http://infinite.ikanow.com/api/social/gui/modules/delete/4d88d0f1f9a624a4b0c8bd71>

Example Response

Note

```
{
  "response": {
    "action": "Delete module",
    "success": true,
    "message": "module deleted successfully", // (or an error message if
success:false)
  }
}
```

Social - GUI - Modules - Get

/social/gui/modules/get

Note (Intended for integration with the Infinite GUI - not for 3rd party API access)

Returns a list of all approved modules.

Authentication

Required, see [Auth - Login](#)


Arguments

none

Example

<http://infinite.ikanow.com/api/social/gui/modules/get>

Example Response

 (Note that the "modules/install" API documentation contains the formal specification of the widget metadata shown below.)

```
{
  "response": {
    "action": "Get Modules",
    "success": true,
    "message": "modules returned successfully",
    "time": 0
  },
  "data": [
    {
      "_id": "4cb47f41bc945854b2016b71",
      "url": "com/ikanow/infinitt/e/modules/Maps.swf",
      "title": "Maps",
      "description": "Displays documents geospatially on Google Maps",
      "created": "1286897473814",
      "modified": "1286897473814",
      "version": "1.0",
      "author": "ikanow",
      "imageurl":
"http://www.sdltridionworld.com/images/GoogleMaps_tcm89-16510.gif",
      "approved": true,
      "searchterms": [
        "Maps",
        "Displays",
        "documents",
        "geospatially",
        "on",
        "Google",
        "Maps",
        "ikanow"
      ]
    },
    {
      "_id": "4cb47f41bc945854b3016b71",
      "url": "com/ikanow/infinitt/e/modules/ItemDetailView.swf",
      "title": "Details",
      "description": "Displays details of currently selected items",
      "created": "1286897473879",
      "modified": "1286897473879",
      "version": "1.0",
      "author": "ikanow",
      "imageurl":
"http://live.gnome.org/ProjectMonkey?action\u003dAttachFile\u0026do\u003dget\u00
26target\u003dmonkey.png",
      "approved": true,
      "searchterms": [
        "Details",
        "Displays",
        "details",
        "of",

```

```
        "currently",
        "selected",
        "items",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b4016b71",
    "url": "com/ikanow/infini/e/modules/LinkAnalysis.swf",
    "title": "Graph",
    "description": "Link Analysis view of documents and related
entities",
    "created": "1286897473881",
    "modified": "1286897473881",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://www.arvatodigitalservices.com/typo3conf/ext/db_yamltv/template/04_layout
s_styling/navi_link.gif",
    "approved": true,
    "searchterms": [
        "Graph",
        "Link",
        "Analysis",
        "view",
        "of",
        "documents",
        "and",
        "related",
        "entities",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b5016b71",
    "url": "com/ikanow/infini/e/modules/ListView.swf",
    "title": "Results",
    "description": "Displays documents in a list",
    "created": "1286897473883",
    "modified": "1286897473883",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://meritbadge.org/wiki/images/thumb/2/24/ToDoList.png/64px-ToDoList.png",
    "approved": true,
    "searchterms": [
        "Results",
        "Displays",
        "documents",
        "in",
        "a",
        "list",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b6016b71",
    "url": "com/ikanow/infini/e/modules/RadarChart.swf",
    "title": "Radar",
```

```
    "description": "Shows entities and their significance in a radar
chart",
    "created": "1286897473885",
    "modified": "1286897473885",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://www.uwec.edu/Help/Excel07/Images/other/cht_radar.gif",
    "approved": true,
    "searchterms": [
        "Radar",
        "Shows",
        "entities",
        "and",
        "their",
        "significance",
        "in",
        "a",
        "radar",
        "chart",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b7016b71",
    "url": "com/ikanow/infini/e/modules/Significance.swf",
    "title": "Stats",
    "description": "Shows entities and their significance in a bar and a
pie chart",
    "created": "1286897473887",
    "modified": "1286897473887",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://www.veryicon.com/icon/preview/System/Function/pie%20chart%2048%20Icon.jp
g",
    "approved": true,
    "searchterms": [
        "Stats",
        "Shows",
        "entities",
        "and",
        "their",
        "significance",
        "in",
        "a",
        "bar",
        "and",
        "a",
        "pie",
        "chart",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b8016b71",
    "url": "com/ikanow/infini/e/modules/Timeline.swf",
    "title": "Timeline",
    "description": "Shows documents on a timeline",
```

```
        "created": "1286897473889",
        "modified": "1286897473889",
        "version": "1.0",
        "author": "ikanow",
        "imageurl":
"http://clinicaltrials.ifpma.org/clinicaltrials/uploads/RTEmagicC_save-time_01.p
ng.png",
        "approved": true,
        "searchterms": [
            "Timeline",
            "Shows",
            "documents",
            "on",
            "a",
            "timeline",
            "ikanow"
        ]
    },
    {
        "_id": "4cbde99af5398e7ba63af4b9",
        "url": "com/ikanow/infini/e/modules/DetailsView.swf",
        "title": "Details",
        "description": "Displays details of currently selected items",
        "created": "1287514522802",
        "modified": "1287514522802",
        "version": "1.0",
        "author": "ikanow",
        "imageurl":
"http://www.hswallpapers.com/wp-content/uploads/2010/08/Bleach-Episode-284.png",
        "approved": true,
        "searchterms": [
            "Details",
            "Displays",
            "details",
            "of",
            "currently",
            "selected",
            "items",
            "ikanow"
        ]
    }
}
```

```
]
}
```

Social - GUI - Modules - Install

/social/gui/modules/install



Note - the "widget uploader" (`widgetUploader.jsp`) should be used in most cases, rather than this lower level call.

Used to POST a JSON object (specified below) to upload widget metadata (not the widget itself, which is stored separately either on a filesystem or via the [Infinite sharing API](#)).

Once the metadata has been successfully uploaded, the widget can be accessed from the GUI's module browser.

A simple [web-based utility](#) is available for installing widgets in one step.

Authentication

Required, see [Auth - Login](#)

Arguments

The content must be POSTed as a JSON object in the following format (see also the "get" API call).

```
{
  "_id": string, // Optional - auto-generated if not specified. If the "_id" is
  specified then updates the module as described below.
  "url": string, // Will normally either be a public URL or the location of a shared
  object in Infinite.
  "title": string, // The widget title displayed in the widget browser and on the
  widget window title bar
  "description": string, // A description of the widget, displayed in the widget
  browser
  "version": string, // An arbitrary string used for external versioning
  "imageurl": string, // The URL of an image used to represent the widget in the GUI
  framework (again normally either a public URL or share)
  "communityIds": [ string ], // The communities with which to share the module
  (must be owner/moderator/admin to share a module)
  "searchterms": [
    string // A list of strings that are used in the "modules/search" API call
  ]
}
```

There are 2 ways a module can be updated rather than uploaded:

- If the "_id" is specified and matches an existing module
 - If no "_id" is specified but the owner has already uploaded a widget with the same "url"
- Modules can only be updated either by the owner or by the system administrator.

Note also that the URL can contain the special variable "\$infinite". This is replaced at runtime with the URL from which Infinite is currently running (eg in the SaaS version, "http://infinite.ikanow.com/api/"). This is particularly useful when using the share API, eg:

```
{
  //...
  "url": "$infinite/share/get/4d88d0f1f9a624a4b0c8bd71"
  //...
}
```

Note finally that the following fields are added to the uploaded/updated JSON object (and are retrieved via the "module/get" API call):

```
{
  //...
  "_id": string, // If not specified
  "created": string, // First time inserted, the Unix time in ms
  "modified": string, // Each time created or modified, the Unix time in ms
  "author": string, // The (unique) email address of the uploader
  "approved": boolean, // Currently hardwired to true
  //...
}
```


Example

```
#bash> curl -XPOST 'http://infinite.ikanow.com/social/gui/modules/install' -d '{
  "url": "$infinite/share/get/4d88d0f1f9a624a4b0c8bd71",
  "title": "Test widget",
  "description": "A widget designed to test some new visualization",
  "version": "0.9",
  "imageurl": "$infinite/share/get/4d88d0f1f9a624a4b0c8bd72",
  "searchterms": [ "test", "visualization" ]
}'
```

A normal set of steps would be to upload the widget and thumbnails using the share API (saving the returned "_id"s, then upload the widget metadata as above, then test, then finally share the uploaded binaries to the relevant communities to let other users access them).

Note that there is currently a limitation in the above scenario, that all widget metadata are currently visible to all users (even though the share API protects access to the binaries themselves). A future release will fix this limitation.

Response

 The response format is straightforward, and most easily shown with this example:

```
{
  response:
  {
    action: "Install Module",
    success: true,
    message: "module installed/updated successfully" // (or the error
message if success:false)
  },
  data:
  {
    _id: "4d88d0f1f9a624a4b0c8bd71", // The _id of the store, used for
updating/deletion
    approved: true // Currently hardwired to true, since no approval
mechanism currently exists
  }
}
```

Social - GUI - Modules - Search

`/social/gui/modules/search/{term}`

i Note - the "widget uploader" (`widgetUploader.jsp`) should be used in most cases, rather than this lower level call.

(Intended for integration with the Infinite GUI - not for 3rd party API access)

Searches modules across all the user's communities, where the title, author or description contains the search term and returns a list of results.

Authentication

Required, see [Auth - Login](#)

Arguments

term (required)

Search term used to find matching modules (searches on title, author, and description)

Example

<http://infinite.ikanow.com/api/social/gui/modules/search/ikanow>

<http://infinite.ikanow.com/api/social/gui/modules/search/map>

<http://infinite.ikanow.com/api/social/gui/modules/search/rad>

Example Response




```

{
  "response": {
    "action": "Search Modules",
    "success": true,
    "message": "searched modules successfully",
    "time": 0
  },
  "data": [
    {
      "_id": "4cb47f41bc945854b6016b71",
      "swf": "RadarChart.swf",
      "url": "com/ikanow/infini/e/modules/",
      "title": "Radar",
      "description": "Shows entities and their significance in a radar
chart",
      "created": "1286897473885",
      "modified": "1286897473885",
      "version": "1.0",
      "author": "ikanow",
      "imageurl":
"http://www.uwec.edu/Help/Excel07/Images/other/cht_radar.gif",
      "approved": true,
      "searchterms": [
        "Radar",
        "Shows",
        "entities",
        "and",
        "their",
        "significance",
        "in",
        "a",
        "radar",
        "chart",
        "ikanow"
      ]
    }
  ]
}

```

Social - GUI - Modules - User - Get

/social/gui/modules/user/get



Note - the "widget uploader" (widgetUploader.jsp) should be used in most cases, rather than this lower level call.

(Intended for integration with the Infini.e GUI - not for 3rd party API access)

Returns a users current list of saved modules.

Authentication

Required, see [Auth - Login](#)

Arguments

none

Example

<http://infinite.ikanow.com/api/social/gui/modules/user/get>

Example Response



```
{
  "response": {
    "action": "Get User Modules",
    "success": true,
    "message": "users modules returned successfully",
    "time": 0
  },
  "data": [
    {
      "_id": "4cb47f41bc945854b4016b71",
      "swf": "LinkAnalysis.swf",
      "url": "com/ikanow/infinite/modules/",
      "title": "Graph",
      "description": "Link Analysis view of documents and related
entities",
      "created": "1286897473881",
      "modified": "1286897473881",
      "version": "1.0",
      "author": "ikanow",
      "imageurl":
"http://www.arvatodigitalservices.com/typo3conf/ext/db_yamltv/template/04_layout
s_styling/navi_link.gif",
      "approved": true,
      "searchterms": [
        "Graph",
        "Link",
        "Analysis",
        "view",
        "of",
        "documents",
        "and",
        "related",
        "entities",
        "ikanow"
      ]
    },
    {
      "_id": "4cbde99af5398e7ba63af4b9",
      "swf": "DetailsView.swf",
      "url": "com/ikanow/infinite/modules/",
      "title": "Details",
      "description": "Displays details of currently selected items",
      "created": "1287514522802",
      "modified": "1287514522802",
      "version": "1.0",
      "author": "ikanow",
      "imageurl":
"http://www.hswallpapers.com/wp-content/uploads/2010/08/Bleach-Episode-284.png",
      "approved": true,
      "searchterms": [
        "Details",
        "Displays",
        "details",

```

```

        "of",
        "currently",
        "selected",
        "items",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b3016b71",
    "swf": "ItemDetailView.swf",
    "url": "com/ikanow/infinet/e/modules/",
    "title": "Details",
    "description": "Displays details of currently selected items",
    "created": "1286897473879",
    "modified": "1286897473879",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://live.gnome.org/ProjectMonkey?action\u003dAttachFile\u0026do\u003dget\u00
26target\u003dmonkey.png",
    "approved": true,
    "searchterms": [
        "Details",
        "Displays",
        "details",
        "of",
        "currently",
        "selected",
        "items",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b2016b71",
    "swf": "Maps.swf",
    "url": "com/ikanow/infinet/e/modules/",
    "title": "Maps",
    "description": "Displays documents geospatially on Google Maps",
    "created": "1286897473814",
    "modified": "1286897473814",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://www.sdltridionworld.com/images/GoogleMaps_tcm89-16510.gif",
    "approved": true,
    "searchterms": [
        "Maps",
        "Displays",
        "documents",
        "geospatially",
        "on",
        "Google",
        "Maps",
        "ikanow"
    ]
},
{
    "_id": "4cb47f41bc945854b7016b71",
    "swf": "Significance.swf",

```

```
        "url": "com/ikanow/infinet/e/modules/",
        "title": "Stats",
        "description": "Shows entities and their significance in a bar and a
pie chart",
        "created": "1286897473887",
        "modified": "1286897473887",
        "version": "1.0",
        "author": "ikanow",
        "imageurl":
"http://www.veryicon.com/icon/preview/System/Function/pie%20chart%2048%20Icon.jp
g",
        "approved": true,
        "searchterms": [
            "Stats",
            "Shows",
            "entities",
            "and",
            "their",
            "significance",
            "in",
            "a",
            "bar",
            "and",
            "a",
            "pie",
            "chart",
            "ikanow"
        ]
    },
    {
        "_id": "4cb47f41bc945854b5016b71",
        "swf": "ListView.swf",
        "url": "com/ikanow/infinet/e/modules/",
        "title": "Results",
        "description": "Displays documents in a list",
        "created": "1286897473883",
        "modified": "1286897473883",
        "version": "1.0",
        "author": "ikanow",
        "imageurl":
"http://meritbadge.org/wiki/images/thumb/2/24/ToDoList.png/64px-ToDoList.png",
        "approved": true,
        "searchterms": [
            "Results",
            "Displays",
            "documents",
            "in",
            "a",
            "list",
            "ikanow"
        ]
    },
    {
        "_id": "4cb47f41bc945854b6016b71",
        "swf": "RadarChart.swf",
        "url": "com/ikanow/infinet/e/modules/",
        "title": "Radar",
        "description": "Shows entities and their significance in a radar
chart",
```

```
    "created": "1286897473885",
    "modified": "1286897473885",
    "version": "1.0",
    "author": "ikanow",
    "imageurl":
"http://www.uwec.edu/Help/Excel07/Images/other/cht_radar.gif",
    "approved": true,
    "searchterms": [
      "Radar",
      "Shows",
      "entities",
      "and",
      "their",
      "significance",
      "in",
      "a",
      "radar",
      "chart",
      "ikanow"
    ]
  }
```

```
]
}
```

Social - GUI - Modules - User - Set

`/social/gui/modules/user/set/{modules}`

i Note - the "widget uploader" (`widgetUploader.jsp`) should be used in most cases, rather than this lower level call.
(Intended for integration with the Infinite GUI - not for 3rd party API access)
Saves a users current set of modules.

Authentication

Required, see [Auth - Login](#)

Arguments

modules
comma delimited list of module ids to save to db for currently logged in user

Example

<http://infinite.ikanow.com/api/social/gui/modules/user/set/abcde12345,zyxwv98765>

Example Response

i

```
{
  "response": {
    "action": "Save Modules",
    "success": true,
    "message": "modules saved successfully",
    "time": 0
  }
}
```

Social - GUI - UISetup - Get

`/social/gui/uisetup/get`

i (Intended for integration with the Infinite GUI - not for 3rd party API access)
Returns current signed in user's last ui setup to aid in reloading the environment exactly how it was left.

Authentication

Required, see [Auth - Login](#)

Arguments

none

Example

<http://infinite.ikanow.com/api/social/gui/uisetup/get>

Example Response



```
{
  "response": {
    "action": "UISetup",
    "success": true,
    "message": "ui returned successfully",
    "time": 0
  },
  "data": {
    "profileID": "4ca4a7c1b94b6296ea469d36",
    "groupID": "4ca4a7c1b94b6296ec469d36",
    "queryString": "",
    "openModules": [
      {
        "widgetTitle": "com/ikanow/infini/e/modules/",
        "widgetUrl": "Maps.swf",
        "widgetDisplay": "Maps",
        "widgetImageUrl":
"http://www.sdltridionworld.com/images/GoogleMaps_tcm89-16510.gif"
      },
      {
        "widgetTitle": "com/ikanow/infini/e/modules/",
        "widgetUrl": "ListView.swf",
        "widgetDisplay": "Results",
        "widgetImageUrl":
"http://meritbadge.org/wiki/images/thumb/2/24/ToDoList.png/64px-ToDoList.png"
      }
    ]
  }
}
```

Social - GUI - UISetup - Update

/social/gui/uisetup/update/{modules}/{query}/{communityids}

/social/gui/uisetup/update/{modules}/{communityids}



(Intended for integration with the Infini.e GUI - not for 3rd party API access)

Updates a users setup so that it can be reloaded next time they log in.

Note that this API call is useful for clearing out saved GUI configurations (eg if it has become corrupt and causes problems on startup), eg: "<server>/api/social/gui/uisetup/update/null/%2A/4c927585d591d31d7b37097a"

Authentication

Required, see [Auth - Login](#)

Arguments

modules (required)

List of modules and the necessary information needed to reload them, null if no modules are loaded, the module list is delimited by], while individual module arguments are delimited by ,

query (required)

Last query ran so it can be reloaded into the browser, null if no query has been ran, query pieces are delimited by]

communityids (required)

Comma delimited list of community ids that are currently active



***Note: All arguments must be url encoded (specifically anytime a / is used it must be encoded to %2F so the url processor does not assume you are sending a different API call)**

Example

http://infinite.ikanow.com/api/knowledge/uisetup/update/com%2Fikanow%2Finfinit%2Fe%2Fmodules%2F,Maps.swf,Maps,http%3A%2F%2Fwww.sdtridionworld.com%2Fimages%2FGoogleMaps_tcm89-16510.gif/null/abcde12345

Example Response



```
{
  "response": {
    "action": "UISetup",
    "success": true,
    "message": "modules updated successfully",
    "time": 0
  }
}
```

Social - Person - Delete

`/social/person/delete/{id}`



Removes a user from the system.

A simple [web-based utility](#) is available for managing users.

Authentication

Required, see [Auth - Login](#), must come from [ikanow.com](#) website on Saas, or must be admin on deployed version

Arguments

id (required)

Person ID to delete, this can be the person id, the WPUserID, or their primary email address

Example

<http://infinite.ikanow.com/api/social/person/delete/4f5f4a56b154000000004581>

<http://infinite.ikanow.com/api/social/person/delete/WPUserID123>

http://infinite.ikanow.com/api/social/person/delete/sterling_archer@ikanow.com

Example Response



```
{
  response: {
    action: "WP Delete User"
    success: true
    message: "User Deleted Successfully"
    time: 45
  }
}
```


Social - Person - Get

/social/person/get

/social/person/get/{personid}



Retrieves information for the specified person. The format of the returned information is described [here](#).

Authentication

Required, see [Auth - Login](#)

Arguments

personid (optional) (requires admin)

Person ID or email address to search on, optional (if not included then the system searches on the ID of the user making the API call). This currently only works if you are an admin, in the future a public person search will be included.

Example

<http://infinite.ikanow.com/api/person/get>

<http://infinite.ikanow.com/api/person/get/4d88d0f1f9a624a4b0c8bd71>

<http://infinite.ikanow.com/api/person/get/user@ikanow.com>

Example Response



```
{
  response: {
    action: "Person Info"
    success: true
    message: "Person info returned successfully"
    time: 45
  }
  data: {
    _id: "4d88d0f1f9a624a4b0c8bd71",
    created: "Apr 19, 2011 9:46:25 PM",
    modified: "Jul 12, 2011 6:56:48 AM",
    accountStatus: "active",
    email: "jdoe@ikanow.com",
    firstName: "John",
    lastName: "Doe",
    displayName: "John Doe",
    organization: "IKANOW",
    title: "Software Engineer",
    location: "Woodbridge, VA",
    phone: "5555555555",
    languages: [
      "English",
      "Norwegian"
    ],
    avatar:
"http://profile.ak.fbcdn.net/hprofile-ak-snc4/161344_521863976_7207646_q.jpg",
    communities: [
      {
        _id: "4c927585d591d31d7b37097a",
        name: "Infinite System"
      },
      {
        _id: "4d88d0f1f9a624a4b0c8bd71",
        name: "John Doe's Personal Community"
      },
    ],
    WPUserID: "151",
    SubscriptionID: "172",
    SubscriptionTypeID: "1",
    SubscriptionStartDate: "Mar 22, 2011 4:40:12 PM"
  }
}
```

Social - Person - List

/social/person/list



Retrieves a list of person objects that are in communities with the current user

Authentication

Required, see [Auth - Login](#)

Arguments

none

Example

<http://infinite.ikanow.com/api/person/list>

Example Response



```
{
  "response": {
    "action": "People List",
    "success": true,
    "message": "List returned successfully",
    "time": 421
  },
  "data": [
    {
      "_id": "51cc5800e4b01b59e226dbcf",
      "created": "Jun 27, 2013 03:19:28 PM UTC",
      "modified": "Jun 27, 2013 06:08:28 PM UTC",
      "accountStatus": "active",
      "email": "cburch@test.com",
      "firstName": "calebtest",
      "lastName": "burch",
      "displayName": "calebtest burch",
      "phone": "5555555555",
      "communities": [
        {
          "_id": "51cc5800e4b01b59e226dbcf",
          "name": "calebtest burch's Personal Community"
        },
        {
          "_id": "4c927585d591d31d7b37097a",
          "name": "Infinite System"
        }
      ],
      "WPUserID": "cburch@test.com"
    }
  ]
}
```

Social - Person - Register

[/social/person/register](#)



Creates a user based on parameters passed in. Note there is a GET version of this call that follows the format [/social/person/register/user/auth](#) but is deprecated, please use the POST version.

A simple [web-based utility](#) is available for managing users.

Authentication

Required, see [Auth - Login](#), must come from [ikanow.com](#) website on Saas, or must be admin on deployed version

Arguments

POST.setup (required)

JSON object of a user and authentication following below format


Example


To use these examples: update the fields in the json objects to suit yourself (for example minimally change the fields: firstname, lastname, email, username, password)

These objects are explained in more detail on this page: [User creation and updating JSON object formats](#)

SetupJSON

```
SetupJSON =
{
  "user":
  {
    "created":"Oct 21, 2011 14:13:08 PM","modified":"Oct 21, 2011 14:13:08
PM","firstname":"jill","lastname":"smith","phone":"5555555555","email":["jillsmith@ika
now.com"]
  },
  "auth":
  {
    "password":"SHA256_HASHED_PASSWORD","accountType":"user","created":"Oct 21, 2011
14:13:08 PM","modified":"Oct 21, 2011 14:13:08 PM"
  }
}
```

 Dates must follow the format: **MMM dd, yyyy kk:mm:ss aa** as specified via java SimpleDateFormat: <http://download.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html>

 Passwords must be hashed by SHA256 into base64 (most websites hash to HEX).
<http://insidepro.com/hashe.php?lang=eng#1> will perform many hashes, the 2nd SHA256 is the base64 hashed password we require.

 You can use a site like: <http://meyerweb.com/eric/tools/dencoder/> to encode your JSON objects before adding them to the url

Method.Post

```
curl -XPOST 'http://infinite.ikanow.com/api/social/person/register' -d '{ "user":
{"created":"Oct 21, 2011 14:13:08 PM","modified":"Oct 21, 2011 14:13:08
PM","firstname":"jill","lastname":"smith","phone":"5555555555","email":["jillsmith@ika
now.com"]},"auth":{"username":"jillsmith@ikanow.com","password":"SHA256_HASHED_PASSWOR
D","accountType":"user","created":"Oct 21, 2011 14:13:08 PM","modified":"Oct 21, 2011
14:13:08 PM"}}'
```

Example Response



```
{
  response: {
    action: "WP Register User"
    success: true
    message: "User Registered Successfully"
    time: 10
  }
}
```

Common Error Messages:

- **Need to specify email:** the user object is required to have email field populated.
- **Need to specify one of firstname,lastname:** the user object is required to have the first and last name fields populated
- **User already exists, either WPUserId or first email:** Either the email address or the wordpress ID field you submitted have already been used.
- **Need to specify password:**the authentication object requires the password be set
- **error while saving objects:** this is a general error for any other failing.

Social - Person - Update

`/social/person/update/{wpuser}/{wpath}`



Updates a user based on parameters passed in.

Update works very similar to [social/person/register](#) but you are required to pass in an existing users email address or WPUserID (both in the user object).

A GET method exists as `/social/person/update/user/auth` but is deprecated, please use this POST version.

A simple [web-based utility is available](#) for managing users.

NOTE: If an [API key](#) is added via this update call, it is not applied until a [manual login](#) occurs.

