newslynx Documentation

Release

Brian Abelson
Michael Keller

October 20, 2015

Contents

NewsLynx is a research project from the Tow Center for Digital Journalism at Columbia University by Brian Abelson & Michael Keller that has been open-sourced to use. To get a general sense of the project, the user manual is a great place to start.

For a deeper explanation of the project, our white paper (PDF, released June 2015) or the Executive Summary contain more detail. If you want more, you can read our project update posts that look at the problem of scraping the whole internet, why taxonomies are like Jorge Luis Borges or read a walkthrough of our initial prototype.

All of our code is open source and on GitHub. Please read license information before using or contact us if you have questions. Happy Lynxing!

CHAPTER 1

Contents

1.1 Configuration

NewsLynx requires specific configurations to run. By default, these should all be stored in a file in your home directory named `~/.newslynx`. In this directory there should at least be one file named `config.yaml`. This file can also be JSON. If you would like to configure another location for this file, set the environment variable `NEWSLYNX_CONFIG_FILE`. Finally, we should note that all configurations described below can also be stored as environment variables, with the naming convention: `NEWSLYNX_{SETTING_NAME}`, however you will still need to have a `config.yaml` file.

1.1.1 Required Settings

`config.yaml` contains all the information we need to run NewsLynx. As you'll see below, the App also utilizes this file for it's configurations.

In this file, we require, at, minimum, the following fields:

- **`super_user`:**
  - The name of the Super User for this NewsLynx install. / org settings.
- **`super_user_email`:**
  - The Super User's email (what you use to login with).
- **`super_user_password`:**
  - The Super User's login password. This password will enable login to all other user profiles.
- **`super_user_apikey`:**
  - The Super User's `apikey`. Principally here for development ease. Enables the regeneration of the datbase without resetting credentials.
- **`api_dns`:**
  - The URL of the final app. This is used when you authenticate with services to redirect you back to where you came from. This field isn't required if you aren't using the app at all.

1.1.2 API credentials

This file is also where you configure your credentials for Google Analytics, Twitter, Facebook, and other services. These aren't credentials for the news organization(s), rather, your instance of NewsLynx needs an "app" with each

of these services that each news organization gives its tokens **to**. For example, it's the app that you will see in your Google account under apps that have access to the news organization's account tokens. You can give that application the NewsLynx logo if you like, which will be presented to users when they authenticate with these different services from within the NewsLynx interface.

- **twitter_api_key:**
  - See Twitter's developer docs for for details on how