Authentication

Required, see [Auth - Login](#), must come from [ikanow.com](#) website on Saas, or must be admin on deployed version

Arguments

POST.setup (required)

JSON object of a user and authentication following below format

Example

Fields that can be changed include: `user.email`, `user.firstname`, `user.lastname`, `user.phone`, `user.SubscriptionEndDate`, `user.SubscriptionID`, `user.SubscriptionStartDate`, `user.SubscriptionTypeID`, `auth.password`, `auth.accountType`

These objects are explained in more detail on this page: [User creation and updating JSON object formats](#)

SetupJSON

```
SetupJSON =
{
  "user":
  {

    "firstname": "jill", "lastname": "smith", "phone": "5555555555", "email": ["jillsmith@ikanow.com"]
  },
  "auth":
  {

    "username": "jillsmith@ikanow.com", "password": "SHA256_HASHED_PASSWORD", "accountType": "user"
  }
}
```

Method.Post

```
curl -XPOST 'http://infinite.ikanow.com/api/social/person/update' -d '{ "user":
{"firstname": "jill", "lastname": "smith", "phone": "5555555555", "email": ["jillsmith@ikanow.com"] }, "auth":
{"username": "jillsmith@ikanow.com", "password": "SHA256_HASHED_PASSWORD", "accountType": "user"} }'
```

Example Response



```
{
  response: {
    action: "WP Update User"
    success: true
    message: "User Updated Successfully"
    time: 10
  }
}
```

Common Error Messages:

- **Need to specify WPUserID (or email address):** the user object is required to have the WPUserID field or the email field populated.
 - **Can't find user specified by WPUserID:** the WPUserID submitted was not linked to a person in the system
 - **The primary email address is not unique:** the first email submitted in the user object email field has already been used for another user
- error while updating wp objects:** a general error for other issue occurred

Social - Person - Update - Email

/social/person/update/email/{id}/{email}

i Changes a users email addresses associated with the account. The first email address given will be used as the login (only 1 needs to be given).

A simple **web-based utility** is available for managing users.

Authentication

Required, see [Auth - Login](#) To update a user other than yourself you must be an admin.

Arguments

id (required)

ID of a users wordpress account, or a person._id, or their primary email

email (required)

A list of email addresses to replace on the account, the first email address will be used as the new login

Example

http://infinite.ikanow.com/api/social/person/update/email/wp12345/sterling_archer@ikanow.com,archer_sterling@ikanow.com

Example Response



```
{
  "response": {
    "action": "WP Update User",
    "success": true,
    "time": 0,
    "message": "User Updated Successfully"
  }
}
```

Social - Person - Update - Password

</social/person/update/password/{id}/{password}>

i Changes a users password.

A simple **web-based utility** is available for managing users.

Authentication

Required, see [Auth - Login](#). To update a user other than yourself you must be an admin.

Arguments

id (required)

ID of a users wordpress account, or a person._id, or their primary email

password (required)

New password of user. This can be [SHA-256](#) hashed already or if it is not, will hash for you.

Example

<http://infinite.ikanow.com/api/social/person/update/password/wp12345/newsecretpassword>


Example Response



```
{
  "response": {
    "action": "WP Update User",
    "success": true,
    "time": 0,
    "message": "User Updated Successfully"
  }
}
```

Social - Share - Add - Binary

/social/share/add/binary/{title}/{description} (POST)

 Saves a binary file to the share library. Typical uses for binary files include widget thumbnail images, widget swf files, and hadoop map reduce jars.

A simple [web-based utility](#) is available for performing many share management activities.

Authentication

Not required (this call gives auth)

Arguments

title (required)

a title of the binary file you are adding

description (required)

the description of the binary file you are adding


binary (required)

byte steam you want to save

Example

See [this example in Java](#) for how to share a file. **Note:** The java example sets the content-type before sending the object to the API. The api will store whatever content type it gets sent and use that to return the object when you get it at a later time. If no content type is set the API cannot determine what the stream of bytes means and will return it with a general content type later. This may cause issues if you are trying to download the files from the API at a later date if the content type is not set. A good list of content/media types can be found here: http://en.wikipedia.org/wiki/Internet_media_type

Example Response



```
{
  response: {
    action: "Share"
    success: true
    message: "New binary share added successfully. ID in data field"
    time: 10
  }
}
```

Social - Share - Add - Community

`/social/share/add/community/{shareid}/{comment}/{communityid}`



Adds a community to a share document allowing users within a community to view the share.

An overview of how shares work can be found [here](#).

A simple [web-based utility is available](#) for performing many share management activities.

Note that whenever a community moderator or admin adds a share to a community, then that share is automatically **endorsed** for that community.

Authentication

Required, see [Auth - Login](#)

Arguments

shareid (required)

Share ID of document to add community to

communityid (required)

Community ID to add, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

comment (required)

Text comment describing the share/reason for sharing with community

Example

<http://infinite.ikanow.com/api/social/share/add/community/4e175bfeb8ed6403f48ea7e2/Thought+that+the+group+would+find+this+interesting/4e176ee94a10e0279560d977>

Example Response



```
{
  response: {
    action: "Share"
    success: true
    message: "Community successfully added to the share"
    time: 10
  }
}
```

Social - Share - Add - JSON

`/social/share/add/json/{type}/{title}/{description}` (POST)



Add a shared object (JSON document) to the share collection. **Note:** Can be called via post or get.

Shares are just entries in our database where you can save any information you want. Currently our application uses the shares to store things like query history (JSON), map reduce jars (BINARY), widget swfs (BINARY), among other things. To create a share you can either call this call to add a JSON string entry or [share/add/binary](#) to insert a byte stream (such as a file). Once you have added a share only you have access to it (via your personal community). If you want others to have access to the share you can add their communities via [share/add/community](#).

A simple [web-based utility is available](#) for performing many share management activities.

Authentication

Required, see [Auth - Login](#)

Arguments

type (required)

Type of share: dataset, query, etc.

title (required)

Arbitrary display title for the share

description (required)

Arbitrary display description for the share

json (required, unless POST in which case the share JSON should be the body)

JSON object

Example

Method.Get

`http://infinite.ikanow.com/api/social/share/add/json/type/title/description?json=%7B%22userid%22%3A%224f5e189ada7f00000000510f%22%2C%22settings%22%3A%7B%22maximized%22%3Afalse%2C%22lastTabOpen%22%3A2%7D%7D`

Method.Post

Example using curl:

```
curl \-XPOST
'http://infinite.ikanow.com/api/social/share/add/json/type/title/description' \-d
'{"userid": "4f5e189ada7f00000000510f", "settings": {"maximized": false, "lastTabOpen": 2}}'
```

A detailed example of adding a jar file to the share db and then accessing it to schedule a map reduce job in JAVA can be found here: [Java Share Example](#)

Example Response



```
{
  response: {
    action: "New Share"
    success: true
    message: "New share added successfully."
    time: 10
  },
  data: "4e175bfeb8ed6403f48ea7e2"
}
```

Social - Share - Add - Reference

`/social/share/add/ref/{type}/{database}/{documentid}/{title}/{description}`



Add a reference to a JSON document from one of the Infinit.e databases - when it is requested via the [get](#) or [search](#) API calls, the current value of that referenced document is returned. Currently the following are supported: the [document metadata](#) collection, the aggregated [entity](#) or [association](#) objects, or the first record in a custom collection (ie the custom collection is treated differently from the others, see below).

A simple [web-based utility](#) is available for performing many share management activities.

For references to custom collections, only the first object in the collection is displayed.

Authentication

Required, see [Auth - Login](#)

Arguments

type (required)

The share "type", an arbitrary string that is used in the [search](#) API call - ie it is treated identically as for normal JSON shares.

database (required)

Type of object shared, the following values are supported:

- document.doc_metadata
- feature.entity
- feature.association
- custommr.customlookup

documentid (required)

ID of document to share, from the database specified by the "database" field above (the value of "_id" field)

title (required)

Display title for share

description (required)

Display description for share

Example

```
curl -XGET
'http://infinite.ikanow.com/api/social/share/add/ref/ReferenceType/custommr.customlook
up/5e175bfeb8ed6403f48ea7b4/My+reference+share/No+Description'
```

Example Response



```
{
  response: {
    action: "New Share"
    success: true
    message: "New share added successfully."
    time: 10
  },
  data: "4e175bfeb8ed6403f48ea7e2"
}
```

Social - Share - Endorse

`/social/share/endorse/{shareid}/{communityid}/{isendorsed}`



Adds or removes an **endorsed** tag for the specified community and the specified share.

Authentication

Required, see [Auth - Login](#) - must be an administrator or moderator for the community in question.

Arguments

shareid (required)

Share ID of document to endorse

communityid (required)

Community ID for which to endorse, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

isendorsed (required)

"true" or "false"

Example

<http://infinite.ikanow.com/api/social/share/endorse/4e175bfeb8ed6403f48ea7e2/4e176ee94a10e0279560d977>true>

http://infinite.ikanow.com/api/social/share/endorse/4e175bfeb8ed6403f48ea7e2/*sentiment>true

Example Response



```
{
  response: {
    action: "Share"
    success: true
    message: "Share endorsed successfully"
    time: 10
  }
}
```

Social - Share - Get

/social/share/get/{shareid}



Retrieves a specified share document by ID.
A detailed explanation of the share object can be found [here](#).

Setting `nocontent` will return just the metadata about a share.

Normally you will not set this parameter and get the actual content of a share. If a share is not a binary file the content will be returned in the `response.data.share` field of the json. If the share contains a binary file and `nocontent` is not set, then in place of returning the metadata of a share, the actually binary bytes will be returned as a response. NOTE: there will be no json response with the success header. The `mediatype` of the response bytes will be set to what the share has for `share.mediaType`

Authentication

Required, see [Auth - Login](#). You must be a member of one of the communities associated with a share to be able to access that share.

Arguments

shareid (optional)
Share ID to search on

nocontent (optional)
Set parameter to true if you want just the metadata of a share

nometa (optional)
Set parameter to true (JSON shares only) to return the share itself in the "data" field of the reply, rather than the database object consisting of share metadata with the share itself encoded as a string in the "data.share" field.

Example

<http://infinite.ikanow.com/api/social/share/get/4e175bfeb8ed6403f48ea7e2>
<http://infinite.ikanow.com/api/social/share/get/4e175bfeb8ed6403f48ea7e2?nocontent=true>

Example Response



```
{
  response:
  {
    action: "Share"
    success: true
    message: "Share returned successfully"
    time: 41
  }
  data:
  {
    _id: "4e175bfeb8ed6403f48ea7e2"
    created: "Jul 8, 2011 11:35:26 AM"
    modified: "Jul 10, 2011 9:27:36 AM"
    owner:
    {
      _id: "4d88d0f1f9a624a4b0c8bd71"
      email: "jdoe@ikanow.com"
      displayName: "John Doe"
    }
    type: "test"
    title: "Test Share"
    description: "Just a test of share functionality"
    share: "random stuff that gets put in share"
    communities: [
      {
        _id: "4d88d0f1f9a624a4b0c8bd71"
        name: "John Doe's Personal Community"
      }
      {
        _id: "4c927585d591d31d7b37097a"
        name: "Infinite System"
        comment: "Share with everyone!"
      }
    ]
  }
}
```

Social - Share - Remove

/social/share/remove/{shareid}



Removes a specified share document from the database by ID. This document will no longer be accessible to **ANYONE**, even other shared communities than your own.

A simple [web-based utility is available](#) for performing many share management activities.

Authentication

Required, see [Auth - Login](#)

Arguments

shareid (optional)
Share ID to remove from the database

Example

<http://infinite.ikanow.com/api/share/remove/4e175bfeb8ed6403f48ea7e2/>

Example Response



```
{
  response:
  {
    action: "Share"
    success: true
    message: "Share removed successfully"
    time: 41
  }
}
```

Social - Share - Remove - Community

[/social/share/remove/community/{shareid}/{communityid}](#)



Removes a community from share document.

An overview of how shares work can be found [here](#).

A simple [web-based utility is available](#) for performing many share management activities.

Authentication

Required, see [Auth - Login](#)

Arguments

shareid (required)

Share ID of document to remove the community from, can also be a regex ([community ID - regex](#)) provided that matches only a single community.

communityid (required)

Community ID to remove

Example

<http://infinite.ikanow.com/api/social/share/remove/community/4e175bfeb8ed6403f48ea7e2/4e176ee94a10e0279560d977>

Example Response



```
{
  response: {
    action: "Share"
    success: true
    message: "Community successfully removed from the share"
    time: 10
  }
}
```

Social - Share - Save - JSON



NOT USED, SAME AS SHARE/ADD

/social/share/save/json/{id}/{type}/{title}/{description}?json=<json-object> (GET)

/social/share/save/json/{id}/{type}/{title}/{description} (POST)



Saves some JSON to the share library.

Authentication

Not required (this call gives auth)

Arguments

id (required)

if you want to update a share, submit a valid ID otherwise you can submit null

type (required)

a descriptor of the type of json you are adding

title (required)

a title of the json you are adding

description (required)

the description of the json you are adding

json (required)

JSON object that you want to save

Example

Method.Get

<http://infinite.ikanow.com/api/share/save/json/idornull/type/title/description?json={}>

Method.Post

Example using curl:

```
curl -XPOST
'http://infinite.ikanow.com/api/share/save/json/idornull/type/title/description' -d
'{"field": "value"}
```

Example Response

Social - Share - Search

/social/share/search



Searches the share collection
A detailed explanation of the share object can be found [here](#).

Authentication

Required, see [Auth - Login](#). You must be a member of one of the communities associated with a share to be able to access that share.

Arguments

Note: All of the following arguments are optional.

searchby

V0 release supports "**community**" or "**person**". The latter is only supported for admin users (except for obtaining only your shares, see below). If "searchby" is not specified, searches across all joined communities.

id

Comma separated list of community IDs, or a community regex (for "searchby=community"), or a list of email addresses or a list of ids (for "searchby=person"). If this is not specified for "person", it defaults to the calling user.

type

Comma separated list of (arbitrary string) share types. Example values: "dataset", "feed", "source", "query". "binary".

skip

Number of results to skip.

limit

Number of results to return.

Example

<http://infinite.ikanow.com/api/social/share/search>

<http://infinite.ikanow.com/api/social/share/search?type=feed>

<http://infinite.ikanow.com/api/social/share/search?searchby=person&id=user1@test.com,user2@test.com>

<http://infinite.ikanow.com/api/social/share/search?searchby=community&id=4c927585d591d31d7b37097a&skip=0&limit=10>

http://infinite.ikanow.com/api/social/share/search?searchby=community&id=*system

<http://infinite.ikanow.com/api/social/share/search?searchby=community&id=4c927585d591d31d7b37097a&skip=0&limit=10>

Example Response




```

{
  response:
  {
    action: "Share"
    success: true
    message: "Share returned successfully"
    time: 41
  }
  data:
  [
    {
      _id: "4e175bfeb8ed6403f48ea7e2"
      created: "Jul 8, 2011 11:35:26 AM"
      modified: "Jul 10, 2011 9:27:36 AM"
      owner:
      {
        _id: "4d88d0f1f9a624a4b0c8bd71"
        email: "jdoe@ikanow.com"
        displayName: "John Doe"
      }
      type: "test"
      title: "Test Share"
      description: "Just a test of share functionality"
      share: "random stuff that gets put in share"
      communities: [
        {
          _id: "4d88d0f1f9a624a4b0c8bd71"
          name: "John Doe's Personal Community"
        },
        {
          _id: "4c927585d591d31d7b37097a"
          name: "Infinit.e System"
          comment: "Share with everyone!"
        }
      ]
    }
  ]
}

```

Social - Share - Update - Binary

/social/share/update/binary/{id}/{title}/{description} (POST)

Updates a binary file already in the share collection


See [social/share/add/binary](#) for a more detailed explanation of how binary files work.

A simple [web-based utility is available](#) for performing many share management activities.

Note that updating a share [unendorses](#) it for all communities for which the logged-in user is not a moderator.

Authentication

Required, see [Auth - Login](#)

 There is one exception to needing to be the owner or an administrator - if a share is only currently shared with a single community (this means it must have been explicitly unshared from the personal community of the user who created it), then it can be modified by any content publisher or community moderator for that community.

Arguments

id (required)
id of the existing binary share you want to replace

title (required)
a title of the binary file you are adding

description (required)
the description of the binary file you are adding

binary (required)
byte steam you want to save

Example


See [this example in Java](#) for how to share a file. It only needs modified to call update instead of add and to supply a share id.

Example Response

Social - Share - Update - JSON


`/social/share/update/json/{id}/{type}/{title}/{description}?json=<object-json> (GET)`

`/social/share/update/json/{id}/{type}/{title}/{description} (POST)`

 Update a document in the share collection. **Note:** Can be called via post or get.
A simple [web-based utility is available](#) for performing many share management activities.
Note that updating a share [unendorses](#) it for all communities for which the logged-in user is not a moderator.

Authentication

Required, see [Auth - Login](#).

 There is one exception to needing to be the owner or an administrator - if a share is only currently shared with a single community (this means it must have been explicitly unshared from the personal community of the user who created it), then it can be modified by any content publisher or community moderator for that community.

Arguments

id (required)
ID of share to update

type (required)
Type of share: dataset, query, etc.

title (required)
Arbitrary display title for the share

description (required)
Arbitrary display description for the share

json (required)
JSON object

Example

Method.Get

<http://infinite.ikanow.com/api/share/update/json/id/type/title/description?json=%7B%22userid%22%3A%224f5e189ada7f00000000510f%22%2C>

%22settings%22%3A%7B%22maximized%22%3Afalse%2C%22lastTabOpen%22%3A1%7D%7D

Method.Post

Example using curl:

```
curl -XPOST
'http://infinite.ikanow.com/api/share/update/json/id/type/title/description' \-d
'{"userid":"4f5e189ada7f00000000510f","settings":{"maximized":false,"lastTabOpen":1}}'
```

Example Response



```
{
  response: {
    action: "Update Share"
    success: true
    message: "Share updated successfully."
    time: 10
  }
}
```

Social - Share - Update - Reference

/social/share/update/ref/{id}/{type}/{database}/{documentid}/{title}/{description}



Update a reference to a JSON document from one of the Infinite databases - when it is requested via the [get](#) or [search](#) API calls, the current value of that referenced document is returned. Currently the following are supported: the [document metadata](#) collection, the aggregated [entity](#) or [association](#) objects, or the first record in a custom collection (ie the custom collection is treated differently from the others, see below).

A simple [web-based utility](#) is available for performing many share management activities.

For references to custom collections, only the first object in the collection is displayed.

Authentication

Required, see [Auth - Login](#)

Arguments

id (required)

ID of share to update

database (required)

Type of object shared, the following values are supported:

- document.doc_metadata
- feature.entity
- feature.association
- custommr.customlookup

documentid (required)

ID of document to share, from the database specified by the "database" field above (the value of "_id" field)

title (required)

Display title for share

description (required)

Display description for share

Example

Example Response



```
{
  response: {
    action: "New Share"
    success: true
    message: "New share added successfully."
    time: 10
  }
}
```

JSON object formats

This set of pages describes the format of JSON objects with which API users commonly interact:

- [Query-related objects](#)
- [Source configuration objects](#)
- ["Social" configuration objects](#)

Earlier versions of the data model are archived [here](#) .

Documents and their sub-objects (entities, associations, user metadata, aggregations)

Overview

The query API call returns a complex JSON object:

Query response format

```
{
  "response": {
    "action": "Query",
    "success": true,
    "message": string, // A human readable version of the query
    "time": integer // The number of milliseconds spent in the server to perform
the query
  },
  "stats": {
    "found": integer, // the number of documents matching the query
    "start": integer, // the number of documents skipped (in score order) to get
the "data" return array below
    "maxScore": number, // the highest score in the returned documents
    "avgScore": number // the average score across all analyzed documents
("score.numAnalyze" in the query, default 1000)
  },
  "data": [
    {...} // The document objects described in link below below
  ],

  // Aggregations of document sub-objects

  "events": [
    {...} // The association objects described in link below, "assoc_type":
"Event"
  ],
}
```

```
"facts": [
  {...} // The association objects described in link below, "assoc_type": "Fact"
],
"entities": [
  {...} // The entity objects described below
],

// Other aggregations

"geo": [
  {...} // See link to aggregation objects
],
"maxGeoCount": long,

"times": [
  {...} // See link to aggregation objects
],
"timeInterval": long,

"moments": [
  {...} // See link to aggregation objects
],
"momentsInterval": long,

"sources": [
  {...} // See link to aggregation objects
],
"sourceMetaTags": [
  {...} // See link to aggregation objects
],
"sourceMetaTypes": [
  {...} // See link to aggregation objects
]
```

```
    ],  
  }  
}
```

In this complex JSON object, the various aggregation formats are specified [here, in the query documentation](#).

The key objects in the JSON are [documents](#), and its sub-objects [entities](#), [associations](#) (relationships between entities, or between an entity and free text), and [source-specific metadata](#).

In addition to the [query API call](#), these can also be accessed in other ways, eg via the [knowledge/document/get API call](#), or via the [Widget API](#).

The [entities](#) and [associations](#) are both sub-objects of the document object, but also "aggregation" objects in their own right. The differences are described [here](#).

Association JSON format

Event object format

```
{
  "assoc_type": string, // "Event", "Fact", "Summary" - see below

  // Subject-verb-object
  "entity1": string, // A free form text field containing information about the
association "subject"
  "entity1_index": string, // 0-1, if present is the "index" field from one of the
entities in the parent document
  "verb": string, // A free form text field describing the association "verb"
  "verb_category": string, // Also a free form text field describing the
association "verb", but intended to group related verbs together (eg "travel" for
verbs: "flew", "drove")
  "entity2": string, // A free form text field containing information about the
association "object"
  "entity2_index": string, // 0-1, if present is the "index" field from one of the
entities in the parent document

  // Temporal
  "time_start": string, // 0-1, the start time, in ISO format
(yyyy-MM-dd[ 'T'HH:mm:ss])
  "time_end": string, // 0-1, the end time - if there is no start time, the
association is considered to be "instantaneous" (at least to within a day)

  // Geo-spatial
  "geo_index": string, // If the association geotag maps into an entity from the
parent document then this field is the "index" of that entity
  "geotag": { // 0-1, only if the association has been geotagged
    "lat": number,
    "lon": number,
  },

  // Sentiment:
  "sentiment": number, // The sentiment from entity1_index toward entity2 (and
optionally entity2_index)
  // Scoring
  "assoc_sig": number, // A significance score for the association object (see
below)
  "doccount": number, // The number of documents containing the association,
aggregations only
  "entity1_sig": number, // If "entity1_index" is populated, the significance of
entity1
  "entity2_sig": number, // If "entity2_index" is populated, the significance of
entity2
  "geo_sig": number // If "geo_index" is populated, the significance of that entity
}
```

The association object is intended to provide a minimal but generic description of various activities and relationships. It can be thought of as "subject verb object at location over time", where the subjects and objects can be free text and/or point to entities within the document. In this representation:

- The **subject** is "entity1" (free form text), or "entity1_index" (the "index" of an entity in the "entities" array of the document or the query).
- The **object** is "entity2"/"entity2_index"
- The **verb** is the "verb" field, with the option of providing a higher level "verb_category" field, which allows grouping of related fields (eg "walk", "drive" would both have category "travel")
- Times and locations are represented by "time_start", "time_end", "geo_index", and "geotag" as described in the JSON code block above.
 - Note that events might have neither a "time_start" nor a "time_end" - in general the document "publishedDate" field can be used

- (this is what happens automatically in the [event timeline aggregation](#)).
- Conversely, if an event does not have any geo information, it does not follow that fields such as the document geotag can be used.

The "assoc_type" field sub-categorizes the association object into one of three types, "Event", "Fact", or "Summary". Examples provided below should make the distinction clearer, but it can be simply described as follows:

- "Event": link multiple entities (via "entity1_index", "entity2_index", "geo_index") and represent a transient activity (eg travel)
- "Fact": link multiple entities like "Events" but represent (transient or permanent) relationships (eg being president)
- "Summary": generally link one entity to a free text (eg a quotation: "Obama says...").

Scoring relationships between entities is an interesting problem, more so than scoring the entities themselves (see [significance discussion](#)). Eg you can score them based on the entities that comprise them, the frequency of the relationship itself, as well as on various graphical criteria. We intend to do some more research on this topic for the future. For now, the "association significance" calculated by Infinit.e is simply a Pythagorean combination of the entity significances.

There are a few other minor differences between the association object in as an [aggregation](#) vs [document](#) child:

- "entity1", "entity2", and "verb" fields are not present for aggregations (since they can have large numbers of values across all the matching documents).
- Only "Event" and "Fact" association types appear in aggregations (since "entity1", "entity2" and "verb" fields are not present).
- "geotag" fields are not present for aggregations (because of the way in which the aggregated event is generated, ie another implementation limitation).
- In aggregations, the entity/geo significances may be set to 0 for some entities (if they don't occur in the most relevant subset of documents, ie another implementation limitation).

Note finally that there is a third representation of associations available from Infinit.e queries: the [events timeline](#)". In this case, associations are the same as those within the documents, except that scoring is different:

- The entity significances are not included
- The association significance ("assoc_sig") is simply the Pythagorean significance of all documents in which the association is included
- Where times are not included intrinsically in the association, they are filled in to span the time range of the documents including them.

Examples - document children

Example "Events"

```
// No time information:
{
  "entity1" : "Rowan Companies Inc.",
  "entity1_index" : "rowan companies, inc./company",
  "verb" : "announced",
  "verb_category" : "acquisition",
  "entity2" : "LeTourneau Technologies Inc.",
  "entity2_index" : "letourneau technologies inc/company",
  "assoc_sig": 23.34546557,
  "entity1_sig": 13.3454654,
  "entity2_sig": 20.2134568,
  "assoc_type" : "Event"
},
// Example event generated from structured data sources:
{
  "entity1" : "unknown from unknown",
  "entity1_index" : "unknown from unknown/personperpetrator",
  "verb" : "attacked",
  "verb_category" : "assault/attack",
  "entity2" : "targeted,business",
  "entity2_index" : "targeted,business/facilitytype",
  "time_start" : "2004-05-7",
  "geotag" : {
    "latitude" : "35.1666667",
    "longitude" : "33.3666667"
  },
  "geo_index" : "nicosia,nicosia,cyprus/location",
  "assoc_sig": 28.34546557,
  "entity1_sig": 13.3454654,
  "entity2_sig": 20.2134568,
  "geo_sig": 9.124365
  "assoc_type" : "Event"
}
```

Example "Facts"

```
// Time range:
{
  entity1: "amazon"
  entity1_index: "amazon, inc./company"
  entity2: "kindle"
  entity2_index: "kindle/product"
  assoc_type: "Fact"
  verb_category: "company product"
  time_start: "2011-05-27"
  assoc_sig: 23.34546557,
  entity1_sig: 13.3454654,
  entity2_sig: 20.2134568,
  time_end: "2011-06-21"
},
// No time:
{
  entity1: "joe faulhaber"
  entity1_index: "joe faulhaber/person"
  verb: "current"
  verb_category: "career"
  entity2: "software engineer"
  entity2_index: "software engineer/position"
  assoc_sig: 23.34546557,
  entity1_sig: 13.3454654,
  entity2_sig: 20.2134568,
  assoc_type: "Fact"
}
```

Example "Summaries"

```
{
  entity1: "dannel malloy"
  entity1_index: "dannel malloy/person"
  verb_category: "quotation"
  entity2: "this exciting project will be a blueprint for people all around the
country who are interested in developing this type of green solar charging technology,
linking renewable energy with electric vehicles and making our lives cleaner and
greener.; i'm excited to witness the future of this project, and i'm energized about
the innovative projects ge is undertaking in our state"
  assoc_type: "Summary"
  time_start: "2011-05-26"
  assoc_sig: 13.3454654
  entity1_sig: 13.3454654
},
{
  "entity1" : "questetra inc.",
  "entity1_index" : "questetra inc./company",
  "verb_category" : "company founded",
  "time_start" : "2008-00-01 12:00:00",
  "assoc_type" : "Summary"
  "assoc_sig": 13.3454654
  "entity1_sig": 13.3454654
}
```

Example aggregated associations

Example aggregated associations

```
{
//...
"facts": [
  {
    "assoc_sig": 13.799482037068763,
    "verb_category": "career",
    "doccount": 208,
    "entity1_sig": 14.467955770723677,
    "entity1_index": "goodluck jonathan/person",
    "entity2_sig": 13.096933419701786,
    "entity2_index": "president/position",
    "assoc_type": "Fact"
  },
  //...
],

"events": [
  {
    "assoc_sig": 0.5558116549888003,
    "verb_category": "joint venture",
    "doccount": 5,
    "entity1_sig": 0.12740185072834573,
    "entity1_index": "eni spa/company",
    "entity2_sig": 0.7756429335717384,
    "entity2_index": "nigerian national petroleum co./company",
    "assoc_type": "Event"
  },
  //...
],

//...
}
```

Example "events timeline":

Example events timeline

```
{
//...
"eventsTimeline": [
  {
    "time_start": "2010-10-15",
    "assoc_sig": 134.8044472232056,
    "verb_category": "career",
    "doccount": 1,
    "entity1": "ibrahim babangida",
    "entity2": "head of state , general",
    "entity1_index": "ibrahim babangida/person",
    "verb": "current",
    "entity2_index": "head of state , general/position",
    "assoc_type": "Fact"
  },
  {
    "time_start": "2010-10-02",
    "assoc_sig": 126.6919059017276,
    "verb_category": "career",
    "doccount": 2,
    "entity1": "ibrahim babangida",
    "entity2": "military president , general",
    "entity1_index": "ibrahim babangida/person",
    "verb": "past",
    "time_end": "2010-10-06",
    "entity2_index": "military president , general/position",
    "assoc_type": "Fact"
  },
//...
],
//...
}
```

Document JSON format

Document object format

```
{
  "data": [
    {
      // Administrative metadata:
      "_id": "string", // A unique and immutable ID for the document (DB-side
this changes with every update, updateId contains the invariant value)
      "updateId": "string", // (DB-side: this is the immutable ID copied to "_id" when
accessed via the API)
      "created": string, // A Java format data representing when the document
was harvested
      "modified": string, // A Java format data representing when the document
was last modified harvested (eg by an update)

      // Content metadata:
```

```

        "title": string, //The title of the document, as set by RSS (RSS feeds),
or source-specified for files/databases
        "description": string, //A summary of the document, as set by RSS (RSS
feeds), or source-specified for files/databases
        "url": string, //The location of the document. For feed types other than
RSS, its specification is described below
        "displayUrl": string, // Optional: an alternative URL (playing no functional role),
should be used instead of the url for display when present
        "sourceUrl": string, // For files containing large numbers of documents,
this is the URL of the master file (see below)
        "publishedDate": string, //A Java format date representing when the
content was published (not when it was added to Infinit.e)
        "docGeo": {
            "lat": number, // 0+, the "docGeo" object contains the lat and long in
degrees of tagged documents. Docs can be tagged using GeoRSS or in source-specific
ways
            "lon": number
        },

    // Content:
    "fullText": string, // Not normally present - see below under "Document Content"
sub-heading
        // Source metadata:
        "source": [ string ], // The "description" field of the document's parents
in "sources" (normally just one entry)
        "sourceKey": [ string ], // The "key" field of the document's parents in
"sources" (normally just one entry)
        "sourceType": string, // The source type from which the document was
harvested: "feed" (eg RSS), "file", or "db"
        "mediaType": string, // The "mediaType" field of the document's parent in
"sources"
        "tags": [ string ], // The "tags" field of the document's parent or
parents in "sources" (combined if more than one)
        "communityId": [ string ], // A list of "_id" fields from the communities
for which this document has been harvested (see below, normally just one entry)
        "index": string, // (Ignore, for internal use only: the real-time (Lucene)
index in which this document matched the query)

        // Sub-objects:
        "entities": [
            { ... } // 0+ Entity objects, see link below
        ],
        "associations": [
            { ... } // 0+ Association objects, see link below
        ],
        "metadata": { ... }, // 0-1, The metadata object, see link below

        // Query enrichment:
        "aggregateSignif": number, // Per-query normalized significance, see below
for link to basic scoring documentation
        "queryRelevance": number, // Per-query normalized (Lucene) relevance, see
below
        "score": number // Per-query combined normalized significance/relevance
scores, see below (this is the field used to rank documents)

```

```
}
]
}
```

There is one non-obvious characteristic of the "created" and "modified" fields, applying only to documents generated from file shares. In these cases, "created" is the date when the harvest occurred, and "modified" is the file time (which can obviously be earlier).

The source configuration mentioned in the description of the "source", "sourceUrl", "sourceKey", "mediaType", and "tags" fields is [here](#). To summarize the discussion there about "url" and "sourceUrl":

- "sourceUrl" is only used when a single file in a share pointed to by the File Harvester can contain multiple documents.
- In this case, the "sourceUrl" points to the file itself (ie many documents will have the same "sourceUrl").
- In this case, the "url" is by default set to "sourceUrl" + "/" + md5sum("metadata") - ie an approximately unique "url" that has no locational meaning and is just present for deduplication purposes. It is far preferable to use the "XmlSourceName" and "XmlPrimaryKey" fields in the "file" sub-object of the "source" object.
 - "XmlSourceName" and "XmlPrimaryKey" can also be used for JSON, in exactly the same way.

Conversely, "displayUrl" (which can be set in the structured analysis handler) is guaranteed not to be used by the Infinit.e platform. It is therefore useful for linking documents to external content. For reference, the way that it is used in the [Infinit.e GUI](#) is as follows:

- If it starts with "http://" then it is treated as a web link
- Otherwise, it is assumed to be a relative file path to the fileshare specified in the [source url](#) field. (eg you can use the "[Document - File - Get](#)" call with the "sourceKey" concatenated to the "displayUrl" to retrieve the file directly from the fileshare).

The "communityId" field is a list of [community IDs](#), taken from the "communityIds" field of the "source" object. Normally it will just consist of a single ID, but it can also be a list, where:

- The document has been harvested by multiple sources in different communities
- The sources have the same configuration, ie generate the same document (they might for example, use different entity extractors or post-processing and this generate very different document metadata)

The following "mediaType" values are currently used: "News", "Video", "Imagery", "Social", "Discussion", "Blog", "Record" (ie database record), "Report", "Intel" (essentially the same as "Report"). In practice "mediaType" is a freeform string, but it is recommended to try to restrict and manage possible values.

The scoring algorithms used to generate the significance, relevance, and aggregate scores for documents are discussed [here](#). An average score is 100.

Example documents

HTML document from RSS - entities and events disable, no metadata

```
{
  "_id": "4ddfd53f2ba07f612a69260d",
  "communityId": [ "4c927585d591d31d7b37097a" ],
  "created": "Wed May 25 07:04:00 EDT 2011",
  "description": "Testimony of National Cybersecurity and Communications
Integration Center Director Sean McGurk, NPPD, before the House Committee on Oversight
and Government Reform, Subcommittee on National Security, Homeland Defense and Foreign
Operations, \"Cybersecurity: Assessing the Immediate Threat to The United States\",
  "index": "doc_4c927585d591d31d7b37097a",
  "mediaType": "News",
  "modified": "Wed May 25 07:04:00 EDT 2011",
  "publishedDate": "Wed May 25 07:00:00 EDT 2011",
  "source": [ "DHS: National Cyber Security Division" ],
  "sourceKey": [ "http.www.dhs.gov.feeds.press_room.xml" ],
  "tags": [
    "industry:technology",
    "news",
    "topic:technology"
  ],
  "title": "Testimony of National Cybersecurity and Communications Integration
Center Director Sean McGurk, NPPD, before the House Committee on Oversight and
Government Reform, Subcommittee on National Security, Homeland Defense and Foreign
Operations, \"Cybersecurity: Assessing the Immediate Threat to The U.S.",
  "url": "http://www.dhs.gov/ynews/testimony/testimony_1306421842051.shtm",
  "aggregateSignif": 102.87293273096262,
  "queryRelevance": 100.0012728510481,
  "score": 101.69357226562605
}
```

Example [entities](#) and [associations](#) are shown in the linked pages.

Document from fileshare - entities and events disabled, no metadata

```
{
  "_id": "4e028d22a0ec7d6ef60da760",
  "associations": [],
  "communityId": [ "4db5c05fb246d25364aceca0" ],
  "created": "Wed Sep 17 19:39:00 EDT 2010",
  "description": "171939Z SEP 08\r\nBAGHDAD: CAR BOMBS ACROSS IRAQ, INCLUDING AT
LEAST ONE NEAR THE POLISH EMBASSY IN BAGHDAD, KILLED [...]",
  "docGeo": {
    "lat": 33.37586144623292,
    "lon": 44.47143713123417
  },
  "index": "doc_4db5c05fb246d25364aceca0",
  "mediaType": "Intel",
  "modified": "Wed Sep 16 19:39:00 EDT 2010",
  "publishedDate": "Wed Sep 17 19:39:00 EDT 2008",
  "sourceKey": [ "smb.fileshare.local.139.modus_input." ],
  "tags": [
    "Modus",
    "IED",
    "Intel",
    "extraction",
    "Iraq"
  ],
  "title": "vt123.kl",
  "url": "smb://fileshare.local:139/modus_input/inprocess/vt123.kl",
  "aggregateSignif": 176.07324737430665,
  "queryRelevance": 99.99991570637258,
  "score": 150.73734148607468
}
```

Example [entities](#) and [associations](#) are shown in the linked pages.

Generated document from database - entities, associations, and metadata disabled

```
{
  "_id": "4dcd39d96ab97f61f6dab9a6",
  "communityId": [ "4c927585d591d31d7b37097a" ],
  "created": "Sun Jan 30 00:01:00 EST 2011",
  "description": "Jan 30, 2011 12:00:00 AM: ADW OTHER was reported at the 1200 Block
of 1st NW",
  "index": "doc_4c927585d591d31d7b37097a",
  "mediaType": "Record",
  "modified": "Sun Jan 30 00:01:00 EST 2011",
  "publishedDate": "Sun Jan 30 00:00:00 EST 2011",
  "source": [ "data.dc.gov - Crime Incidents (ASP)" ],
  "sourceKey": [ "Bergen.washingtondc.IncidentReport" ],
  "title": "11012990",
  "url": "jdbc:mysql://mysqlserver.local:3306/washingtondc/IncidentReport/956013",
  "aggregateSignif": 99.5614992027838,
  "queryRelevance": 100.0012728510481,
  "score": 99.47491180174625
}
```

Example metadata is shown [here](#).

Generated document from XML file - entities, associations, and metadata disabled

```
{
  "_id": "4de67aad24757d6e99258a3c",
  "associations": [],
  "communityId": [ "4dd53fb4e40d93afb096c484" ],
  "created": "Wed Sep 17 19:39:00 EDT 2010",
  "description": "On 4 October 2006, late in the morning, near the al-Massudi School
in the Camp Sarah neighborhood of central Baghdad, Iraq, assailants detonated two
improvised explosive devices (IED) [...]",
  "docGeo": {
    "lat": 33.3386111,
    "lon": 44.3938889
  },
  "index": "doc_4dd53fb4e40d93afb096c484",
  "mediaType": "Report",
  "modified": "Wed Sep 16 19:39:00 EDT 2010",
  "publishedDate": "Wed Oct 04 00:00:00 EDT 2006",
  "source": "NCTC WITS Data",
  "sourceKey": [ "smb.fileshare.local.139.wits." ],
  "sourceUrl": [ "smb://fileshare.local:139/wits/allfiles/WITS_2006_10.xml" ],
  "tags": [
    "incidents",
    "nctc",
    "ied",
    "terrorism",
    "wits",
    "events",
    "worldwide",
    "incident"
  ],
  "title": "3 bodyguards, 18 civilians killed, 15 police officers, 11 bodyguards, 63
civilians wounded in IED and VBIED attacks in Baghdad, Iraq",
  "url":
  "https://wits.nctc.gov/FederalDiscoverWITS/index.do?N=0&Ntk=ICN&Ntx=mode%20match&Ntt=2
00695603",
  "aggregateSignif": 181.3723329799639,
  "queryRelevance": 100.0012728510481,
  "score": 154.28817043245692
}
```

Example metadata is shown [here](#)

Note the "url" field, which was constructed out of the XML and provides both guaranteed uniqueness and points to a location hosting the document (the default "url" provides neither of those things).

Document Content

Normally, the "fullText" field is discarded from the document metadata and instead stored in the "gzip_content" collection of the "doc_content" database in the format described [here](#). There are some exceptions to this (eg database records) - this is also described [here](#).

The "Knowledge - Document - Get" API call has a "return FullText" option that will re-inject the full text where available.

Document Contents JSON Format

doc_content.gzip_content format

```
{
  // Location
  url: string,
  sourceKey: string,
  communityIds: [ ObjectId ],

  // The gzipped content stored in the Lucene index (eg after the harvest processing
  pipeline)
  gzip_content: binary,
  gzip_len: integer,

  // Optional content:
  gzip_raw_content: binary, // The original text, before the processing pipeline (but
  after Tika for PDFs etc)
  gzip_raw_len: integer,

  gzip_md_content: binary, // The compressed document metadata object
  gzip_md_len: integer
}
```

Notes

Note that the "binary" type serializes to byte[].

The fields used in deployments are controlled by the following parameters from "infinite.service.properties" (in "/opt/infinite-home/config", normally auto-generated from "/opt/infinite-install/config/infinite.configuration.properties" and "/opt/infinite-home/config/infinite.service.properties.TEMPLATE" - currently only "store.maxcontent" is copied across from the infinite-install directory; to override the others they should be added to "/opt/infinite-home/config/infinite.service.properties.TEMPLATE"):

- store.maxcontent: (long) the maximum *uncompressed* length in bytes - data beyond this is truncated. Note that is *not* applied to gzip_md_content.
- store.rawcontent: (boolean) whether to store the original text
- store.metadata_as_content: (boolean) whether to store the metadata



Currently the content records are not generated in the following circumstances:

- the URL starts "jdbc://"
- the URL starts "smb://", "s3://", or "file:" and ends in either ".xml" or ".json"
- The document has a "sourceUrl" field (many XML/JSON/token-separated records in a single file)

The original idea behind this was that JSON and XML files, and RDBMS records, would typically not have a large block of data as the full text, instead the full text (if present at all) would be a composite of small fields.

There are a few ways this is not optimal for many current use cases:

- These days you can choose to ignore specified JSON/XML fields, so you will often have a large JSON field that generates the fullText via the structured analysis handler and then discarded
- You can change the URL

The medium term plan will be to allow users to specify manually the content saving behavior on a per source basis (with the above as the fallback if no manual override is specified). The source configuration is in the middle of a major overhaul for both functional and readability reasons, so this will be rolled into that forthcoming change.

Misc

Here is some sample Java code that shows how to access the unzipped content:

Unzipping the content

```
BasicDBObject dboContent = (BasicDBObject) contentDB.findOne(contentQ);
if (null != dboContent) {
    byte[] compressedData =
((byte[])dboContent.get(CompressedFullTextPojo.gzip_content_));
    ByteArrayInputStream in = new ByteArrayInputStream(compressedData);
    GZIPInputStream gzip = new GZIPInputStream(in);
    int nRead = 0;
    StringBuffer output = new StringBuffer();
    while (nRead >= 0) {
        nRead = gzip.read(storageArray, 0, 200000);
        if (nRead > 0) {
            String s = new String(storageArray, 0, nRead, "UTF-8");
            output.append(s);
        }
    }
    doc.setFullText(output.toString());
}
```

Entity JSON format

Entity object format

```
{
    // Basic metadata:

    "index": string, // the entity "primary key" within Infinit.e, of the form
disambiguated_name.toLowerCase() + "/" + type.toLowerCase()
    "disambiguated_name": string, // for a given "type", this is (aside from case) a
unique identifier for the entity
    "actual_name": string, // the most common string for which the entity was seen in
the document
    "type": string, // The entity type (see below)
    "dimension": string, // One of "Who" (people, organizations), "Where" (places),
or "What" (everything else)

    // Statistics:

    // Statistics - per document
    "relevance": number, // A value between 0 and 1, indicating the entity extraction
engine's "opinion" on the entity's relevance within the document
    "frequency": integer, // The number of times the entity occurs in the document
    "sentiment": number, // 0-1, the entity extraction engine's "opinion" on whether
the document refers to the entity approvingly (positive, <= 1.0) or disapprovingly
(negative, >= -1.0)

    // Statistics - global
    "totalfrequency": long, // The number of times the entity occurs in all documents
in the Infinit.e database (currently across all communities, see below)
    "doccount": long, // The number of documents in which the entity occurs in the
Infinit.e database (currently across all communities, see below)
```

```
// Statistics - per query, global
"datasetSignificance": number, // The (approximate) significance of the entity
aggregated across all matching documents (see below for link to scoring algorithms)
"queryCoverage": number, // The (approximate) % of all matching documents in
which the entity appears
"averageFreq": number, // The (approximate) average frequency (including
documents in which the entity doesn't appear) across all documents that match the
query
"positiveSentiment": number, // The sum of the positive sentiment counts across
the matching documents
"negativeSentiment": number, // The sum of the negative sentiment counts across
the matching documents
"sentimentCount": long, // The total number of sentiment counts (positive or
negative) across the matching documents

// Statistics - per query, per document
"significance": number, // The significance of the entity in this document (see
below for link to scoring algorithms)

// Other enrichment:

"geotag": { // 0-1, only if entity has been geotagged
  "lat": number, // (floating point)
  "lon": number // (floating point)
},
"ontology_type": string, // 0-1, only if entity has been geotagged - an OpenCyc
type mapped from the "type" field, see below under discussion about types
```

```
"linkdata": [ "string" ] // 0+, A list of useful links relating to the entity (eg
Wikipedia entries)
}
```

The set of values permitted by the "type" field depends on how the entity was extracted:

- "Commercial" third party entity extractors have a fixed set of types they generate, for example [OpenCalais](#) or [AlchemyAPI](#).
- Many other entity extractors ([NetOwl](#), [ModusOperandi](#)) are customizable, allowing deployers to add their own entity types.
- Similarly, using the Infinite [Structured Analysis Harvester](#), new entity types can easily be added.
 - In general, where custom entities are being created using the [Structured Analysis Harvester](#), it would be preferred if source developers took entity types from the [OpenCyc](#) repository. There are plans to use OpenCyc more formally in the future.

Although as noted above there are plans to integrate OpenCyc fully into Infinite, currently it is only used formally in one place: the "ontological_type" field that accompanies geotag fields. The purpose of this field is to map from different geotagged entity types into a single hierarchy that can be interpreted both internally (by searches) and externally (by visualization widgets and other follow-on analytics). Ontological type is discussed further in the [Geo JSON format](#) section.

The scoring algorithms used to generate the significance, relevance, and aggregate scores for documents are discussed [here](#). Significance is a % (0 indicating there is no correlation between the entity appearing in the document and the document matching the query, 100 indicating the entity only appears in matching documents).

The statistics and scoring for entities that span multiple communities follows these rules:

- Only entity instances from documents in communities over which a search is run (and hence to which a user belongs) are counted (for "doccount", "totalfreq", etc and the derived fields "significance" etc). *Therefore there is no leakage of either numeric or textual data across community boundaries except when desired, eg where a user belongs to both and is searching across both.*
- Occasionally, for implementation reasons, statistics for a community will not be available: eg no instances of a particular entity matched in documents from some community. In these cases, the statistics returned will be estimates.

Note that sentiment is currently only available when [AlchemyAPI](#) is used as the entity extractor.

Entity geo-tags are intended to be used to identify the permanent location of an entity; associations' geo-tags should be used to indicate the transient location of an entity. This may not always be the case however (and in fact nothing internal prevents entities from having different geo-tags in each document).

The "linkdata" field is an array of HTTP links from [The Linked Data Project](#) that provide additional information about common entities, where information is publicly available on the Internet. Currently "linkdata" is only populated when [AlchemyAPI](#) or [OpenCalais](#) are used as the entity extractor:

- [OpenCalais](#) links to a single web-page which then links to different resources such as Wikipedia, CIA Factbook, etc.
- [AlchemyAPI](#) generates a number of different links, to its different available resources (basically the same set as OpenCalais).

Finally, as noted in the [parent section](#), the above entity format is essentially the same for both entities as sub-objects of the [document JSON](#) and also as aggregated objects in their own right (ie the "entities" array in the [query reply object](#)). The only differences when they are aggregations are as follows:

- No "actual_name" field (since there can obviously be many such fields across all documents containing that entity: the [Knowledge - Feature - Alias Suggest](#) query call can be used to obtain these "aliases").
- No "relevance" or "sentiment" statistics, since these are specific to the mentions of an entity in a single document.
- The "significance" and "frequency" fields are the maximum values occurring in the most relevant subset of matching results (normally the top 1000).

Examples - as sub-object of document

Entity example, generated by OpenCalais

```
{
  actual_name: "Atheros Communications Inc."
  dimension: "Who"
  disambiguated_name: "ATHEROS COMMUNICATIONS, INC."
  doccount: 2
  frequency: 2
  index: "atheros communications, inc./company"
  linkdata: [
    "http://d.opencalais.com/er/company/ralg-trlr/e704df00-837e-3722-9c85-8537d37871d7"
  ]
  relevance: 0.326
  totalfrequency: 3
  type: "Company"
  significance: 0.5527277770637631
  datasetSignificance: 0.5527277770637631
  queryCoverage: 0.004475719993688294
  averageFreq: 0.0023094688221709007
}
```

Entity example, generated by AlchemyAPI

```

{
  actual_name: "England"
  dimension: "Where"
  disambiguated_name: "England"
  doccount: 32
  frequency: 57
  index: "england/country"
  linkdata: [
    "http://dbpedia.org/resource/England"
    "http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000014381"
    "http://sw.opencyc.org/concept/Mx4rvViWaZwpEbGdrcN5Y29ycA"
    "http://umbel.org/umbel/ne/wikipedia/England"
    "http://mpii.de/yago/resource/England"
  ]
  relevance: 0.810086
  sentiment: 0.0143614
  totalfrequency: 148
  type: "Country"
  significance: 31.537547455550165
  datasetSignificance: 19.636774116648777
  queryCoverage: 6.067441361627046
  averageFreq: 0.8454545454545455
  positiveSentiment: 0.19462859999999998
  negativeSentiment: -0.43113379999999996
  sentimentCount: 9
},
{
  actual_name: "Cambridge"
  dimension: "Where"
  disambiguated_name: "Cambridge"
  doccount: 274
  frequency: 1
  gazateer_index: "cambridge/city"
  geotag: {
    lat: 52.20805555555555
    lon: 0.1225
  }
  ontology_type: "city"
  linkdata: [
    "http://dbpedia.org/resource/Cambridge"
    "http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000049d17"
    "http://mpii.de/yago/resource/Cambridge"
  ]
  relevance: 0.223275
  sentiment: -0.285151
  totalfrequency: 325
  type: "City"
  significance: 0.3884393421493454
  datasetSignificance: 1.1061519850829167
  queryCoverage: 1.5071120430337133
  averageFreq: 0.01818181818181818
  positiveSentiment: 0
  negativeSentiment: -0.285151
  sentimentCount: 2
}

```


Entity example, as generated from XML

```
{
  actual_name: "Indiscriminate/Incidental, Civilian, Adult from Afghanistan"
  dimension: "Who"
  disambiguated_name: "Indiscriminate/Incidental, Civilian, Adult from Afghanistan"
  doccount: 548
  frequency: 41
  index: "indiscriminate/incidental, civilian, adult from afghanistan/victimtype"
  totalfrequency: 2838
  type: "VictimType"
  significance: 9.419613148150061
  datasetSignificance: 6.211531263035471
  queryCoverage: 0.9106727792752375
  averageFreq: 0.131
}
```

Examples - as standalone aggregation

Entity example (aggregation)

```
{
  "entities": [
    {
      "index": "linkedin/company",
      "datasetSignificance": 49.24796071489491,
      "sentimentCount": 357,
      "type": "Company",
      "sentiment": 0.155566,
      "totalfrequency": 954,
      "queryCoverage": 99.44289693593316,
      "doccount": 242,
      "dimension": "Who",
      "frequency": 13,
      "negativeSentiment": -0.7647487,
      "positiveSentiment": 51.84659600000009,
      "averageFreq": 4.6155988857938715,
      "significance": 61.000762242212744,
      "actual_name": "LinkedIn",
      "disambiguated_name": "LinkedIn",
      "linkdata": [
        "http://dbpedia.org/resource/LinkedIn",
        "http://rdf.freebase.com/ns/guid.9202a8c04000641f80000000003d3af7",
        "http://umbel.org/umbel/ne/wikipedia/LinkedIn",
        "http://mpii.de/yago/resource/LinkedIn"
      ]
    },
    // etc
  ],
  //etc
}
```

Geo JSON format

Overview

This section describes both the JSON format used to represent geospatial information, and also some rationale for that representation (specifically the meaning and use of the "ontology_type" field).

Geospatial information is present in the following objects:

- **Document objects** contains the "docGeo" field.
- **Entity objects** (both as document children and aggregated) contain "geotag" and "ontology_type" fields.
- **Association JSON format** contains a "geotag" field.
- **Geo aggregation objects** have a slightly different format, though again including latitude, longitude and ontology type.

There are essentially two types of object format: for the first three objects in the above list:

Most common geo format

```
//(entity or document object)
{
  "geotag": { // (or "docGeo" for documents
    "lat": number,
    "lon": number
  }
  "ontology_type": string // Only present for entities, for documents/associations,
  defaults to "point" (see below)
}
```

And then for geo aggregation:

Geo aggregations

```
// (query object)
{
  "geo": [
    {
      "type": string, // this is the ontology type
      "count": integer, // the number of times this lat/long/type appears in the query
      "lat": number,
      "lon": number
    },
    //etc
  ]
}
```

Ontology type - overview

The meaning and usage of the "ontology_type" field merits further discussion.

One of the issues with using a single (lat,long) point to geotag locations is that actual locations are areas (ignoring height, which is a problem for another day). Whether you care about this depends on the scale at which the query is being launched, eg:

- If you're looking from a national level then viewing cities as points is fine...
- ... If you're looking at a city plus the surrounding area of then it likely isn't.
- If you just want to see which countries are being talked about on a map, then viewing a country as a point is fine...
- ... If you're viewing a mix of geotagged entities including cities and countries, then it becomes confusing.

Obviously the ideal solution is to represent "where" entities with a polygon of points, but this is computationally expensive, difficult to generate, etc (and also doesn't solve many of the visualization problems - eg how do you render a set of entities with some polygons, some points on a map?).

The Infinit.e roadmap includes plans to incorporate an initially limited set of polygons in an initially limited way, which was the "last story out" in V0.

In the meantime, the "ontology type" provides a consistent terminology for dealing with the size of a geotagged entity in order to either include or ignore it in searches and visualization (for visualizations, eg inside actionsript or javascript, you can also use the type as an indicator to treat the point differently - eg go fetch its polygon from a third party source).

The remainder of this page should clarify this overview by explaining what the different values are allowed, how they are generated, and how they are used.

Ontology type - values

The values of the ontology type are a small subset of geographic types from the [OpenCyc OWL ontology](#):

- [continent](#)
- [country](#)
- [countrysubsidiary](#) (state, county, region)
- [city](#)
- [geographicalregion](#) (mountain range, lake)
- [point](#)

Note that apart from "[geographicalregion](#)", there is a clear hierarchy to the different types.

When entities are generated from [OpenCalais](#) or [AlchemyAPI](#), the following regex mappings are applied (substring matching is allowed):

- `/continent/i`: [continent](#)
- `/country/i`: [country](#)
- `/provinceorstate|stateorcountry/i`: [countrysubsidiary](#)
- `/city/i`: [city](#)
- `/naturalfeature|region|geographicalfeature/i`: [geographicalregion](#)
- any non-matching entity with a geotag: [point](#)

When generated by hand (ie the [Unstructured Analysis function](#)) or by a different (plugin) entity extractor, the following rules are applied:

- The user/plugin developer can specify the ontology type by hand
- The above matching rules are applied (ie defaulting to "[point](#)")

Other values may be added in the future if needed (together with the possibility of allowing users to generate their own ontology type mappings).

Ontology type - usage

Ontology types are currently used in three ways in the Infnit.e platform (including the stock visualizations):

- **When querying:** If the user specifies the "ontology_type" in the [geo query](#), then only strictly "smaller" types will be searched (eg if [countrysubsidiary](#) is specified then only [city](#) and [point](#) types will match). [geographicalregion](#) counts as being at the same level as continent for the purpose of this heuristic.
- **Entities returned from a query** (either as aggregations or child objects of documents): The ontology type is returned and the map widget allows the user to include or exclude flags based on type.
- **Geo aggregations:** The ontology type is returned and the map widget allows the user to include or exclude points from the heatmap based on type

Metadata JSON format

Document "metadata" contains source-specific information in a source-specific format. It can then be used in source-aware queries, aggregations and visualizations. It is useful as a compromise between taking advantage of Infnit.e's "flat" data model (that allows entities and documents from diverse sources to be meaningfully compared) while not losing the original structure (which likely would not have been there if it wasn't useful in the first place!).

Metadata can be generated in one of three ways:

- [By importing database records](#)
- [By importing XML objects](#)
- [By using the Infnit.e "unstructured analysis module"](#)

Metadata can be used in a few different ways:

- To build entities and events using the [Infnit.e "structured analysis module"](#)
- To perform source-specific [queries](#) and [aggregations](#)
- To use in domain-specific widgets/visualizations/code outside of the Infnit.e platforms.

The basic format of the "metadata" sub-object is a list of field,value pairs, where the value is always an array (often of size 1) of either atomic types or objects (arbitrarily nested).

Generic metadata example

```
{
  // (rest of the document object)
  "metadata": {
    "field1__double": [ 1.0 ], // (single atomic type)
    "field2": [ "1", "2", "3", "4" ], // (array of atomic types)
    "field3": [ { "type": "simple" } ], // (single simple object)
    "field4": [ { "type": { "nested": true } } ], // (single nested object)
    "field3": [ { "type": "simple", "index": 1 }, { "type": { "nested": true } },
    "index": 2 } ], // (array of objects)
    // etc
  }
}
```

i There is one important subtlety of which to be aware: in the real-time (Lucene) index used for *queries*, metadata fields across all sources must share the same type. Since metadata field names can be specified on a per source basis, the following steps are taken:

- **Only 2 types are supported normally:** objects (eg see XML metadata below), and strings (ie any "atomic" field)
- **In the index only**, and therefore only applying to free text searches (and in the future custom aggregations), object field names have "__obj" appended. So to search on the "atomic_field" value of the "example" object in a free text query, you would use the Lucene syntax **"example__obj.atomic_field: <value>"**.
- **Some special field name patterns are supported** and if used allow different types to be stored:
 - ***__double, *__long, *__bool:** force the values to the specified types.
 - ***__date/ISO:** errors unless the value is a date in **ISO format**; if it is, stores as a date.
 - ***__dateTimeJava:** errors unless the value is a date in format "MM/dd/yy hh:mm a||MM/dd/yy||MMM dd, yyyy hh:mm:ss a||MMM dd, yyyy"; if it is, stores as a date.
 - ***__dateYYYYMMDD:** errors unless the value is a date in format "yyyyMMdd"; if it is, stores as a date.
 - ***__dateRFC822:** errors unless the value is a date in format "EEE, dd MMM yyyy HH:mm:ss Z"; if it is, stores as a date.
 - ***__dateGMT:** errors unless the value is a date in format "dd MMM yyyy HH:mm:ss GMT"; if it is, stores as a date.
 - ***__discard:** does not add the field to the index.
 - ***__term:** indexes the field as a phrase (ie searching for "example" in "field__term": "this is an example" would return no matches; a match would only be returned if the entire field matched).

i One other field naming issue to be aware of is the following encodings:

- The "." character (which is forbidden in MongoDB field names) is encoded to "%2e"
- The "%" character is encoded to "%25"
 - (which guarantees that `URLDecoder.decode(encoded_metadata_field_name)` is the original pre-encoded name)

The remainder of this section describes the different ways in which the metadata can currently be constructed from the source data.

Metadata generated from RSS

Any source-specific metadata in RSS is added under the "_FEED_METADATA" object. For example, the following twitter-specific RSS object:

```
<item>
  <title>(TITLE)</title>
  <description>(DESCRIPTION)</description>
  <pubDate>Thu, 26 Apr 2012 20:17:31 +0000</pubDate>
  <link>(URL)</link>
  <twitter:source>&lt;a href=&quot;http://twitter.com/#!/download/iphone&quot;
rel=&quot;nofollow&quot;&gt;Twitter for iPhone&lt;/a&gt;</twitter:source>
</item>
```

Is rendered like this:

```
{
  "metadata": {
    "_FEED_METADATA": [{
      "twitter:source": "&lt;a href=&quot;http://twitter.com/#!/download/iphone&quot;
rel=&quot;nofollow&quot;&gt;Twitter for iPhone&lt;/a&gt;"
    }]
  }
}
```

(And object/array nesting inside XML is supported and mapped into JSON as you'd expect).
Metadata generated from databases

Data in (RDBMS) databases are organized into tables, eg:

rowA	rowB	...	rowN
valA1	valB1	...	valN1
valA2	valB2	...	valN2
...
valAm	valBm	...	valNm

The individual values in the database have atomic values (integers, strings, floating point numbers), although they can also be arrays (this is rarely used).

In Infnit.e, each row generates a separate document (ie record), as described in the "structured analysis module" (TODO link). Within these documents, the column names are the metadata fields, and the values are the entries.

If the entries are arrays then they generate multi-value arrays in the JSON; otherwise they generate single-value arrays as described above.

Generic database metadata

```
// Document generated from row "n"
{
  // Rest of document, then
  "metadata": {
    "rowA": [ valAn ],
    "rowB": [ valBn ],
    //...
    "rowF": [ valFn_1, valFn_2, ..., valFn_q ], // Example array entry in database
    //...
    "rowN": [ valNn ]
  }
}
```

The next code block shows an example of a real metadata block, generated from the following database record:

nid	ccn	repor tdate time	shift	offen se	meth od	block sitea ddre ss	latitu de	longi tude	city	state	ward	anc	smd	distri ct	pnc
9449 13	1100 1478	"Jan 4, 2011 12:00 :00 AM"	"UNK "	"BUR GLA RY"	2	"600 B/O 8TH ST NE"	"38.8 9812 0674 3302 0"	"-76. 9949 6375 3432 40"	"WA SHIN GTO N"	"DC"	6	"6A"	"6A0 2"	"FIR ST"	102

Generates:

Real metadata object generated from database entry

```
{
  "metadata" : {
    "nid" : [
      944913
    ],
    "ccn" : [
      11001478
    ],
    "reportdatetime" : [
      "Jan 4, 2011 12:00:00 AM"
    ],
    "shift" : [
      "UNK"
    ],
    "offense" : [
      "BURGLARY"
    ],
    "method" : [
      "2"
    ],
    "blocksiteaddress" : [
      "600 B/O 8TH ST NE"
    ],
    "latitude" : [
      "38.89812067433020"
    ],
    "longitude" : [
      "-76.99496375343240"
    ],
    "city" : [
      "WASHINGTON"
    ],
    "state" : [
      "DC"
    ],
    "ward" : [
      6
    ],
    "anc" : [
      "6A"
    ],
    "smd" : [
      "6A02"
    ],
    "district" : [
      "FIRST"
    ],
    "psa" : [
      102
    ]
  ]
}
```

Metadata generated from "documents" (PDF, doc, docx, ppt, pptx)

"Office" documents can generate various metadata fields. They are contained in an object called "_FILE_METADATA". Examples include:

- "title"
- "Author"
- "Creation-Date"
- "Original-Date"
- "Last-Modified"
- "latitude"
- "longitude"

See [Tika](#) (the underlying technology - eg [here](#)) for a more complete list.

Metadata generated from XML

XML documents can be very complex, containing arbitrary levels of nesting. Also, it is not possible without the XML specification to know what type the fields are.

Aside from this field type issue, XML documents can always be converted into JSON objects, with repeated fields turned into arrays. The typing issue is worked around by treating everything as strings.

For example,

```
<root>
  <value1>1</value1>
  <object>
    <nested1>string</nested1>
    <nested1>-2</nested1>
    <nested_object>
      <nested11>1.0</nested11>
    </nested_object>
  </object>
</root>
```

Can be converted into the following JSON object:

```
{
  "value1": "1",
  "object": {
    "nested1": [ "string", "-2" ],
    "nested_object": {
      "nested11": "1.0"
    }
  }
}
```

And then it is clear how this can be mapped into the document metadata:

```

{
  // (Rest of document)
  "metadata": {
    "value1": [ "1" ],
    "object": [ {
      "nested1": [ "string", "-2" ],
      "nested_object": {
        "nested11": "1.0"
      }
    } ]
  }
}

```

There is one further subtlety worth noting. Often in XML documents, lists are nested, eg:

```

<root>
  <elementList>
    <element>value1</element>
    <element><nested>value2</nested></element>
    <element>value3</element>
  </elementList>
</root>

```

This would get converted into the following metadata object:

```

{
  // (Rest of document)
  "metadata": {
    "elementList": [ {
      "element": [ "value1", { "nested": "value2" }, "value3" ],
    } ]
  }
}

```

Aside from the unnecessary extra level of nesting, the double array is ungainly. The [XML extraction configuration](#) allows specified XML elements to be ignored, eg nested lists such as "elementList" in the above example, resulting in the much more palatable:

```

{
  // (Rest of document)
  "metadata": {
    "element": [ "value1", { "nested": "value2" }, "value3" ],
  }
}

```

Here is a real-world example of the metadata generated by a complex XML object:

Metadata generated from an XML object

```

{
  "metadata" : {
    "summary" : [
      "On 7 May 2004, in Nicosia, Cyprus, three small bombs exploded

```


at the facility of the Cyprus Media Group, causing only minor damage and no casualties. No group claimed responsibility."

```
],
  "perpetrator" : [
    {
      "nationality" : "Unknown",
      "characteristic" : "Unknown"
    }
  ],
  "location" : [
    {
      "region" : "Europe",
      "citystateprovince" : {
        "stateprovince" : "Nicosia",
        "city" : "Nicosia"
      },
      "country" : "Cyprus"
    }
  ],
  "subject" : [
    "Newspaper offices damaged in bombing in Nicosia, Cyprus"
  ],
  "icn" : [
    "200460104"
  ],
  "multipliedays" : [
    "No"
  ],
  "incidentdate" : [
    "05/07/2004"
  ],
  "ied" : [
    "No"
  ],
  "facility" : [
    {
      "indicator" : "Targeted",
      "nationality" : "Cyprus",
      "targetedcharacteristic" : "Unknown",
      "definingcharacteristic" : "Unknown",
      "damage" : "Light",
      "combatant" : "No",
      "quantity" : "1",
      "facilitytype" : "Business"
    }
  ],
  "approximatedate" : [
    "No"
  ],
  "assassination" : [
    "No"
  ],
  "weapontype" : [
    "Explosive"
  ],
  "eventtype" : [
    "Bombing"
  ],
  "suicide" : [
```



```
    ],
  },
}
```

Metadata generated from the Infinite "unstructured analysis module"

Finally, and most simply, the "unstructured analysis module" allows community owners to specify "regex" expressions to extract strings from the text of documents.

In this case, the regex configuration block specifies the field name, and each instance of matches is added to the top-level array field of the metadata. Field values are always strings.

For example, if "regex1" matched twice with values "aaa" and "aab", and "regex2" matched once with the value "1234", then the following "metadata" object would be generated.

```
{
  // (Rest of document)
  "metadata": {
    "regex1": [ "aaa", "aab" ],
    "regex2": [ "1234" ]
  }
}
```

In addition the "unstructured analysis module" allows javascript to be specified to create arrays of arbitrary JSON objects. These are stored just like the XML metadata described above.

Social configuration objects

This set of pages describe the different objects that will be retrieved and posted from/to the API in order to create and configure users.

Note that the API calls are all described [from here](#).

- [User and User Authentication objects](#), used to [create](#) and [update](#) users.
- [Person objects](#), returned from [Social - Person - Get](#) API requests and most useful for determining one's own community memberships
- [Community objects](#), returned from various API requests, most commonly [Social - Community - Get Public](#).

Community JSON formats

Community objects are retrieved from API calls such as [Social - Community - Get Public](#). They can be useful for determining whether they can be joined, who is already a member, and more detailed information about their purpose.

Basic format

The basic format is simple, with some more complex extensions discussed at the bottom:

Community JSON format

```
{
  // Basic system parameters:
  "_id": string, // String representation of MongoDB ObjectId
  "created": string, // Java date format, time when community was created
  "modified": string, // Java date format, time when community details were last
  changed

  // Advanced system parameters:
  "communityStatus": string, // "active", "disabled", or "pending" (awaiting admin
  approval)
  "isSystemCommunity": boolean, // true for the one system community each cluster has
  (that every member joins by default)
  "isPersonalCommunity": boolean, // Every user also has a personal community, of which
  only they are a member
  "parentId": string, // String representation of MongoDB ObjectId, the "_id" of the
```

parent community, see below.

```
"parentName": string, // The "name" of the "parent" community, see below.

// Community metadata:
"ownerId": string, // The "_id" field of the community owner (ie the user who created
it)
"ownerDisplayName": string, // The above user's "displayName"
"name": string, // The display name of the community
"description": string, // A longer description
"tags": [ string ], // Metadata tags (for future searching applications)

// Member information:
"numberOfMembers": integer,
"members": [{
  // From "person" object, see link below
  "_id": string,
  "email": string,
  "displayName": string,
  "languages": [ string ],
  "contacts": [ {...} ], // The same format as for the "person" object
  "links": [ {...} ], // The same format as for the "person" object

  "userType": string, // One of "member", "content_publisher", "moderator" or "owner"
(moderators have the same abilities as the owner; content publishers can create active
sources without moderator/owner authorization)
  "userStatus": string, // One of "active", "disabled" or "pending" ("pending" means
awaiting approval from an owner/moderator - this functionality is not yet implemented)

  "userAttributes": [{ // Overrides for the default
    "type": string, // This actually maps onto the key from the parent "userAttributes"
map, see below
    "value": string // The default or overridden value
  }]
}],

// Community properties:
"userAttributes": { // This is a list of things that members can do (see below)
  "USERPROP1": { // This is a map, ie this format is for USERPROP1, USERPROP2, etc;
    "type": string, // "boolean", "string"
    "defaultValue": string, // string representation of the default value
    "allowOverride": boolean // if "true", users can override the default values;
defaults to false
  },
  //etc
},
"communityAttributes": { // This is a list of community properties (see below)
  "COMMPROP1": { // This is a map, ie this format is for COMMPROP1, COMMPROP2, etc;
    "type": string, // "boolean", "string"
    "value": string
  },
},
```

```
//etc
}
}
```

Note that most of the "members" object fields are simply copied from the [Person object](#).

i By default community parent attributes point to the system community, in which case they are ignored (the same as not being present). They can be set to a different community via the [community create API call](#). In that case the "sub-community" shares its Lucene index with the parent community (note this is purely an implementation detail, it makes no functional/security). It can be used where there are large numbers of communities, since creating large numbers of Lucene indexes gets inefficient.

Three of the fields in the above format are lists (actually sets) or maps of type/value objects:

- **userAttributes**: governs what community members can and can't do
- **communityAttributes**: describes what communities allow (normally in terms of discovery and joining)
- **members.userAttributes**: overrides of the default community "userAttributes"

These are now described.

Community attributes

The following attributes are currently supported:

- **isPublic**: If "true", can be seen by anyone (via [Social - Community - Get Public](#)); if "false", can only be seen by members and administrators
 - type: boolean
 - value: "true"/"false"
- **usersCanSelfRegister**: If "true" (and "isPublic" is also "true"), non-members can attempt to invite themselves to join the community (via [Social - Community - Member - Join](#)), note this doesn't guarantee success, see "registrationRequiresApproval"; if "false", only existing members (or admins) can invite non-members (via [Social - Community - Member - Invite](#)).
 - type: boolean
 - value: "true"/"false"
- **registrationRequiresApproval**: If "true" then any accepted invite (eg including normal invites and "self-invites" by non-members) must be approved by a community moderator (an email with a clickable link to [Social - Community - Request Response](#) is sent out).
 - type: boolean
 - value: "true"/"false"
- **usersCanCreateSubCommunities**: This is a placemaker for allowing users to create child communities (currently the "parent" of a community is not used for any functionality, so this is a spurious attribute).
 - type: boolean
 - value: "true"/"false"

User attributes

User attributes are slightly more complicated than community attributes because they are applied to each member when they join and then can sometimes be overridden by the user (or an administrator or a community moderator).

As a result, two object types are needed:

- The community has a global attribute map containing the attribute names, their default values and whether they can be overridden
- Each member has a list of these attributes (as type/values, with the "type" field somewhat confusingly mapping to the attribute name; see below for an example), whether they inherited the default or were overridden.

The following attributes are currently allowed (actually none are supported within the current functionality):

- **publishLoginToActivityFeed**: will determine if the community's activity feed will show a user's login.
 - type: boolean
 - defaultValue: "true"
- **publishCommentsToActivityFeed**: will determine if comments made by a user on objects like documents will appear on the community activity feed.
 - type: boolean
 - defaultValue: "true"
- **publishSharingToActivityFeed**: will determine whether the community activity feed will be notified when the user shares artifacts (queries, datasets etc).
 - type: boolean
 - defaultValue: "true"
- **publishQueriesToActivityFeed**: will determine whether new queries made by a user will be published to the community's activity feed.
 - type: boolean
 - defaultValue: "true"
- **publishCommentsPublicly**: if "true", non-members will be able to see comments made on objects like documents.

- type: boolean
- defaultValue: "false"

The following JSON fragment shows an example of attributes in action:

Community JSON object fragment - attributes in action

```
{
  //...
  "members": [
    {
      "_id" : "4cc945379889a84940870102",
      "email" : "jill@ikanow.com",
      "displayName" : "Jilll Smith",
      "userType" : "member",
      "userStatus" : "active",
      "userAttributes" : [
        {
          "type" : "publishQueriesToActivityFeed",
          "value" : "true"
        },
        {
          "type" : "publishCommentsPublicly",
          "value" : "false"
        },
        {
          "type" : "publishCommentsToActivityFeed",
          "value" : "true"
        },
        {
          "type" : "publishSharingToActivityFeed",
          "value" : "true"
        },
        {
          "type" : "publishLoginToActivityFeed",
          "value" : "true"
        }
      ]
    },
  ]
}
//...
"communityAttributes" : {
  "isPublic" : {
    "type" : "boolean",
    "value" : "true"
  },
  "usersCanSelfRegister" : {
    "type" : "boolean",
    "value" : "false"
  },
  "registrationRequiresApproval" : {
    "type" : "boolean",
    "value" : "false"
  },
  "usersCanCreateSubCommunities" : {
    "type" : "boolean",
    "value" : "false"
  }
}
```

```
},
"userAttributes" : {
  "publishLoginToActivityFeed" : {
    "type" : "boolean",
    "defaultValue" : "true",
    "allowOverride" : false
  },
  "publishCommentsToActivityFeed" : {
    "type" : "boolean",
    "defaultValue" : "true",
    "allowOverride" : false
  },
  "publishSharingToActivityFeed" : {
    "type" : "boolean",
    "defaultValue" : "true",
    "allowOverride" : false
  },
  "publishQueriesToActivityFeed" : {
    "type" : "boolean",
    "defaultValue" : "true",
    "allowOverride" : false
  },
  "publishCommentsPublicly" : {
    "type" : "boolean",
    "defaultValue" : "false",
    "allowOverride" : false
  }
}
```

```
    },  
    //...  
}
```

Person JSON format

The Person object is normally retrieved by users for themselves (or occasionally admins) with the [Social - Person - Get](#) API request.

Its main use is to get community identifiers (and also the user's "_id"), though users can always alternatively be identified by their email address, and communities can usually be identified by [pattern matching on their titles](#) (but not always, eg [Config - Source - Save](#)).

Person JSON format

```
{
  // System parameters:
  "_id": string, // String representation of MongoDB ObjectId
  "created": string, // Java date format, time when user was created
  "modified": string, // Java date format, time when account details were last changed

  // Joint system/display parameters:
  "accountStatus": string, // "active" or "suspended"
  "email": string, // The user's primary email address, used both for display and for
system communications

  // Display parameters copied or derived from "register" API call
  "firstName": string, // (from register/update)
  "lastName": string, // (from register/update)
  "displayName": string, // Generated as "firstName" + " " + "displayName"
  "phone": string, // (from register/update)

  // Community information:
  "communities": [{
    "_id": string, // String representation of MongoDB ObjectId, unique identified for
the community
    "name": string // The title of the community (as it was when the user joined; not
updated if it is later changed by the owner)
  }],

  // CMS parameters copied from "register" API call
  "WPUserID": string,
  "SubscriptionID": string,
  "SubscriptionTypeID": string,
  "SubscriptionStartDate": string, // Java date format
  "SubscriptionEndDate": string, // Java date format

  // Display parameters that currently must be populated by hand in the DB
  "organization": string,
  "title": string,
  "location": string,
  "languages": [ string ],
  "biography": string,
  "avatar": string, // (intended to be a URL pointing to a JPG or PNG)
  "contacts": [{
    "type": string, // (eg mobile, home, office, fax)
    "value": string
  }],
  "links": [{
    "title": string,
    "url": string
  }],
  "tags": [ string ]
}
```

Share JSON Format

Share objects are retrieved from API calls such as [social/share/get](#). They hold the metadata about something that has been stored in the

database.

```
{
  "_id" : string, //String representation on a mongodb ObjectId
  "created" : string, // Java date format, time when community was created
  "modified" : string, // Java date format, time when community was modified
  "owner" : {
    "_id" : string, //owner is string representation of mongodb ObjectId
    "email" : string, //email address of owner
    "displayName" : string, //the owners display name
  },
  "type": string, //the type of share dataset,query,etc
  "title": string, //an arbitrary title for this share
  "description": string, //an arbitrary title for this share
  "mediaType": string, //if this share has a binary object, the media type of that
object
  "tags" : [string], //a list of metadata tags, for searching on
  "share" : string, //the data if this share is not a binary share
  "documentLocation" : {
    "_id" : string, //string representation of mongodb objectId where document is stored
    "database" : string, //name of db where document is stored
    "collection" : string, //name of mongodb collection where document is stored
  },
  "communities" : [{
    "_id" : string, //String representation of a mongodb ObjectId for communityID
    "name" : string, //name of community
    "comment" : string, //a comment on why a community was added
  }],
  "endorsed": [string], // A list of community IDs for which this share has been
endorsed (see below)
  "binaryData" : [bytes] //deprecated location of binary files, only very old documents
will still contain this
}
```

Endorsed shares

The "endorsed" parameter allows for applications that use shares (eg [aliasing](#) and the [widget save framework](#)) to take advantage of the Infinite security framework. A share can only be endorsed for a community by one of the following:

- A system admin
- A community owner
- A community moderator

(And it is automatically unendorsed whenever it is modified by someone who is not one of the above user types).

Note that a share is *automatically* endorsed when shared to a community by someone who is one of the above user types.

How the "endorsed" parameter is used is completely up to the application - as a rule of thumb, it can be ignored where the share just contains user generated content that is viewed by other users in the community, but should probably be applied in most cases where the share is used to control the processing.

User creation and updating JSON object formats

In order to create a new user ([Person - Social - Register](#)), or to update an existing user ([Person - Social - Update](#)), the following object is POSTed:

Setup object

```
{
  "user": { ... },
  "auth": { ... }
}
```

Where the "user" object has the following format:

User create/update object

```
"user": {
  // System parameters:
  "WPUserID": string, // (See below for "WP" explanation) The primary key for users,
  defaults to "email[0]" if not specified
  "created": string, // Optional Java date format - set to be the time of the API call
  if not specified (and is immutable after that)
  "modified": string, // Optional Java date format - set to be the time of the API call
  if not specified (and is updated on each "update" call after that)
  // (created/modified might want to be explicitly set in CMS cases where the actual
  Infinit.e account creation is deferred)

  // Joint system/display parameters:
  "email": [ string ], // Must contain at least one entry; all but the first are
  ignored. (Also "email[0]" must be unique if WPUserID is not specified.)

  // Display parameters:
  "firstname": string, // Optional, though at least 1 of "firstname", "lastname" must
  be specified; used for display purposes only
  "lastname": string, // Optional, though at least 1 of "firstname", "lastname" must be
  specified; used for display purposes only
  "phone": string, // Optional, for display purposes only
  "mobile": string, // Optional, for display purposes only

  // CMS parameters (currently none of these are used, so are optional!)
  "SubscriptionID": string,
  "SubscriptionTypeID": string,
  "SubscriptionStartDate": string, // Java date format
  "SubscriptionEndDate": string // Java date format - once implemented, this date will
  be used to suspend user accounts once expired
}
```

and the "auth" object has the following format:

User authentication create/update format

```
"auth": {
  "WPUserID": string, // Optional - in "update" commands this can be populated (equal
to email address if no WPUserID originally specified) and the "user" object left blank
  "password": string, // The password, mandatory for "register" API calls (otherwise
optional). Can either be in the clear or SHA-256/Base64 encoded
  "accountType": string, // Optional, defaults to "user". Admins can set this to be
"admin" to create new administrators.
  "apiKey": string // Optional, if specified then the URL parameter "infinite_api_key"
can be used instead of logging in
}
```

Note that in many places, these objects are referred to as "wpuser" or "WordPressUser" and "wpauth"/"WordPressAuth" because they were originally only used in integrated CMS (eg WordPress) scenarios. Their role was later expanded to being the sole way of managing users.

(Note also that in addition to posting the aggregate object, the individual objects can be encoded and sent as URL parameters in a GET request, as described in the API pages linked above.)

Source configuration objects

Overview

Note that there is a [separate overview of how to use these objects to ingest data into Infinit.e](#). These pages are reference information.

The [Config - Source - Get](#) API returns a source document in the following response format ([Config - Source - Good](#) is similar but returns an array of source JSON objects instead):

Query Response Format

```
{
  response:
    "action": "Source Info"
    "success": boolean,
    "message": "string", // A human readable message (i.e. "Successfully
retrieved source info")
    "time": integer // The number of milliseconds spent performing the query
  },
  data: { ... } // The JSON format below
}
```

JSON format

Source Document

```
{
// User-defined top level metadata:
"title" : "string", // String, display title for source
"description" : "string", // String, display description of documents to be harvested
"url" : "string", // String, url/path to documents to harvest
"mediaType" : "string", // Type of document being harvested, i.e. Record, Report,
etc. Basically a free form string used to populate the corresponding field in the
document

"tags" : [ "string" ], // Array of tags that are appended to documents harvested for
```

this source

```
// Auto-generated top level metadata:
"_id" : "string", // A unique ID for the document
"key" : "string", // String, unique identifier for a source based on the url
"created": "string", // When the source was originally created (Java date format)
"modified" : "string", // When the source was last modified (Java date format)

// Social metadata
// User-generated:
"isPublic" : boolean, // Described below, under source privacy (summary: if
"isPublic" is true, only a restricted set of fields are visible)
// Admin-generated:
"isApproved" : boolean, // When a source is first added to a community, the admin (if
different to the owner) must approve it.
// Auto-generated:
"ownerId": "string", // The "_id" of the creating user (see person object) - only the
user and admins have write privileges on the source
"communityIds" : [ "string" ], // A list of "_id"s of communities (normally only one)
across which the source is shared
"appendTagsToDocs": boolean, // Defaults to true, if false then the "tags" array
isn't copied to the document

// Different extraction types:
"extractType" : "string", // Currently supported: "Feed" (for HTTP/RSS), "File" for
SMB (shared filesystem) file access, "Database" for SQL access
"authentication" : { ... }, // a generic authentication configuration object used
(currently) by "Feed" and "Database" harvesters
"rss": { ... }, // See RSS object below ("extractType":"Feed" only)
"file" : { ... }, // See File object below ("extractType":"File" only)
"database" : { ... }, // See Database object below ("extractType":"Database" only)

// Enrichment engines (all optional)
"useTextExtractor": string, // See "Using enrichment engines" below
"useExtractor": string, // See "Using enrichment engines" below
"extractorOptions": { ... } // See "Using enrichment engines" below
// Custom enrichment:
"structuredAnalysis" : { ... }, // See StructuredAnalysis object below
"unstructuredAnalysis" : { ... }, // See UnstructuredAnalysis object below

// Harvest status:
"harvest" : {
  "harvested" : "string", // The last time the source was checked for new documents
(Java date format)
  "harvest_status" : "string", // The status of the harvest: "success", "in_progress",
or "error"
  "harvest_message" : "string" // A free form message containing the most recent
errors encountered while harvesting
  "synced": "string", // The last time an internal "synchronization" process was
performed (not of general interest, Java date format)
},
"harvestBadSource" : boolean, // The source is ignored by the harvester if true and
reset daily, this is used by the harvester to discard "bad" sources that might recover
// (where the harvester deems a source unlikely to recover, it sets its "isApproved"
to false.) Note use "searchCycle_secs" to disable sources manually.

"searchCycle_secs": integer, // Optional, if set then the source will only be
harvested every "searchCycle_secs" seconds (eg set to 86400 to recheck source daily,
set to -1, or - the current value, to disable source temporarily)
```

```
"maxDocs": integer, // Optional, if set then once this threshold is reached then 1
document is deleted for every new document added, in age order
"duplicateExistingUrls": boolean, // Optional, if true then this source will never
duplicate existing documents within the community, even if the processing performed is
different

"searchIndexFilter:" { // Optional object that lets the user control which fields are
indexed into Lucene, ie are searchable (by default: all of them) - used to improve
performance
  "entityFilter": "string", // (regex applied to entity indexes, plus starts with "+"
or "-" to indicate inclusion/exclusion, defaults to include-only)
  "assocFilter": "string", // (regex applied to new-line separated entity indexes in
associations, starts with "+" or "-" to indicate inclusion/exclusion, defaults to
include-only)
  "entityGeoFilter": "string", // (regex applied to entity indexes if the entity has
geo, starts with "+" or "-" to indicate inclusion/exclusion, defaults to include-only)
  "assocGeoFilter": "string", // (regex applied to new-line separated entity indexes
in associations with geo, starts with "+" or "-" to indicate inclusion/exclusion,
defaults to include-only)
  "fieldList": "string", // (comma-separated list of doc fields, starts with "+" or
"- " to indicate inclusion/exclusion, defaults to include-only)
  "metadataFieldList": "string" // (comma-separated list of doc fields, starts with
```

```
"+" or "-" to indicate inclusion/exclusion, defaults to include-only)
}
}
```

The sub-objects in the above JSON are described from the following links:

- [Harvesting objects:](#)
 - [Authentication configuration format](#)
 - [RSS configuration format](#)
 - [Using the Feed Harvester](#)
 - [File configuration format](#)
 - [Using the File Harvester](#)
 - [Database configuration format](#)
 - [Using the Database Harvester](#)
- [Using enrichment engines](#)
- [Custom enrichment objects:](#)
 - [Structured Analysis Harvester configuration format](#)
 - [Using the Structured Analysis Harvester](#)
 - [Unstructured Analysis Harvester configuration format](#)
 - [Using the Unstructured Analysis Harvester](#)

Source privacy

Anyone in a community can view all sources within that community. If the "isPublic" field is set to true, then all fields are visible.



Note this includes passwords and javascript code - anything sensitive should be protected with "isPublic": false.

If the "isPublic" field is set to false, then the following fields are removed:

- "authentication"
- "file", "rss", "database"
- "structuredAnalysis"
- "unstructuredAnalysis"

And the following fields are modified:

- "url": everything after the leading "?" is truncated
- "rss.extraUrls.url": as above.

Authentication object

JSON format

The Authentication object of the Source document is a subset of the full Authentication object in that it only uses the 'username' and 'password' fields.

It is used for the authentication of [RSS sources](#) and [database sources](#), but not to allow access to [file sources](#) (which includes the domain and occurs in the linked object).

Source.authentication object

```
{
  "username" : "string", // String
  "password" : "string", // String - NOTE this is sha-256/Base64 encoded
}
```

Note: The password field in the Authentication object is currently clear text. If the string value placed in password is clear text it is not encrypted by Infinit.e. Encryption of the password field is planned for a future release.

Database object

JSON format

Note that there is a separate overview of how to use the Database Harvester. This page is reference information.

The Source.database object describes how documents can be harvested from a traditional RDBMS database using JDBC drivers.

Source.database object

```
{
  "databaseType" : "string", // String, type of DB to connect to
    // eg: mysql, db2, oracle, mssqlserver, sybase
  "hostname" : "string", // String, hostname of database server to connect to
  "port" : "string", // String, port database server is listening for connections on
  "databaseName" : "string", // String, name of database containing the data to be
  extracted
  "query" : "string", // String, SQL query used to retrieve full dataset from source
  (the first time the source is added)
  "deltaQuery" : "string", // String, SQL query used to retrieve updates from source
  after the first time through
  "deleteQuery" : "string", // String, not currently implemented
  "primaryKey" : "string", // String, primary key field in data set, used to help
  identify whether a record is new or previously harvested

  "title" : "string", // String, one of the columns from query/deltaQuery; populates
  the document's title field
  "snippet" : "string", // String, one of the columns from query/deltaQuery; populates
  the document's description field
  "publishedDate" : "string", // String, one of the columns from query/deltaQuery;
  populates the document's published date field
}
```

Enrichment engines

Infini.t.e's entity extractors take [harvested documents](#), ie URLs (RSS/HTML), text (files), or [metadata objects](#) (XML, databases), and add meaning in the form of [entities](#) and [associations between entities](#).

Examples of the built-in entity extractors (JSON field **"useExtractor"**) include:

- **"textrank"** - A "behind-the-firewall" OSS solution that uses [TextRank](#) (with some [OpenNLP](#)-based pre-processing) to extract key phrases from the text.
- **"AlchemyAPI"** - uses the [Named Entity Extraction](#) and [Sentiment Analysis](#) functions of the commercial [AlchemyAPI](#) service (there is a [free tier for AlchemyAPI](#) but it is very restrictive). [AlchemyAPI has the ability to extract associations](#) (as well as much more), but this feature has not yet been integrated into the tool.
- **"OpenCalais"** - extracts entities and associations using the free [OpenCalais service](#). No sentiment analysis function is available at this time.
- **"AlchemyAPI-metadata"** - uses the [Keyword Extraction](#) and [Sentiment Analysis](#) functions of the commercial [AlchemyAPI](#) service (there is a [free tier for AlchemyAPI](#) but it is very restrictive). This service also tags [AlchemyAPI Concepts](#) to documents as metadata.
- *We have developed a keyword extractor and geo-tagger based on [OpenNLP](#), which has been deployed operationally and will get released as Open Source soon, once it has been tidied up a bit more.*
- *Note that IKANOW has also integrated with behind-the-firewall entity extractors. These are typically commercial or GOTS products (our current favorite is [Saliency by Lexalytics](#)). Please contact us for more details.*

In addition to the above entity extractors, Infini.t.e has three options for "text extractors", which convert URLs into text (eg Advert removal, HTML tag cleansing etc):

- **"AlchemyAPI"** (not needed if it is being used as the entity extraction service)
- **"boilerpipe"**, an open source built-in HTML->text extractor (which may need help from the [Unstructured Analysis Handler](#), "simpleTextCleanser" function on some sources).
 - (Mostly [AlchemyAPI](#) outperforms boilerpipe, though this is not always the case, eg <http://english.aljazeera.net/Services/Rss/?PostringId=2007731105943979989>)
- **"tika"**, Takes URLs pointing to office documents, PDFs, emails, etc and uses [Tika](#) to convert them to text (and also to add [metadata](#) using the `_FILE_METADATA_` field)

In either of the above cases ("useExtractor" or "useTextExtractor"), the field can be set to **"none"**.

Per-source configuration for extractor engines

The "extractorOptions" field of the [source JSON object](#) allows for custom configuration of text and entity extractors.

The format of the object is in the form:

```
{
  // "app.<EXTRACTOR_NAME>.<PARAMETER_NAME>" : "<PARAMETER_VALUE>"
  // ...
  // eg:
  "app.alchemyapi-metadata.sentiment" : "true"
}
```

Currently the following configuration options are available:

- **alchemyapi:**
 - **postproc:** "1", "2", "or "3", "3" by default. "1" does some post-processing of geographic entities (AlchemyAPI tends to prefer US results even when the context clearly indicates a US location), "2" does some post-processing of person entities (AlchemyAPI tends to prefer famous people even when the context does not support that), "3" does both.
 - **sentiment:** "true"/"false", "true" by default. If enabled, a sentiment metric is attached to each extracted entity. Note that this results in use of an extra AlchemyAPI credit per document.
 - **concepts:** "true"/"false", "false" by default. If enabled, a **metadata field** called "concepts" is tagged to the document containing Wiki titles that are related to the contents of the document. Note that this results in use of an extra AlchemyAPI credit per document.
- **alchemyapi-metadata:**
 - **sentiment:** "true"/"false", "true" by default. If enabled, a sentiment metric is attached to each extracted entity. Note that this results in use of an extra AlchemyAPI credit per document.
 - **concepts:** "true"/"false", "true" by default. If enabled, a **metadata field** called "concepts" is tagged to the document containing Wiki titles that are related to the contents of the document. Note that this results in use of an extra AlchemyAPI credit per document.
 - **batchSize:** a string containing an integer, turned off by default. If turned on, the AlchemyAPI call goes out on a batch of documents (the specified number). This makes processing of small documents like tweets more economical (in return for a reduction in accuracy, eg the sentiment is calculated over the batch not each individual tweet).
 - **numKeywords:** a string containing an integer, uses the AlchemyAPI default (currently 50) if not specified. If specified, controls the number of keywords returned. If batching is enabled then the requested number is multiplied by the batch size.
 - **strict:** "true"/"false", "false" by default. If enabled, fewer high quality keywords are extracted.
- **opencalais:**
 - **store_raw_events:** "true"/"false", "false" by default. If enabled, a **metadata field** called "OpenCalaisEvents" is tagged to the document containing the raw JSON for **events**. This can be used to analyze new event definitions so they can be incorporated into the global OpenCalais configuration. It can also be used as a workaround via the [structured analysis harvester](#) where this is not possible.
- **textrank:** currently none
- **boilerpipe:** currently none
- **tika:** currently none

Feed object

JSON format

Note that there is a [separate overview of how to use the Feed Harvester](#). This page is reference information.

Feed Harvester configuration

```
"rss": {
  "feedType": string, // Currently not used - will allow for RSS vs Atom in future
releases (currently only RSS is supported)

  "waitTimeOverride_ms": integer, // Optional - if specified, controls the amount of
time between successive reads to a site (default: 10000ms):
  // ie if a site is timing out it may limit the number of accesses from a given IP
- set the number higher
  // for large sites you can increase the performance of the harvester by setting
this number lower
  "updateCycle_secs": integer, // Optional - if present harvested URLs may be replaced
if they are older than this time and are encountered from the RSS or in the
"extraUrls"
  "regexInclude": string, // Optional - if specified, only URLs matching the regex will
be harvested
  "regexExclude": string, // Optional - if specified, any URLs matching the regex will
not be harvested

  "extraUrls": [ // This array allows for manually specified URLs to be harvested once
  {
    "url": string, // The URL
    "title": string, // The title that the document will be given (ie the equivalent to
the RSS title)
    "description": string, // (Optional) The description that the document will be
given (ie the equivalent to the RSS description)
    "publishedData": string, // (Optional) The date that will be assigned to the
document (default: now) - this can be overridden from "structuredAnalysis"
    "fullText": string // (Optional) If present and "useTextExtractor" is "none", then
uses the specified string instead of the URL contents (mainly for debugging)
  },
  //etc
],
  "userAgent": string, // (Optional) If present overrides the system default user agent
string
  "proxyOverride": string, // (Optional) "direct" to bypass proxy (the default), or a
proxy specification "(http|socks)://host:port"
  "cookies": string, // (Optional) appends this string to the "Cookies" field (can
included multiple semi-colon separated cookie values)

  "searchConfig": { ... } // (Optional) A complex configuration object that allows the
contents of URLs to be used generate more URLs/docs to harvest
}
```

The "searchConfig" field is described in more detail [here](#). Here is its format for reference:

```

"searchConfig": {
  "userAgent": string, // (Optional) Overrides the "parent" (rss) setting for "search"
operations (see usage guide)
  "proxyOverride": string, // (Optional) "direct" to bypass proxy (the default), or a
proxy specification "(http|socks)://host:port"
  "cookies": string, // (Optional) appends this string to the "Cookies" field (can
included multiple semi-colon separated cookie values). If not specified then uses
rss.cookies (if specified).
  "globals": string, // Optional Javascript that is evaluated before script or
extraMeta (ie to define global functions)
  "script": string, // (Mandatory) Script, must "return" (last statement evaluated) an
array of the following format:
    // [ { "url": string, "title": string /* optional-ish */,
    //      "description": string /* optional */, publishedDate: string /* optional */,
    //      "spiderOut": string /*optional */ }
  "scriptlang": string, // (Mandatory) Only "javascript" is supported, use extraMeta
for different script types
  "scriptflags": string, // (Optional) The flags to apply to the above script, see
"unstructuredAnalysis.meta" for more details
  "extraMeta": [ {...} ], // (Optional) A pipeline of metadata extraction operations
that are applied prior to "script", see "Using The Feed Harvester" overview
  "pageChangeRegex": string, // (Optional) If non-null, this regex should be used to
match the pagination URL parameter (which will be replaced by pageChangeReplace)
    // Also, group 1 should be the start, to allow any offsets specified in the URL
to be respected
  "pageChangeReplace": string, // (Optional) Mandatory if pageChangeRegex is non-null,
must be a replace string where $1 is the page*numResultsPerPage
  "numPages": integer, // (Optional) Mandatory if pageChangeRegex is non-null -
controls the number of pages deep the search will go
  "numResultsPerPage": integer, // (Optional) Mandatory if pageChangeRegex is non-null
- controls the number of results per page
  "waitTimeBetweenPages_ms": integer, // (Optional) Only used if pageChangeRegex is
non-null - controls a wait between successive pages if set

  "maxDepth": integer // (Optional, defaults to 2) If spidering out (returning
"spiderOut": "true" from the script) the maximum depth to go
}

```

File object

JSON format

Note that there is a separate overview of how to use the File Harvester. This page is mostly reference information.

The Source.file object describes how documents can be harvested from a local or network attached file stores.

Source.file object

```
{
  "username" : "string", // Username for file share authentication,
  "password" : "string", // Password for file share authentication,
  "domain" : "string", // Domain location of the file share,

  "pathInclude": "string", // Optional - regex, only files with complete paths
  matching the regular expression are processed further
  "pathExclude": "string", // Optional - regex, files with complete paths matching
  the regular expression are ignored (and matching directories are not traversed)
  "renameAfterParse" "string", // Optional, renames files after they have been ingested
  - the substitution variables "$name" and "$path" are supported; or "" deletes the file
  // (eg "$path/processed/$name")

  "type": "string", // One of "json", "xml", "tika", "*sv", or null to auto decide

  "XmlRootLevelValues" : "string", // The root level value of XML to which parsing
  should begin
  // also currently used as an optional field for JSON, if present will create
  a document each time that field is encountered
  // (if left blank for JSON, assumes the file consists of a list of
  concatenated JSON objects and creates a document from each one)
  // (Also reused with completely different meaning for CSV)
  "XmlIgnoreValues" : "string", // XML values that, when parsed, will be ignored -
  child elements will still be part of the document metadata, just promoted to the
  parent level.
  // (Also reused with completely different meaning for CSV)
  "XmlSourceName" : "string", // If present, and a primary key specified below is also
  found then the URL gets built as XmlSourceName + xml[XmlPrimaryKey], Also supported
  for JSON and CSV.
  "XmlPrimaryKey" : "string", // Parent to XmlRootLevelValues. This key is used to
  build the URL as described above. Also supported for JSON and CSV.
}
```

Overview of Harvester

As of beta, the File harvester is set to harvest NetBIOS/Samba file shares. This assumes that the source url is a smb:// url.

The Harvesting Process

The harvester accesses the Samba file share via the Authentication credentials provided in the source's FilePojo. If the directory is not accessible, an error is logged and zero files are returned. The harvester will traverse a directory up to a depth of 5 directories after a successful connection and return those files for further harvesting and extraction.

As of beta, the harvesting has 2 unique paths based on the file extension.
XML Files

In order to parse xml files, several additional variables need to be defined within the File.

XmlRootLevelValues are XML keys that signify a new object or act as a container to specific data.

For example:

```

<IncidentList>
  <Incident>
    <ICN>987654321</ICN>
    <Subject>Test Data</Subject>
  </Incident>
  <Incident>
    <ICN>123456789</ICN>
    <Subject>Subject Test</Subject>
  </Incident>
</IncidentList>

```

In the above example, one wishing to consider per-incident data, would specify "Incident" in the **XmlRootLevelValues** list.

XmlIgnoreValues are XML keys that contain data that does not need to be harvested.

For Example:

```

<IncidentList>
  <Incident>
    <ICN>987654321</ICN>
    <Subject>Test Data</Subject>
  </Incident>
  <Unrelated>
    <ICN>123456789</ICN>
    <Subject>Subject Data</Subject>
  </Unreated>
  <Incident>
    <ICN>123456789</ICN>
    <Subject>Subject Test</Subject>
  </Incident>
</IncidentList>

```

If the Unrelated key contained data that need not be extracted, "Unrelated" would be added to the **XmlIgnoreValues** list.

XmlSourceName is the url of the FeedPojo's source. Sometimes XML data contains it's own reference to it's data's url which is then set to be the feed's url. This makes it difficult to keep from harvesting duplicated data without parsing out the large XML file again.

XmlPrimaryKey is a XML key that contains a value that is unique to the data.

For Example:

```

<IncidentList>
  <Incident>
    <ICN>987654321</ICN>
    <Subject>Test Data</Subject>
  </Incident>
  <Incident>
    <ICN>123456789</ICN>
    <Subject>Subject Test</Subject>
  </Incident>
</IncidentList>

```

For the above example, the ICN is unique for each incident, so the **XmlPrimaryKey** would be set to "icn". More information about setting the URL using XmlPrimaryKey is described in the [File Harvester overview](#).

Deduplication

The file harvester uses 2 methods of deduplication to ensure both performance and accuracy. First, the harvester checks to see if the Feed

source url has been harvested before. If the feed source url has been harvested, the XML is then completely parsed into feeds as if the file were new. The deduplication then continues on each feed created by the XML against it's extracted url.

Other File Types

The File Harvester uses Apache Tika v1.0 to extract data for other file types. Supported document formats can be found [here](#).

Deduplication

The harvester checks to see if the file's URI has been harvested before. If it has not, it will be harvested for the first time. If the feed has been harvested, the file's modified date is checked against the modified date of the file. If the file is newer, it will be harvested. If the file contains multiple documents (for XML/JSON), then all documents in that file will be deleted and re-harvested.

StructuredAnalysis object

JSON format

Note that there is a [separate overview of using the Structured Analysis Harvester](#). This page is reference information.

The StructuredAnalysis object of the Source document (the string fields are generally regexes of javascript, see link above):

Source.structuredAnalysis object

```
{
  "scriptEngine" : "string", // OPTIONAL: String, Infinite currently only supports
  "javascript" (or "JavaScript"), which is the default
  "script" : "string", // OPTIONAL: String, can contain one or more JavaScript
  functions,
    // i.e. "function func() { var foo = 'test'; return foo; }"
  "scriptFiles" : [ "string" ], // OPTIONAL: Array of Strings, URLs of JavaScript
    // files to import at runtime
  "caches": { "string": "string", ... } // A list of caches in the format
  <CACHE_NAME>:<ID> where <ID> is the "_id" of a JSON share, see overview

  "title" : "string", // OPTIONAL: String, else document title is whatever is generated
  by the harvester (eg from RSS/filename)
  "fullText" : "string", // OPTIONAL: String, else full text is taken from the document
  contents as per usual.
  "description" : "string", // OPTIONAL: String, else document description is whatever
  is generated by the harvester (or an entity extractor if supported)
  "displayUrl" : "string", // OPTIONAL: String, this field is just used for display
  "publishedDate" : "string", // OPTIONAL: String, must return a date string in a
  standard format (eg Java, Javascript, ISO, SMTP, MM/dd/yy, MM/dd/yyyy etc)
    // If not present, published data either comes from harvester (eg created date
  for files), or is the current time

  "entities" : [ { ... } ], // OPTIONAL: to create entities from the metadata (see
  below)
  "associations" : [ { ... } ], // OPTIONAL: to create associations
  (events/facts/summaries) from the metadata (see below)
  "docGeo" : { // (OPTIONAL, to specify the document geo tag)
    // ONE OF THE FOLLOWING 2 SETS OF FIELDS:
    // Specify directly:
    "lat" : "string", // latitude
    "lon" : "string", // longitude

    // Or fill in as many search options as possible, if a match can be found it
  populates the lat/long
    "city" : "string", // String
    "stateProvince" : "string", // String
    "country" : "string", // String
    "countryCode" : "string" // String
  },

  "rejectDocCriteria": "string", // OPTIONAL: String, an optional script that returns
  null to keep the document, any string to reject the doc (the string is logged)
  "metadataFields": "string", // OPTIONAL: String, if present a comma-separated list of
  top-level metadata fields to either exclude (if "metadataFields" starts with '-'),
    // or only include (starts with '+', default) - the fields are deleted after all
  processing but before indexing and storage.
  "onUpdateScript": "string" // OPTIONAL: Used to preserve existing metadata when
  documents are updated, and also to generate new metadata based on the differences
    // between old and new documents. This script is discussed further in the
  Structured Analysis Overview linked at the top of this page
}
```

The entity format this specification object generates is documented [here](#).

Source.structuredAnalysis.entities object

```
{
  "iterateOver" : "string", // OPTIONAL: If specified, a metadata field (nesting
supported using dot notation) which is looped over to generate calls with
_value/_iterator/_index
  "disambiguated_name" : "string", // MANDATORY: String/script, the disambiguated name
of the entity
  "actual_name" : "string", // OPTIONAL: String/script, the actual name of the entity
if different to the disambiguated name
  "dimension" : "string", // MANDATORY: String/script: Must be/return one of "Who",
"What", "Where"
  "type" : "string", // MANDATORY: String/script: It is recommended to use a type from
the
  // OpenCyc, AlchemyAPI, or OpenCalais ontologies, for compatibility with future
Infinite features

  "linkdata": "string", // OPTIONAL: if present should return a comma-separated list of
URLs (commas should be URL-encoded)
  "relevance" : "string", // OPTIONAL: String/script: Must specify/return a
double/string-parsable-into-a-double
  "sentiment" : "string", // OPTIONAL: String/script: Must specify/return a
double/string-parsable-into-a-double, by convention this is between -1.0 and 1.0.
  "frequency" : "string", // OPTIONAL: Must specify/return a
long/string-parsable-into-a-long
  "geotag" : { // OPTIONAL: Format is identical to the docGeo format specified above
    "lat": "string", "lon": "string",
    "city": "string", "stateProvince": "string", "country": "string", "countryCode":
"string"
  },
  "ontology_type": "string", // OPTIONAL: String/script: Only used if geotag is
specified:
  // allows specification of the scale of the geographic entity (see below for
useful link), defaults to "point"
  "useDocGeo": "boolean", // OPTIONAL: If true, uses any lat/long generated from the
top level "docGeo" specification, defaults to false
  "creationCriteriaScript" : "string", // OPTIONAL: script: If populated, runs a user
script function and if return value is false doesn't create the object
  "entities" : [ { ... } ] // If iterateOver is specified, use this array to create
multiple entity types per iteration (NOW DEPRECATED, use dot notation in
"iterateOver")
}
```

More information on the ontology type is provided [here](#).

Associations

The association format this specification object generates is documented [here](#).

Source.structuredAnalysis.events object

```
{
  "iterateOver" : "string", // OPTIONAL: If specified as a list of entity types, steps
over entities with matching types (again, lock step or combinatorially)
  // Can also specify a metadata field (nesting supported using dot notation), in
which case they are looped over to generate calls with _value/_iterator/_index
  "entity1" : "string", // OPTIONAL: String/script: In "iterateOver"/type cases, the
disambiguated name of the entity type; otherwise using entity1_index is preferred.
  "entity1_index" : "string", // OPTIONAL: String/script: should return the
'disambiguated_name/type' string, must resolve to an entity or is discarded
  "entity2" : "string", // OPTIONAL: String/script: In "iterateOver"/type cases, the
disambiguated name of the entity type; otherwise using entity1_index is preferred.
  "entity2_index" : "string", // OPTIONAL: String/script: should return the
'disambiguated_name/type' string, must resolve to an entity or is discarded
  "verb" : "string", // MANDATORY: String/script
  "verb_category" : "string", // MANDATORY: String/script
  "assoc_type" : "string", // MANDATORY: Must specify/return one of "Fact", "Event",
"Summary" and be overridden (eg converted to summary if there is only 1 index)
  // (if left blank, the Structured Analysis Handler will auto-generate this field
reasonably accurately based on the contents)
  "time_start" : "string", // OPTIONAL: String/script: Must specify/return a time in
ISO date format ("yyyy-MM-dd'T'HH:mm:ss") or Javascript time format
  "time_end" : "string", // OPTIONAL: String/script: Must specify/return a time in ISO
date format ("yyyy-MM-dd'T'HH:mm:ss") or Javascript time format
  "geo_index" : "string", // OPTIONAL: String/script: The entity index corresponding to
the "geotag" below (or the Type in "iterateOver" cases)
  "geotag" : { // OPTIONAL: Format is identical to the docGeo format specified above
    "lat": "string", "lon": "string",
    "city": "string", "stateProvince": "string", "country": "string", "countryCode":
"string
  },
  //(note the ontology_type for associations is always "point" - use geo_index to
specify larger areas)
  "creationCriteriaScript" : "string", // // OPTIONAL: script: If populated, runs a
user script function and if return value is false doesn't create the object
  "sentiment" : "string", // OPTIONAL: String/script: Must specify/return a
double/string-parsable-into-a-double, by convention this is between -1.0 and 1.0.
  "associations" : [ { ... } ] // If iterateOver is specified, use this array to create
multiple association types per iteration (NOW DEPRECATED, use dot notation in
"iterateOver")
}
```

UnstructuredAnalysis object

JSON format

Note that there is a separate overview of using the Unstructured Analysis Harvester. This page is reference information.

The UnstructuredAnalysis object of the [Source](#) document:

Source.unstructuredAnalysis object

```
{
  "headerRegex" : "string", //regular expression string that represents the header of
the document, see overview
  "footerRegex" : "string", //regular expression string that represents the header of
the document, see overview
  "simpleTextCleanser": [ { ... } ], //complex object (see below) transforming the
text before metadata/entity extraction

  "meta" : [ { ... } ] //definition for objects to parse and place into the source
metadata (see below)

  "caches": { "string": "string", ... } // A list of caches in the format
<CACHE_NAME>:<ID> where <ID> is the "_id" of a JSON share, see overview
}
```

simpleTextCleanser object

simpleTextCleanser specification

```
{
  "field": "string", // The document field to transform, one of: "title",
"description", "fullText", "metdata.<field>"
  "script": "string", // The regex to match the section of text to be modified (other
scripts may be supported in the future)
  "scriptlang": "string", // Optional and currently unused, defaults to regex
  "replacement": "string", // The replacement string, can include replacement groups
in standard Java syntax
  "flags": "string" // The fields (standard Posix/Java, eg "i" for case-insensitive),
*** with additions: 'H' HTML-decodes the resulting string
}
```

Note that the order of "text cleansing" vs all other operations is as follows:

- For RSS objects, get the raw content via HTTP
- Identification of header, footer, body on raw content
- Processing of "meta" objects (see below) with "First" contexts
- All text cleansing operations
- Processing of "metadata" objects (see below) with all other contexts ("Body", "Header", "Footer", "All")

Meta object

Source.unstructuredAnalysis.meta object

```
{
  "fieldName" : "string", //name used to define the metadata (see metadata link
below)
  "context" : "string", // one of "First", "All", "Body", "Header", "Footer": used to
specify location the meta data should be found

  "script" : "string", // either javascript or regular expression (based on
"scriptlang") used to generate the metadata
  "scriptlang": "string", // "javascript", "regex", or "xpath"

  "replace" : "string", // when set, regular expression matches will be replaced with
this string (can include groups, eg $1 or group 1, note $0 is the whole string)
  "groupNum" : int, // A quick alternative to using replace, eg "groupNum":2 is
equivalent is "replace": "$2". "groupNum" has a special meaning for xpath, see below.
  "flags" : "string" // for regexes, the flags to apply (standard Posix/Java flags).
"flags" also has meanings for "javascript" and "xpath", and "xpath"/"regex", see below
}
```

The metadata object is described [here](#). In the javascript case, if an array of objects are returned, that array is embedded into the "metadata" map; if a single object is returned, it is embedded inside a single-element array (ie consistent with how all metadata objects are treated).

As described above, the regex/javascript is applied before text cleansing if the context is "First", faster otherwise.

Examples are provided in the [overview of using the Unstructured Analysis Harvester](#).



By default XPath and Regex fields are deduplicated, ie if the string "apple" is found twice for the same field name, then it is not added to the field array. In cases where multiple fields are being correlated based on index, this is obviously not desirable, and it can be turned off by setting the flag 'D' ("Don't DeDuplicate").

JavaScript "meta" fields

If no flags are specified for a javascript "meta" field, then the following objects are available in the javascript:

- **text**: The full text of the document
- **_iterator**: An array consisting of the current value of that metadata field (null if the field does not currently exist). This object allow successive scripts with the same "fieldName" to perform a processing chain. It also allows the Unstructured Analysis Harvester to modify document-level metadata (eg from the [Feed Harvester](#), [File Harvester](#), or [Database Harvester](#)).

If the flags string is specified, it is a character sequence, with characters interpreted as follows:

- **t**: provides the full text of the document to the script, in the field "text"
- **d**: provides the entire document object to the script, in the field "_doc"
- **m**: provides the entire metadata object ("_doc.metadata") to the script, in the field "_metadata".

XPath "meta" fields

The following flags are supported for XPath:

- **H**: will HTML-decode resulting fields. (Eg "&"; -> "&")
- **o**: if the XPath expression points to an HTML (/XML) object, then this object is [converted to JSON](#) and stored as an object in the corresponding metadata field array. (Can also be done via the deprecated "groupNum":-1)
- **D**: described above (also works for regex)

"groupNum"/"replace" are used for 2 purposes:

- If the "regex" extension is used (see below for explanatory link), then groupNum/replace are used to select the capturing group just like for a normal regex "meta" field.
- If "groupNum" is set to -1, and the XPath expression points to an HTML (/XML) object, then this object is [converted to JSON](#) and stored as an object in the corresponding metadata field array.
 - (Note this is deprecated, it is now recommended to use flags: 'o')

XPath support is discussed further [here](#).

Widget Framework

The Infinit.e Widget framework is available to assist in the creation of visualization applications using the Infinit.e APIs. It is a set up tools and documentations that make this process more efficient.

Getting Started

- Setting up your environment. See [Infinit.e Widget and Plugin Use](#)
- Writing your first widget. See [Writing your first Infinit.e Widget](#)
- Special widget information. See [Special Widget Information](#)
- Uploading the completed widget to the GUI framework.
- The complete Widget API.
- Interacting with the Infinit.e API. See [Infinit.e API Documentation](#)
- [Override the GUI setup from the URL](#)

Different views of the data provided to the widget

Overview

Since there are a number of ways of retrieving queried data from an [IWidgetContext](#) or [IResultSet](#), this page provides a brief overview of the concepts involved and then explains what data each of the calls will retrieve.

Different documents sets

First off, an explanation of the difference between:

- Matching documents
- Top documents
- Filtered documents

When a query is issued, often a large number of [documents](#) will satisfy the query criteria (particularly for a common query like "obama"), these are called **matching documents**. These documents are not directly available to the widget (apart from **top documents**, see below), except

However there are normally too many documents for a person to analyze directly (see below, under aggregations). As a result, a ranked subset of these **matching documents** (according to a [configurable scoring method](#)) is retrieved and only these are returned directly to the GUI. The default number of these **top documents** returned is 100.

The [IWidgetContext](#) API allows for further filtering of these **top documents** within the GUI framework, eg containing a specific set of entities (eg click on one of the bars in the graph in the "Significance" widget). This sub-set is called the **filtered documents**.

Aggregations

Obviously all the **matching documents** can contribute to the "knowledge" that a query can provide, and the documents themselves are not the only objects returned from a query. Instead, relevant information to the analysis is summed/averaged/etc ("aggregated") across all matching documents, and these are referred to as the "aggregations". Examples include:

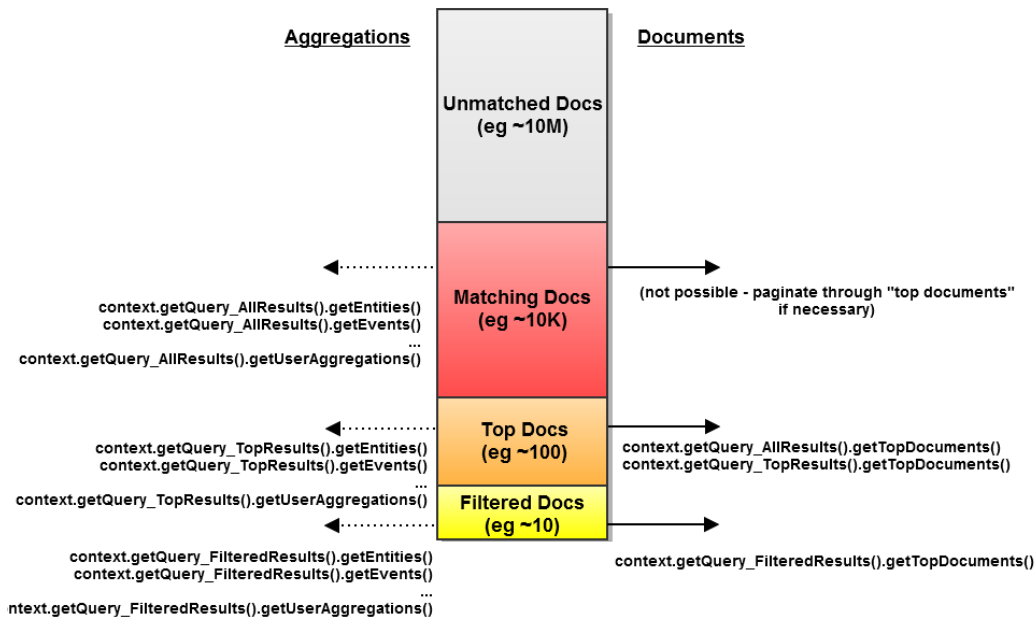
- **Geo:** lat/longs and their frequency in the document set
- **Times:** number of documents per period (day, week, etc) in the document set
- **Entities:** [entity objects](#) found in the document set, ranked by [significance](#).
- **Events:** [event objects](#) found in the document set, ranked by frequency.
- (etc)

It should be noted that [aggregations are sometimes ranked by frequency](#), sometimes summed by significance. This distinction will become more consistent in future versions of the tool.

Finally, note that the idea of an aggregation is valid across all three of the documents sets described above (matching, top, filtered). The [IWidgetContext](#) API lets you select which of the document sets to aggregate over, as shown in the diagram below.

Visual summary

The following diagram provides a visual breakdown of the matching/top/filtered documents described above, and shows which [IWidgetContext](#) call returns data from which documents.



Infini.e Widget and Plugin Use

This space defines how to use the Eclipse plugin to easily develop widgets that are usable in the Infini.e tool.

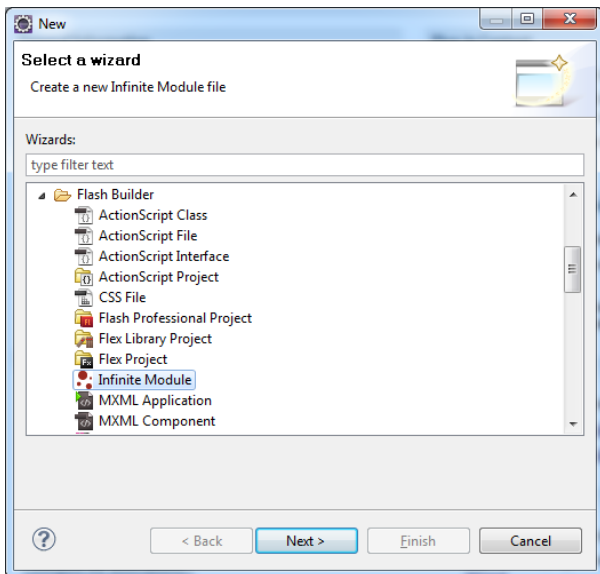
Tools needed: [Eclipse IDE](#), with integrated [Flash Builder](#) with Flex SDK 4.5.x

- In theory Flex SDK 4.6 is compatible, though will require a compiler flag to be set (this is unconfirmed by us)

Files needed: [Infini.e Eclipse plugin](#) and [Infinite Widget Library](#)

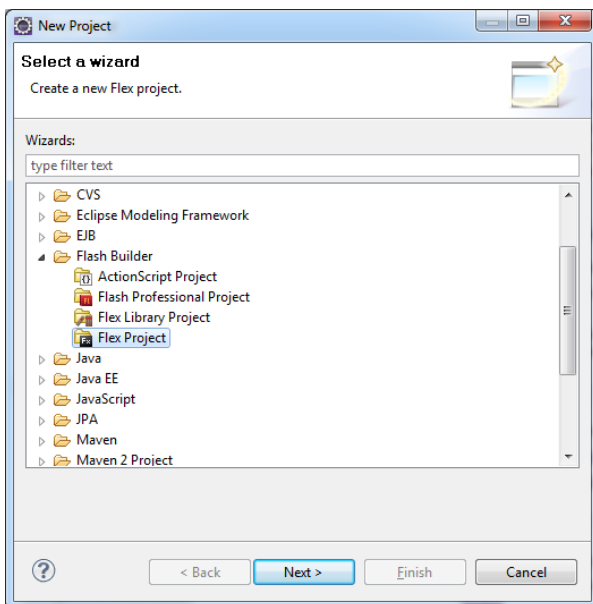
Infini.e Eclipse Plugin

1. Download the [IWD:infini.e plugin](#) shown in the link above.
2. Install the plugin by dropping the file `infini.e.adobe.eclipse.infiniteModule.plugin_x.x.x.x.jar` into your eclipse plugin folder e.g. for windows users: `C:\eclipse\plugins`
3. Restart or open eclipse, to test plugin is working right click on a project > click New > Other and then under the wizard list there should be Infinite Module under the Flash Builder folder (also Infinite Module has its own folder, both do the same action)

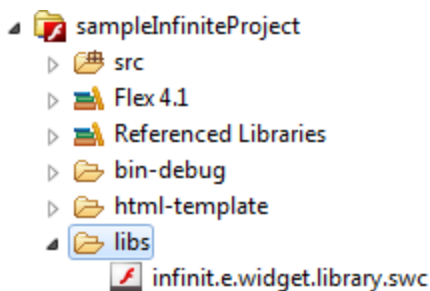


Infinite Widget Library

1. Download the [IWD:infinite widget library](#) shown in the link above. (Already complete if you did the step for the eclipse plugin)
2. Create a Flex project in your eclipse workspace (or you can use an existing flex project) Go to File > New > Project... Select Flex Project under the Flash Builder folder, hit next, give the project a name and select Finish



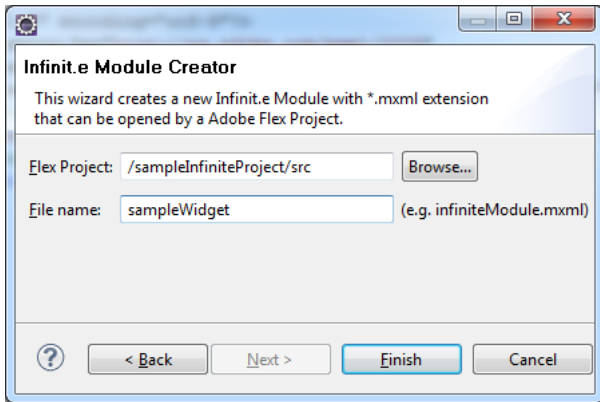
3. Drag the `infinite.widget.library.swc` file and the `flexlib.swc` file from the `InfinitePluginAndLibrary.zip` file and drop them onto the `libs` folder in your new project



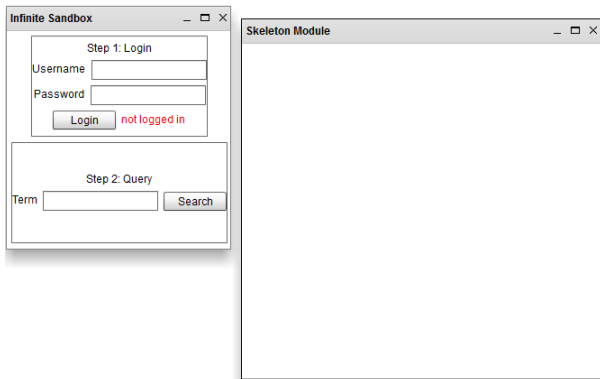
4. Eclipse should have automatically added the swc library to the build path as you can see in the Referenced Libraries

Creating a Widget

1. Right click on the src folder of your Flex Project and select New > Other choose Infinite Module from the Flash Builder folder
2. Give your widget a name and select finish. (If the wizard does not pick up the src folder automatically, select Browse and navigate to your projects src folder e.g. /projectname/src)



3. The wizard should automatically add 2 files to your project a sandbox to run your new widget in named InfiniteTestApplication.mxml and your widget which you named in the previous step (e.g. sampleWidget.mxml)
4. At this point you can run the example by right right clicking the InfiniteTestApplication.mxml selecting Run As > Web Application



5. Add your widgets logic into the sampleWidget.mxml file (or whatever you named your widget) and run sample queries by logging in and then searching on a term.

Alternative Method

As an alternative you can import [IWD:this pre-made project](#) directly into Eclipse. To do so, in eclipse go to File > Import, under General choose Existing Projects into Workspace, Click the radio button for Select archive file, then click browse and navigate to the zip folder you just downloaded ([widgetSampleProject.zip](#)) and click Finish. This should open a flex project named widgetSampleProject into your workspace with pre-made files ready to run.

Override the GUI setup from the URL

By default when the GUI is loaded, the session state from the last time the GUI was used is reloaded (from the [GUI - UISetup - Get API call](#)).

As an alternative, the saved state can be piecewise overridden with the following URL parameters:

- **query**: Pass a URL encoded **JSON query object** with the query to be executed. "qt" is mandatory, the others are optional (and will be taken from the default settings if not specified), except:
 - **userSettings**: If this parameter is present in the URL string then non-specified fields are taken from the saved query instead.
- **communityIds**: A list of "_id" fields for the communities to query over (comma-separated; no spaces; eg can be obtained from the [Social - Person - Get API call](#); note that "4c927585d591d31d7b37097a" is hardwired as the system community).
- **widgetIds**: A list of the "_id" fields for the widgets to display (up to 4 can be specified, comma-separated, no spaces; eg can be obtained from the [Social - GUI - Modules - Get API call](#)).
 - Note the following widget ids are typically hardwired:
 - Doc Browser: "4e4bcd7addda4e72000eeb76"
 - Entity Significance: "4e4bf8adbce84e723f3b297d"
 - Sentiment Widget: "4e4bfa4cbce84e724b3b297d"
 - Event Graph: "4e4bf9ecbce84e72483b297d"
 - Timeline: "4e4bf8f5bce84e72423b297d"
 - Event Timeline: "4e4bfa90bce84e724e3b297d"
 - Query Metrics: "4e4bf986bce84e72463b297d"
 - Map Widget: "4e4bf986bce84e72453b297d"

Example URLs for custom setups

```
http://infinite.ikanow.com/Index.html?query={%22qt%22:[{%22ftext%22:%22obama%22}]}&communityIds=4c927585d591d31d7b37097a&widgetIds=4e4bcd7adda4e72000eeb76
http://SERVER_IP_ADDRESS/Index.html?query={%22qt%22:[{%22ftext%22:%22obama%22}]}&widgetIds=4e4bcd7adda4e72000eeb76,4e4bf8adbce84e723f3b297d
http://SERVER_IP_ADDRESS?query={%22qt%22:[{%22ftext%22:%22obama%22}]}&communityIds=506afc85e4b01d98fcf9bf5f,4c927585d591d31d7b37097a
http://infinite.ikanow.com/Index.html?query={"qt":[{"ftext":"obama"}]}
```



There is currently one other URL parameter supported by the Infnit.e GUI: "redirect": if this is used the following successful login (or directly if already logged in) then the browser tab is redirected to the value specified by the URL parameter. This can be useful for single sign-on applications.

Special Widget Information

Styling

To ensure a similar experience between browsers, operating systems, and other environment factors we have included a default font into Infnit.e. Because of the way widgets are dynamically loaded modules, if you want to take advantage of the global font widgets must include the css stylesheet we use to style all font. If you use the eclipse plugin or predefined project in the getting started section, these are created for you and included in your widget automatically. If you choose not to use the Infnit.e font you are more than welcome to unreference the stylesheet and go with your own approach. The stylesheet included with your project is just an example and is not uploaded with your project when it is being submitted, so do not make changes to that stylesheet as they will not be reflected when uploaded your widget to the main Infnit.e application.

The stylesheet we have included styles all spark components and a selection of common MX components. If you include a MX component that we have not styled in our stylesheet, you have the option to add a style to your widget individually. If using the eclipse plugin or predefined project there is a commented out example in the <fx:Style> tag of adding the font to a mx:Text component. an example is also shown here:

```
<fx:Style>
mx|Text
{
    font-family: infiniteNonCFFFont;
}
</fx:Style>
```

Widget Saving - Community vs User Options

Typically a widget will save any options that it wants on reloading during a onSaveWidgetOptions call. Any json object passed to the widget framework on that call will be returned when the widget is loaded via onLoadWidgetOptions in a WidgetSaveObject. The WidgetSaveObject has 2 interesting fields, userSave and communitySave. The userSave object is the standard object that was saved during a onSaveWidgetOptions call and should be used for any save options that are unique to an individual user; for example their current zoom level and centerpoint on map widget or the graph type on a statistics widget.

Widget Saving Example

```
//Here is an example of creating an anonymous object (tempWidgetOptions)
//and setting some fields to save the zoom and centerpoint of a map widget
//This method will be polled every few minutes
public function onSaveWidgetOptions():Object
{
    var tempWidgetOptions:Object = new Object();
    tempWidgetOptions[ "centerLat" ] = _map.center.lat;
    tempWidgetOptions[ "centerLng" ] = _map.center.lng;
    tempWidgetOptions[ "zoomLevel" ] = _map.zoom;
    return tempWidgetOptions;
}

//The next time a widget is opened it will be sent the anonymous object
//that was last saved from onSaveWidgetOptions
//that object can be used to restore previous states
public function onLoadWidgetOptions( widgetOptions:WidgetSaveObject ):void
{
    if ( widgetOptions != null )
    {
        //this holds a users last map
        if ( widgetOptions.userSave != null )
        {
            this.widgetOptions = widgetOptions.userSave;
            //widgetOptions contains fields centerLat,centerLng,zoomLevel from our
last save
        }
    }
}
```

The communitySave object is used for giving a widget specific saved data for a given community, for example giving an intelligence community KML specific to a region of interest for a map widget or using a certain color scheme specific to business operations for a statistics widget. The communitySave objects can only be set external to a widget in <http://infinite.ikanow.com/manager/fileUploader.jsp> To make a communitySave in the fileUploader just upload a json file with a community selected and use type widgetSave, any json available to a user when a widgetLoads will be passed into the WidgetSaveObject.communitySave when onLoadWidgetOptions is called from then on.

Loading Community Save Data

```
public function onLoadWidgetOptions( widgetOptions:WidgetSaveObject ):void
{
    if ( widgetOptions != null )
    {
        //this holds a users last map
        if ( widgetOptions.userSave != null )
        {
            //...code from above here
        }
    }
    if ( widgetOptions.communitySave != null )
    {
        //Any shares you have access to with type widgetSave will be provided here
        //Here we will be given a map of community ids to shares e.g. { "commid12345":
{"key1":"value1"}, "commid67890":{"anotherkey",["blue","green","red"]}}
        for ( var commid:String in widgetOptions.communitySave )
        {
            //this is the object saved in the share for community <commid>
            //Now you can do what you want with it (we printed it out)
            var community_save_object = widgetOptions.communitySave[commid];
            for ( var key:String in community_save_object )
            {
                trace("CommId: " + commid + " key: " + key + " value: " +
community_save_object[key]);
            }
        }
    }
}
}
```

Widget Drag and Drop

A generic drag and drop interface is implemented for sending document, entities, and associations between widgets. To accept these things a widget only needs to implement an event handler for the widgetDrop event on WidgetModule

```

<components:WidgetModule xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:components="com.ikanow.infinite.widget.library.components.*"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx"
  implements="com.ikanow.infinite.widget.library.widget.IWidget"
  widgetDrop="widgetmodule1_widgetDropHandler(event)">

<fx:Script>
  <![CDATA[
import com.ikanow.infinite.widget.library.events.WidgetDropEvent;

protected function widgetmodule1_widgetDropHandler(event:WidgetDropEvent):void
  {
    trace("Ents: " + event.entities.length);
    trace("Assocs: " + event.associations.length);
    trace("Docs: " + event.documents.length);
    trace("Source: " + event.dragSource);
    trace("WidgetName: " + event.dragWidgetName);
    trace("WidgetClass: " + event.dragWidgetClass);
  }
  ]]>
</fx:Script>
</components:WidgetModule>

```

The WidgetDropEvent object holds 3 anonymous arrays: entities,associations,documents, any/all of these may have data in them. It also holds some information on where the drag came from:

dragSource: the dragger manually specifies this, can be anything

dragWidgetName: the title of the widget the drag came from

dragWidgetClass: the class of the widget the drag came from (i.e. the mxml file name of WidgetModule)

To send dragged items to another widget, dispatch an drag event with the dataformat using WidgetDragUtil.WIDGET_DRAG_FORMAT:

```

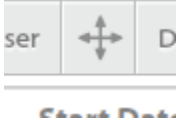
protected function widgetheaderdragimage1_mouseDownHandler(event:MouseEvent):void
{
  var docs:Array = new Array();
  for each ( var doc:Object in docList.selectedItems )
  {
    docs.push(doc);
  }
  var dragObject:WidgetDragObject = new WidgetDragObject();
  dragObject.documents = new ArrayCollection(docs);
  dragObject.dragSource = "MyCustomWidgetName"; //this is the user made drag name, can
be anything
  var ds:DragSource = new DragSource();
  ds.addData(dragObject, WidgetDragUtil.WIDGET_DRAG_FORMAT );
  DragManager.doDrag(dragImage, ds, event);
}

```

A widget header icon has been supplied in the widget library to allow uniformity between widgets, this icon can be added to a widget by adding it to the header block on all widgets (and implement the mousedown handler as shown above, supplying your specific data):

```
<components:WidgetHeaderDragImage id="dragImage"
mouseDown="widgetheaderdragimage1_mouseDownHandler(event)" tooltip="Drag this to
another widget/the query bar to send selected documents" />
```

When added to a widget it will look like:



Widget Framework - uploading widgets

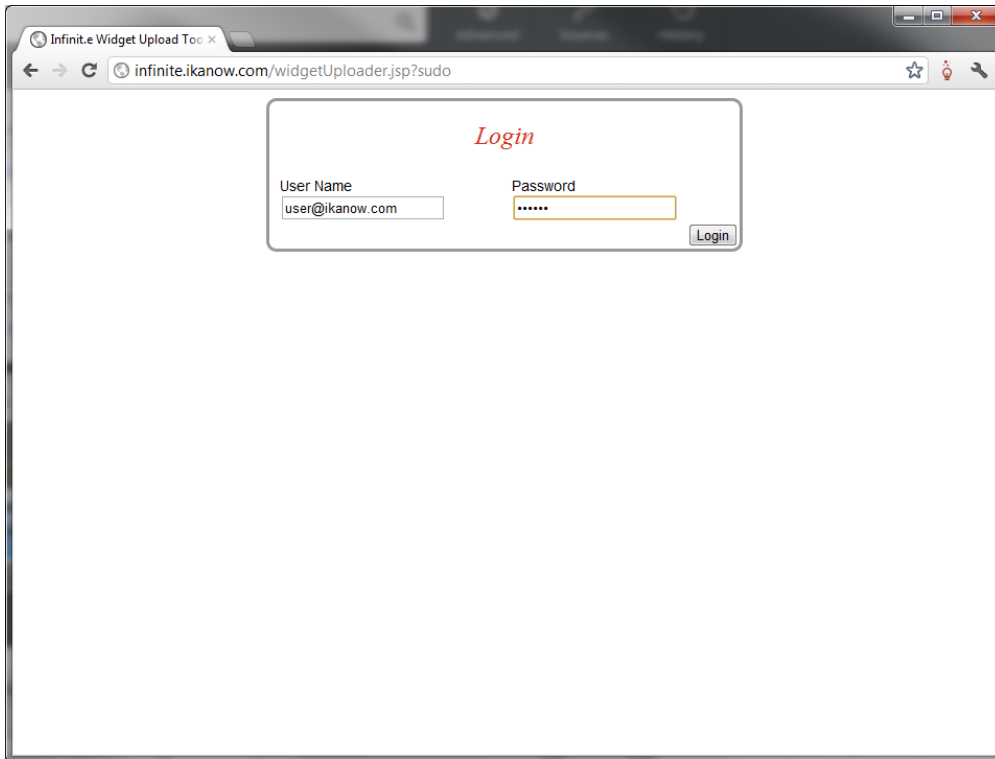
Overview

The widget uploader provides a simple web-based user interface for uploading new widgets to the system, and for managing existing widgets.

Logging In

The widget uploader can be accessed from `<ROOT_URL>/manager/widgetUploader.jsp` (eg <http://infinite.ikanow.com/widgetUploader.jsp>). It can also be reached from the home page of the [Manager webapp](#) (itself linked from the main visualization GUI).

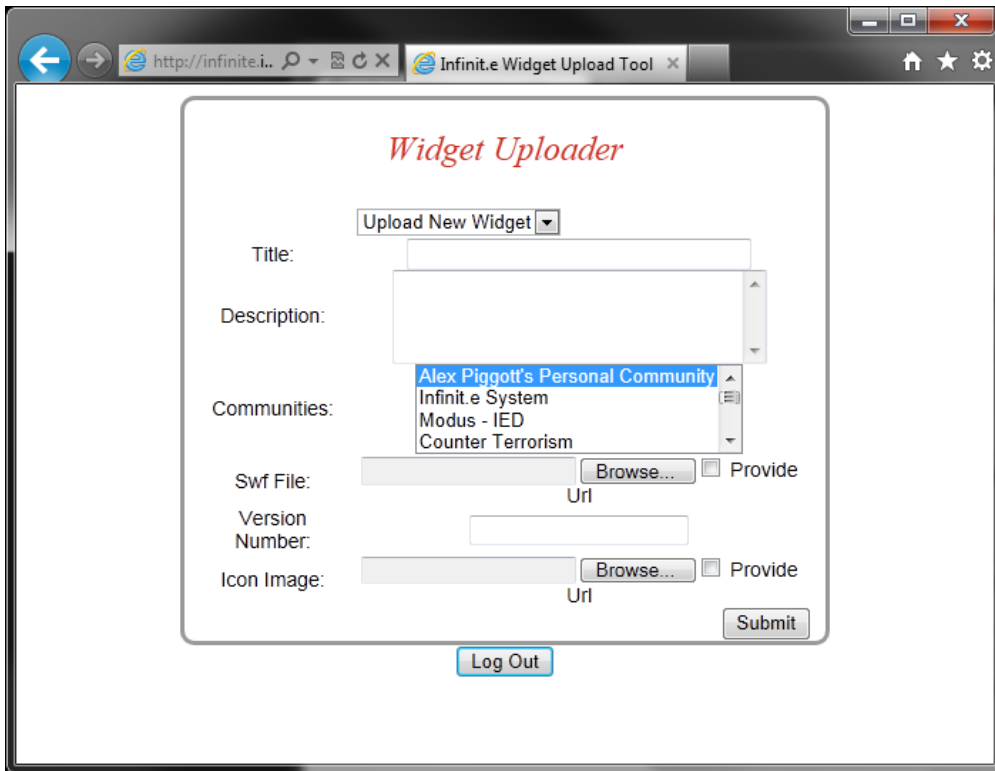
This brings up username and password fields and a login button. (Unless already logged in, eg into the manager or main GUI - in which case skip to the next section).



Two things to note:

- In order to login with admin privileges (for admins only!), append "**?sudo**" or "**?sudo=true**" to the URL (eg <http://infinite.ikanow.com/widgetUploader.jsp?sudo>).
 - *If you are not logged-in with admin privileges you will only be able to edit your own widgets, and will only be able to add widgets to communities you own or on which you are a moderator.*
- The widget uploader shares its cookie with the main GUI, the file uploader, the source builder, and the person manager - logging into any of them will log into all of them.

Uploading a new widget

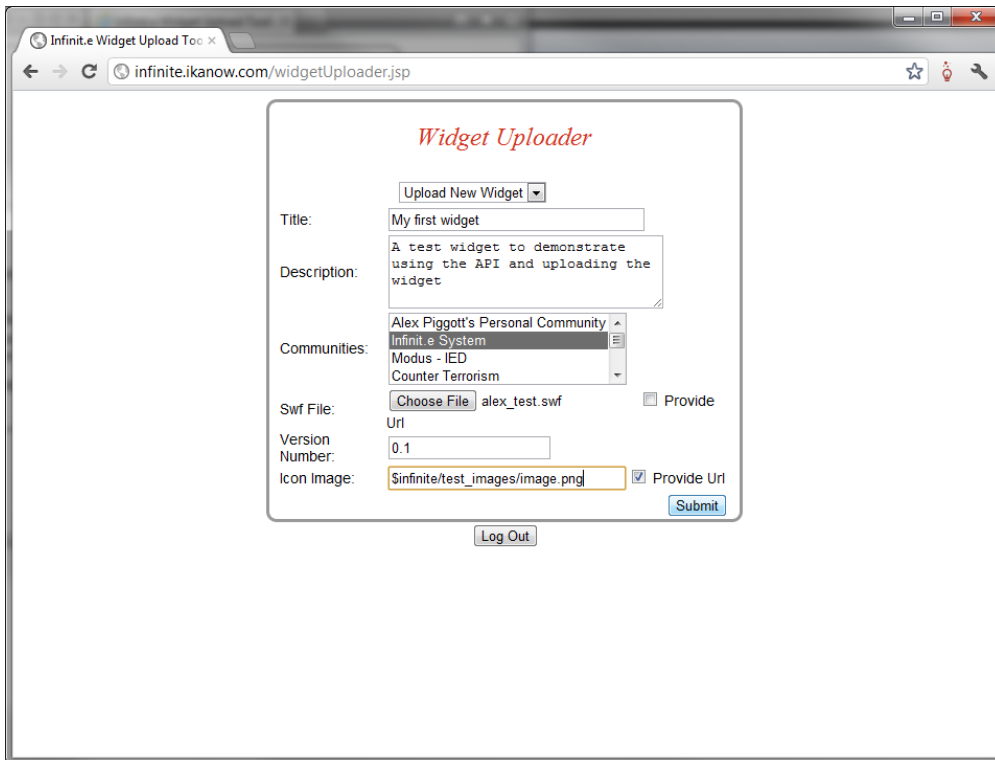


The above figure shows the tool immediately after log-in.

To upload the widget, simply populate the fields shown above and then "submit". Some further information on fields is described below:

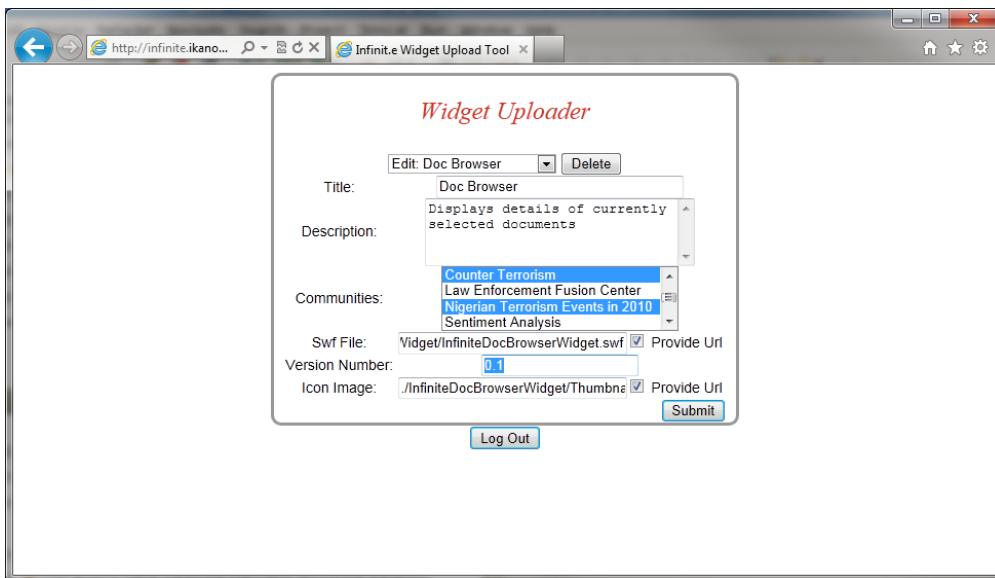
- All fields are mandatory except the description field (which is used for the mouse-over pop up in the main GUI)
- There are 2 ways of providing the ".swf" for the widget (and also its icon):
 - Upload them from your hard disk using the "Browse..." buttons.
 - Select the "Provide URL" checkbox and then enter a URL into the "Swf File" or "Icon Image" filenames
 - You can use the substitution variable "\$infinite" to point to whatever the root URL is (eg "infinite.ikanow.com"), see the example screenshot below.
- The icon image size should be 180x160 for best results.
- The version number field is not used by the system, it is for your configuration management purposes only.
- Note that no de-duplication checks are made, so if you enter an identical widget to one that already exists, you will get 2 widgets that look the same. To edit an existing widget, see below under the next section.
- You can select either a single or multiple (CTRL+click) communities. You must be the owner or moderator of the communities (or an administrator logged in with "?sudo").

An example is shown below:



Editing widgets

To edit the attributes of an existing widget, log-in as before but then select one of the widgets from the drop-down menu at the top of the screen:



(Remember that administrators must log-in with "?sudo" appended to the URLs in order to edit other people's widgets).

Once an existing widget has been selected, its current fields are used to populate the web form. The communities with which the widget is currently shared are highlighted. The non-highlighted communities are those with which the widget can be shared.

Simply edit whichever fields need to be changed (or upload new files) and select the "Submit" button.

Copying widgets

To copy a widget (eg to create multiple widgets with similar fields), simply select the widget to copy in the drop down menu at the top, and then choose the "Copy Current Widget" item in the same menu. Change the title and then edit the fields as normal.

Deleting widgets

To delete a widget, log-in and select it like for editing above, but then press the "Delete" button at the top.

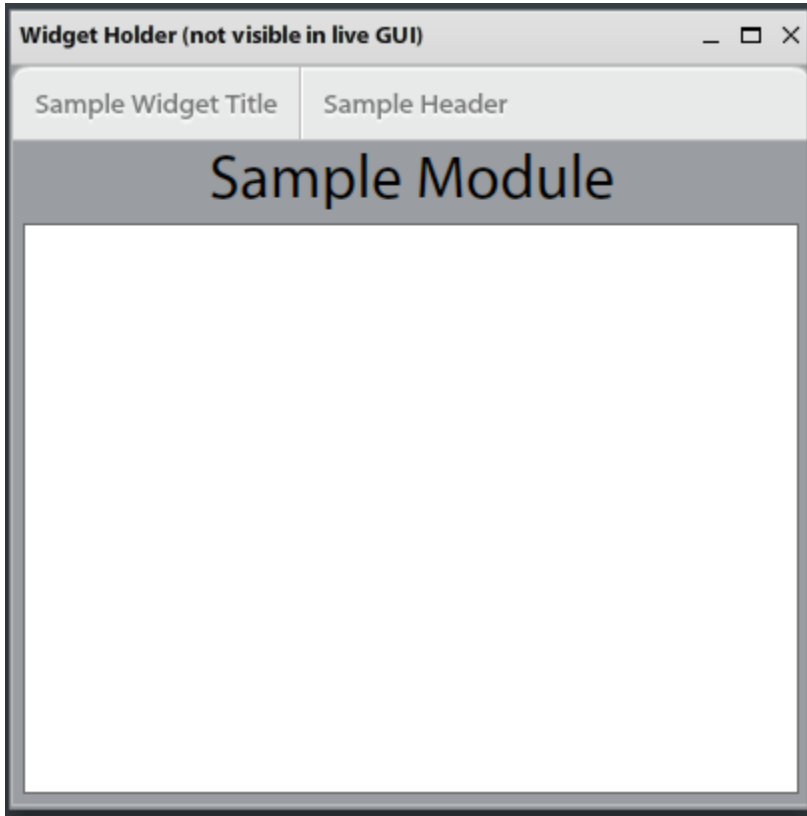
Writing your first Infit.e Widget

This page will give you some example of how to start your first widget and breakdown of how the Infit.e Widget framework interacts with your widget.

This guide is written to be picked up from the end of the [Getting Started tutorial](#) for getting set up with the Eclipse Plugin and Widget Library.

Overview of an Infit.e Widget

An infit.e widget is a flex module that gets run inside a window in the infit.e application.



The Infit.e Webpage handles all interactions with the window such as moving, dragging, changing size, hiding/showing, and data transfer. You are responsible for handling the content inside the widget.



The plugin sandbox application (InfiniteTestApplication.mxml) displays widgets inside a movable/resizable window. This window is not present when deployed in Infit.e and is only for testing your widget at different window sizes.

A widget must implement the [com.ikanow.infit.e.widget.library.widget.IWidget](#) interface which is included in the [infit.e.widget.library.swc](#) library. This involves implementing the following functions: `onInit(IWidgetContext)`, `onReceiveNewQuery()`, `onReceiveNewFilter()`, and `onParentResize(Number,Number)`. The first 3 functions are for data transfer while the last is used for resizing the content you create.

If you are using the Infit.e Eclipse Plugin then these methods will already be stubbed out for you with sample code.

Overview of function to implement

The [IWidget](#) interface requires you to implement a number methods, which will be called from the framework. We will go over each method in detail below.

Note that all these methods are stubbed out to do nothing by the plug-in, therefore only required functionality need be implemented. (All the functions must be present for the module to compile).

Mandatory

- **onInit(context:IWidgetContext)** - This function is called when your widget has completed loading into the Infit.e Framework. The current data context object is passed to the widget which you will want to store locally. This [IWidgetContext](#) object contains methods to access the current result sets from the query, filter, and other associated information. This method has been stubbed out if you are using

the Infinit.e Eclipse Plugin and no further action needs taken.

- **onReceiveNewQuery()** - This function is called by the framework when new data has been loaded into the `IWidgetContext` object. This is your notice that the dataset has changed and you can display new content in your widget. Some sample code has been stubbed out in the Infinit.e Eclipse Plugin that shows how you can get the query result set from the context object (passed to you on the `onInit()` function) and then that data can be iterated over to access various fields in the document set. An example is explained below on using query or filter data. TODO
- **onRecieveNewFilter()** - This function is similiary to the `onReceiveNewQuery` function except it is called when a filter has been put on the query data set (even if this filter was applied by this widget). It can be used similarly to the `onReceiveNewQuery` function and an example can be seen below.

Optional

- **onParentResize(newH:Number,newW:Number)** - This function is used to resize data in your widget. When the parent window the widget is housed in gets resized, it will let the module inside know that it has changed width and height so you can change your data accordingly. If you are using the Infinit.e Eclipse Plugin this method has stubbed out an example for you that will set this modules width and height to the incoming width and height so the parent and internal module will stay the same size. Some examples where you may not want to do this is if you want to instead scale the data in your window so it is always shown or resize in another manor.
- **onSaveWidgetOptions():Object** - This function is called periodically by the framework. The user can return an arbitrary JSON object (which is equivalent in ActionScript to an anonymous object, eg `"var json:Object = { 'test': 'string', 'numbers': [1, 2, 3] };"`) which is stored by the framework and returned when the object is re-opened. This allows developers to save the widget state (note the size and position on the framework canvas is stored by default), such as level of zoom, which graphs are displayed, etc etc.
- **onLoadWidgetOptions(widgetOptions:WidgetSaveObject)** - This function restores the widget state saved by the above callback. Any json object stored using `onSaveWidgetOptions` will be returned here (this can include a community save object)

Advanced

- **supportedExportFormats()** - This function lets developers return an array collection of strings that appears in the widget's context menu. If one of these is selected by the user, `onGenerateExportData` (see below) is called back with the "format" parameter set to whichever string was selected.
- **onGenerateExportData(filename:String, format:String):ByteArray** - This function enables widget developers to generate arbitrary output (normally based on the visualization in the widget, eg KML for a map widget, RDFs for a link analysis widget, etc).
- **onGeneratePDF(printPDF:PDF, title:String):PDF** - This function is a special case of the above data export, allowing use of the AlivePDF library to create PDFs. If this function is left stubbed out then a bitmap snapshot of the widget will be generated instead. Override the function in order to support searchable text in the PDF, different formats etc.

Functionality provided by the framework context

In addition to the above callbacks, the **WidgetContext** object provided to the developer by `onInit` allows a rich set of active operations to be performed, ranging from simple retrieval of documents to complex interactions such as modifying the framework's query builder or even performing local queries.

Future documentation will cover these capabilities in more detail, in the meantime:

- The API documentation is maintained [here](#).
- The sections below provide examples on using this API in order to perform common tasks.
- The source code of the demonstration widgets can be obtained as a source of further examples.

Example of displaying data received from a query or filter

Here we will step through and example of displaying the titles and entities of a query to show how you can display data your widget receives.

1. First we will create 2 lists in our widget, the left list will show the documents titles, and the right list will show all the entities in the resulting dataset.

Lets first create a group to hold our 2 lists so they will align appropriately:

```
<s:HGroup width="100%" height="100%">
  <s:List id="titleList" width="50%" height="100%" dataProvider="{titleArrayList}"
 />
  <s:List id="entityList" width="50%" height="100%"
dataProvider="{entityArrayList}" />
</s:HGroup>
```

Insert this text just below the Module tag in your `sampleWidget`.

Also create the 2 ArrayCollections we are using as dataProviders for the list inside the <fx:Script> block:

```
import mx.collections.ArrayCollection;
[Bindable] private var titleArrayList:ArrayCollection = new ArrayCollection();
[Bindable] private var entityArrayList:ArrayCollection = new ArrayCollection();
```

Now we just need to load some data into these ArrayCollections and the lists will populate so we can show some data. Let's look at the onReceiveNewQuery function.

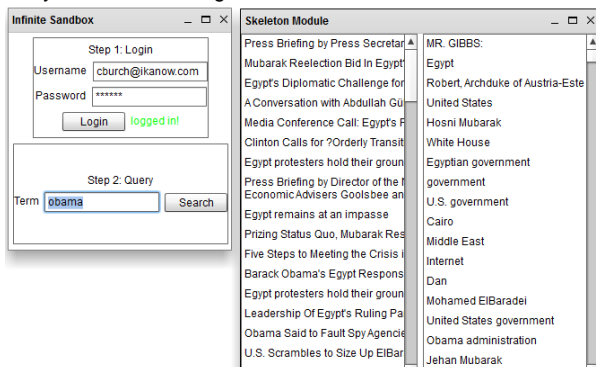
This function is going to get called everytime a query is ran so we know to display new data. Let's first clear our ArrayCollections of any old data, insert this code block at the beginning of the onReceiveNewQuery function:

```
titleArrayList.removeAll();
entityArrayList.removeAll();
```

Next we will loop through the result set we get from the IWidgetContext object we saved in the onInit function and add the documents titles to our ArrayCollection. At the same time we will loop through the document entities adding them to our entity ArrayCollection so we can show their names also: Add this code below the code we just added in the previous step:

```
var queryResults:ArrayCollection = _context.getQuery_AllResults().getTopDocuments();
for each (var doc:Object in queryResults )
{
    titleArrayList.addItem(doc.title);
    for each ( var entity:Object in doc.entities )
    {
        entityArrayList.addItem(entity.disambiguated_name);
    }
}
```

Now we can run our example, login and do a sample query and we should have some data displayed in our lists, all the results titles, and every entity in all the resulting documents!



An alternative that just showed the aggregated entity information might look like:

```
var queryResults:ArrayCollection = _context.getQuery_AllResults().getEntities();
for each (var ent:Object in queryResults )
{
    entityArrayList.addItem(entity.disambiguated_name);
}
```

And so on.

See here for more details about the different views of the data provided by the IWidgetContext class.

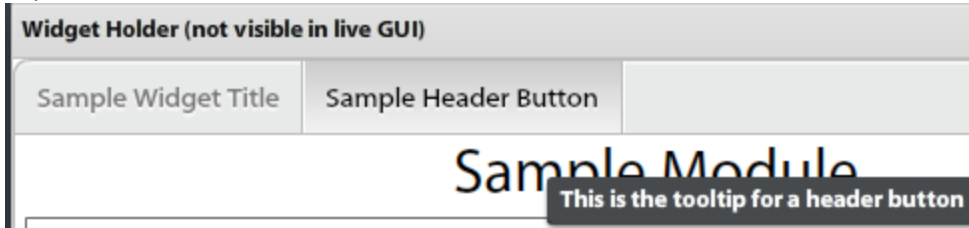
You can download the full code example here or see the widget code below.

Widget Header

The IWidget framework comes with a built in header that is attached to all widgets when initializing. At a minimum the header will have the current title of the widget. (If you are using the plugin, this can be updated by setting the title on your widget in the InfiniteTestApplication.mxml). Any options you want visible to the users at all times can be added to your widget in a section labeled:

```
<components:headerContent>
    <!-- Put components to be displayed in the header here -->
</components:headerContent>
```

The widget library comes complete with some components styled to fit in with the widget theme. A list of all available can be seen in the package `com.ikanow.infinite.widget.library.components.*`. If you have been using the widget plugin then a sample `WidgetToggleButton` is already created for you.



Common suggestions for header items would include things like current graph types such as a toggle between pie or bar charts, a drop down for current map display types such as Terrain, Satellite, or 2d Map or a slider on how many nodes to show on a graph at once.

More advanced operations

Filtering the data visible by widgets

Suppose you want to see quickly only those documents containing a specific set of entities, but don't want to make a whole new query. For example, you have a "Document Browser" widget open (like the first example above), and also an "aggregated event" widget (like the second example above) eg a "Significance" widget, and you want to see all documents in the "document browser" containing the first entity listed.

In the "aggregated event" widget in some callback (eg click on canvas), you would simply write some code like this:

```
var entitiesToFilter:Set = new HashSet();
entitiesToFilter.add(_context.getQuery_AllResults().getEntities().getItemAt(0));
_context.filterByEntities(FilterDataSetEnum.FILTER_GLOBAL_DATA, entitiesToFilter,
EntityMatchTypeEnum.ANY, IncludeEntitiesEnum.INCLUDE_ALL_ENTITIES);
// (Enums mean: (1) filter starting with all data, (2) apply OR to multiple entities
in set, and (3) leave all entities in the resulting document set, not just those in
the filter set)
```

After the final "filterByEntities" call, all active widgets have their "onReceiveNewFilter" callback invoked. This can be used analogously to the "onReceiveNewQuery" example shown above:

```
public function onReceiveNewFilter():void {
    var queryResults:ArrayCollection =
_context.getQuery_FilteredResults().getTopDocuments();
    for each (var doc:Object in queryResults)
    {
        titleArrayList.addItem(doc.title);
        for each ( var entity:Object in doc.entities )
        {
            entityArrayList.addItem(entity.disambiguous_name);
        }
    }
}
```

Note the filtering applies to all widgets, including the one making the call.

A visual example of widget filtering is available [here](#).

Saving the widget state across sessions

By default, when a widget is closed it loses all of its state (for example, in a "document search results" type widget, you might have a drop-down list specifying the number of documents to show per page). The exception to this is the location and size of the widget in the framework canvas.

The "onSaveWidgetOptions" and "onLoadWidgetOptions" provide an easy-to-use capability to store any desired state across sessions (ie closing and then re-opening a widget).

For example, assume a flex "ComboBox" object defined in the MXML, with id="documentsPerPage" (with options "5", "10", "20" and "50", from a bound array "_documentsPerPage"). Then the following code fragment would save this option for the logged-in user:

```
public function onSaveWidgetOptions():Object {
    var json:Object = { documentsPerPage: documentsPerPage.selectedItem.label };
    return json;
}
public function onLoadWidgetOptions(save:WidgetSaveObject):void {
    if (null != save && null != save.userSave) {
        var option:String = save.userSave.documentsPerPage;
        if (null != option) {
            for (var i:int = 0; i < _documentsPerPage.length; ++i) {
                if (option == _documentsPerPage[i]) {
                    documentsPerPage.selectedIndex = i;
                    break;
                }
            }
        }
    }
}
```

Notes:

- "onSaveWidgetOptions" is called periodically by the framework (so shouldn't block)
- "onLoadWidgetOptions" is called after the widget's initialization is complete.
- As can be seen by the code fragments above, the "Object" passed to from the callbacks represents a JSON object.
- See the [Special Widget Information](#) page for a distinction between userSaves and communitySaves.

Adding a query term to the builder

This is a somewhat more advanced use of the [IWidgetContext](#) API, and requires some familiarity with the [JSON query API](#).

Performing a local query

This is also a somewhat more advanced use of the [IWidgetContext](#) API requiring some familiarity with the [JSON query API](#).

Code annex

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Copyright 2012, The Informat.e Open Source Project

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
```

distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

```
-->
<components:WidgetModule xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:components="com.ikanow.infinite.widget.library.components.*"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx"
  implements="com.ikanow.infinite.widget.library.widget.IWidget"
  creationComplete="{ try { onWidgetCreationComplete(); } catch (e:Error) { }
dispatchEvent(new Event('Done Loading'))}; }">
  <fx:Style source="/com/ikanow/infinite/e/assets/styles/infiniteStyles.css" />
  <fx:Style>
    @namespace s "library://ns.adobe.com/flex/spark";
    @namespace mx "library://ns.adobe.com/flex/mx";
    /* If you need to override a style in our stylesheet, or add another
    style that we did not support you can do so here, an example has been commented out
    Please see documentation about over-riding MX component styles to display fonts*/
    /*
    mx|Text
    {
    font-family: infiniteNonCFFFfont;
    }
    */
  </fx:Style>
  <fx:Script>
  <![CDATA[
import com.ikanow.infinite.widget.library.framework.WidgetSaveObject;
import com.ikanow.infinite.widget.library.widget.IWidget;
import com.ikanow.infinite.widget.library.widget.IWidgetContext;
import mx.collections.ArrayCollection;
import org.alivepdf.pdf.PDF;

private var _context:IWidgetContext;

private var titleArray:ArrayCollection = new ArrayCollection();

/**
 * Allow users to export the widget contents in the specified format
 * @format filename: the filename+path to which the data will be written (in case it
needs to be embedded)
 * @param format: the format from the "supportedFormats" call
 *
 * @returns a ByteArray containing the data to output
 */
public function onGenerateExportData( filename:String, format:String ):ByteArray
{
return null;
}

/**
 * This function gets called when the user clicks to output
 * data to a PDF. Return null if custom PDF generation is
 * not desired.
 *
 * @return a new alivePdf Page containing the converted data
 */
public function onGeneratePDF( printPDF:PDF, title:String ):PDF
```

```

{
return null;
}

/**
 * IWidget interface to receive data object (IWidgetContext).
 * Store the iwidgetcontext so we can receive data later.
 */
public function onInit( context:IWidgetContext ):void
{
    _context = context;
}

/**
 * If a save object has been saved from 'onSaveWidgetOptions' then
 * when the app gets reloaded the last save string
 * will be passed to this function.
 *
 * @param widgetOptions the last save object or null if there was none
 */
public function onLoadWidgetOptions( widgetOptions:WidgetSaveObject ):void
{
    //TODO
}

/**
 * function to rescale the component when the parent container is being resized
 *
 * @param newHeight The new height the component needs to be set to
 * @param newWidth The new width the component needs to be set to
 */
public function onParentResize( newHeight:Number, newWidth:Number ):void
{
    this.height = newHeight;
    this.width = newWidth;
}

/**
 * IWidget interface that fires when a new filter is done (including from ourself)
 * We can access the data from the filter by using our
 * iwidgetcontext object _context.getQuery_FilteredResults().getTopDocuments();
 */
public function onReceiveNewFilter():void
{
    //get filtered logic here
    //_context.getQuery_FilteredResults().getTopDocuments();
}

/**
 * IWidget interface that fires when a new query is done.
 * We can access the data from the query by using our
 * iwidgetcontext object context.getQuery_TopResults().getTopDocuments();
 */
public function onReceiveNewQuery():void
{
    //get documents
    var queryResults:ArrayCollection = _context.getQuery_TopResults().getTopDocuments();

    //set the labels of these docs and use as dataprovider for list

```

```

for each ( var doc:Object in queryResults )
doc.label = doc.title;
titlesList.dataProvider = queryResults;

}

/**
 * This function gets called when the workspace is being saved.
 * return null if no save object is needed.
 *
 * @return an object this widget can use to reload state
 */
public function onSaveWidgetOptions():Object
{
return null;
}

/**
 * @returns A list of supported formats, displayed in a context menu in the format
 * "Export <string>" - these are called with "generateExportData"
 * Note this doesn't cover the "built-in" Alive PDF export.
 * However if the developer specifies PDF and generatePdf() returns non-null then this
will be used.
 */

public function supportedExportFormats():ArrayCollection
{
return null;
}

/**
 * The callback handler for clicking the sample button in the header of the app.
 *
 * @param event The mouse event when clicking the button.
 */
protected function sampleButton_clickHandler( event:MouseEvent ):void
{
//perform some action when header button is clicked
}

/**
 * Method fired when module is done loading. Sends
 * message to parent letting it know that module is
 * ready to receive data.
 */
private function onWidgetCreationComplete():void
{
}
]]>
</fx:Script>
<fx:Declarations>
<!-- Place non-visual elements (e.g., services, value objects) here -->
</fx:Declarations>

<!-- If you would like this widget to be styled similar to the other infinite widgets
you may place items in the headerContent section shown below and they will be drawn
at
the top of the widget. If you want to use similar looking buttons explore the

```

com.ikanow.infinite.widget.library.components.* items looking for components prefixed with Widget*. Other components may be added to the header as well.

```
-->
<components:headerContent>
<s:HGroup gap="-3">
<!-- Ignore Filter Toggle Button -->
<components:WidgetToggleButton id="sampleButton"
label="Sample Header Button"
tooltip="This is the tooltip for a header button"
click="sampleButton_clickHandler(event)" />
</s:HGroup>
</components:headerContent>
```

```
<s:VGroup
width="100%"
height="100%"
horizontalAlign="center"
paddingBottom="5"
paddingLeft="5"
paddingRight="5"
paddingTop="5"
verticalAlign="middle">
<s:Label
text="Sample Module"
fontSize="30" />
<s:List id="titlesList"
width="100%"
height="100%" />
</s:VGroup>
```

```
</components:WidgetModule>
```

Third Party Notices

This document contains Third Party Software Notices and/or Additional Terms and Conditions for licensed or open source 3rd party software components included within the current version of Infnit.e software as service (SAAS) product. These notices and/or additional terms and conditions are made a part of and incorporated by reference into the Infnit.e License Agreement. This document is meant to outline obligations regarding third party component utilization.

Third Party Open Source Software Licenses

This portion of the document contains third party software licenses of programs accessed within the Infnit.e software solution.

Alchemy API Terms of Use

The IKANOW Infnit.e application utilizes third party entity extraction engines including alchemyapi to provide content enrichment services. The [alchemyapi](#) services fall under these [terms of use](#).

OpenCalais Terms of Use

The IKANOW Infnit.e application utilizes third party entity extraction engines including opencalais to provide content enrichment services. The [opencalais](#) services fall under these terms of use.



Apache Software License 2.0

Copyright 1999 - 2012 © The Apache Software Foundation
Terms and Conditions for Use, Reproduction, and Distribution

Definitions.

- 'License' shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.
- 'Licensor' shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.
- 'Legal Entity' shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, 'control' means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.
- 'You' (or 'Your') shall mean an individual or Legal Entity exercising permissions granted by this License.
- 'Source' form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.
- 'Object' form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.
- 'Work' shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).
- 'Derivative Works' shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.
- 'Contribution' shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, 'submitted' means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as 'Not a Contribution.'

- 'Contributor' shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a 'NOTICE' text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an 'AS IS' BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets '[]' replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same 'printed page' as the copyright notice for easier identification within third-party archives.

Copyright *yyyy name of copyright owner*

Licensed under the Apache License, Version 2.0 (the 'License'); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an 'AS IS' BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Amazon SDK for Java

Copyright 2010-2012 Amazon.com, Inc. or its affiliates. All Rights Reserved.

ant-contrib

Copyright © 2002-2003 Ant-Contrib Project. All rights Reserved.

as3-commons-collections

Copyright (c) 2010

bigdesk

Copyright 2011-2012 Lukas Vleck

boilerpipe

Copyright (c) 2009 Christian Kohlschütter

collections

Copyright (c) Google

commons

Copyright © 2012 The Apache Software Foundation. All Rights Reserved.

CyberNeko HTML Parser (derived from ***boilerpipe***)

(C) Copyright 2002-2009, Andy Clark, Marc Guillemot. All rights reserved.

elasticsearch

Copyright (c) 2010

elasticsearch-head

contact me via github or on twitter @mobz

elasticsearch-icu

Copyright 2009-2013 Shay Banon and ElasticSearch <<http://www.elasticsearch.org>>

gaforflash

Copyright (c) 2009

geoRSS

Copyright (c) 2006 geonames

gson

Copyright (c) Google

guava

Copyright (c) Google

Hadoop

Copyright © 2011 The Apache Software Foundation

HttpClient

2001-2011, Apache Software Foundation

HttpCore

Copyright © 2005-2013 The Apache Software Foundation. All Rights Reserved

j-calais

Copyright © 2010-2012. All Rights Reserved.

Jackson

© 2003-2008 Codehaus

JASYPT

Copyright © 2011 The JASYPT team. All Rights Reserved.

Kundera

Copyright 2012 Impetus Infotech.

Log4j

© 1999-2010 Apache Software Foundation

Lucene

Copyright © 2011 The Apache Software Foundation, Licensed under the Apache License, Version 2.0. Privacy Policy

Mahout

Copyright © 2011 The Apache Software Foundation - www.apache.org Licensed under the Apache License, Version 2.0.

MongoDB Java driver

Copyright (c) 2009 10gen

MongoDB-Hadoop plugin

Copyright (c) 2009 10gen

opencsv

© 2011 Brought to you by: [glen_a_smith](#), [j_hah](#), [sconway](#), [sullis](#)

Swiz framework

Copyright (c) 2012

ROME**SIGAR**

Copyright (c) Hyperic

StAX

Copyright 2003-2006 - The Codehaus. All rights reserved unless otherwise noted.

tika

Copyright (c) 2004 Apache Software Foundation

tomcat

Copyright © 1999-2012, The Apache Software Foundation

Woodstox

© 2006 Codehaus Foundation.

Xerces

Copyright © 1999-2012 The Apache Software Foundation.

BSD License

As3corelib

Copyright (c) 2008, Adobe Systems Incorporated
All rights reserved.

As3crypto

Copyright (c) 2007 Henri Torgemane
All Rights Reserved.

BigInteger, RSA, Random and ARC4 are derivative works of the jsbn library
(<http://www-cs-students.stanford.edu/~tjw/jsbn/>)
The jsbn library is Copyright (c) 2003-2005 Tom Wu (tjw@cs.Stanford.EDU)

MD5, SHA1, and SHA256 are derivative works (<http://pajhome.org.uk/crypt/md5/>)
Those are Copyright (c) 1998-2002 Paul Johnston & Contributors (paj@pajhome.org.uk)

SHA256 is a derivative work of jsSHA2 (<http://anmar.eu.org/projects/jssha2/>)
jsSHA2 is Copyright (c) 2003-2004 Angel Marin (anmar@gmx.net)

AESKey is a derivative work of aestable.c (http://www.geocities.com/malbrain/aestable_c.html)
aestable.c is Copyright (c) Karl Malbrain (malbrain@yahoo.com)

BlowFishKey, DESKey and TripeDESKey are derivative works of the Bouncy Castle Crypto Package (<http://www.bouncycastle.org>)
Those are Copyright (c) 2000-2004 The Legion Of The Bouncy Castle

Base64 is copyright (c) 2006 Steve Webster (<http://dynamicflash.com/goodies/base64>)

Asx3m

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the **ORGANIZATION** nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ASM (derived from **Jackson**)

Copyright © 1999-2009, OW2 Consortium

HtmlCleaner

Copyright (c) 2006-2013, HtmlCleaner team. All rights reserved.

Java API for KML

Copyright (c) Mictomata

JavaMail

Copyright (c) Oracle

JLine

mwp1@cornell.edu © 2002-2007

Protovis

Copyright 2010 Stanford Visualization Group

Eclipse Public License - v 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ('AGREEMENT'). ANY USE,

REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

'Contribution' means:

1. in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
2. in the case of each subsequent Contributor: i) changes to the Program, and ii) additions to the Program;
where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: **i** are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

'Contributor' means any person or entity that distributes the Program.

'Licensed Patents' mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

'Program' means the Contributions distributed in accordance with this Agreement.

'Recipient' means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

1. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

2. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

3. Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

4. Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

1. it complies with the terms and conditions of this Agreement; and

2. its license agreement: i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose; ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits; iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ('Commercial Contributor') hereby agrees to defend and indemnify every other Contributor ('Indemnified Contributor') against any losses, damages and costs (collectively 'Losses') arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial

Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN 'AS IS' BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Eclipse

Copyright © 2012 The Eclipse Foundation. All Rights Reserved.

Restlet

Copyright © 2005-2012 Restlet S.A.S.

GNU GENERAL PUBLIC LICENSE VERSION 2.0

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original

licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Java Mini Web Server

Copyright Paul Mutton 2001-2010

GNU GENERAL PUBLIC LICENSE VERSION 2.0 WITH CLASSPATH EXCEPTION

As above and then:

Class Path Exception

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

GNU GENERAL PUBLIC LICENSE VERSION 3.0

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1)

displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are

packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control”

includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

[aws](#)

Copyright (c) Timothy Kay 2007

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this

License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

JCIFS

Copyright © 2010 The JCIFS Project

JNA

PostgreSQL License

PostgreSQL Database Management System
(formerly known as Postgres, then as Postgres95)

Portions Copyright (c) 1996-2012, The PostgreSQL Global Development Group

Portions Copyright (c) 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

JDOM license

Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <request_AT_jdom_DOT_org>.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <request_AT_jdom_DOT_org>.

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following:

"This product includes software developed by the JDOM Project (<http://www.jdom.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jdom.org/images/logos> .

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter_AT_jdom_DOT_org> and Brett McLaughlin <brett_AT_jdom_DOT_org>. For more information on the JDOM Project, please see <http://www.jdom.org/>.

JDOM

JSON license

Copyright (c) 2002 JSON.org

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

The Software shall be used for Good, not Evil.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF

CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

JSON in JAVa

Mozilla Public License

Mozilla Public License Version 2.0

1. Definitions

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

1. under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and
2. under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

1. for any code that a Contributor has removed from Covered Software; or
2. for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or
3. under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under

the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

1. such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and
2. You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty

Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability

Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

This Source Code Form is "Incompatible With Secondary Licenses", as defined by the Mozilla Public License, v. 2.0.

Rhino (derived from **elasticsearch**)

©1998–2012 by individual mozilla.org contributors

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Axiis

Copyright (c) 2009 Team Axiis

Birdeye

Copyright (c) 2008 United Nations Office at Geneva Center for Visual Analytics <http://cava.unog.ch>

flex-iframe

Copyright (c) 2007-2011 flex-iframe contributors

jQuery

© 2012 The jQuery Foundation

Tink Flex 4 Spark

Copyright (c) 2010 Tink Ltd - <http://www.tink.ws>

SWFObject MIT License

SWFObject

Copyright (c) 2007 Geoff Stearns, Michael Williams, and Bobby van der Sluis

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL SIMON TATHAM BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

TextRank custom license

<https://github.com/samxhuan/textrank>

Copyright (c) 2009, ShareThis, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the ShareThis, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Third Party Source Code Distribution

This portion of the document references locations of Open Source Software code repositories where required by that software's license. In any of the following cases, if the source code cannot be located at the links provided below (which are correct at the time of writing), then contact Ikanow at support@ikanow.com, and we will provide you with an archive of the source code by email or other standard distribution method.

aws command line

Home page: <http://aws.amazon.com/developertools/739>

Source code repository: <http://www.timkay.com/aws/>

Jersey

Home page: <http://jersey.java.net/>

Source code repository: <http://java.net/projects/jersey/sources>

Java Mini Web Server

Home page: <http://www.jibble.org/miniwebserver/>

Source code repository: <http://www.jibble.org/files/SimpleWebServerFull.jar>

Java Native Access

Home page: <https://github.com/twall/jna#readme>

Source code repository: <https://github.com/twall/jna>

Jcifs

Home page: <http://jcifs.samba.org/>

Source code repository: <http://jcifs.samba.org/src/src/jcifs/>

Restlet

Home page: <http://www.restlet.org/>

Source code repository: <http://www.restlet.org/downloads/2.0/>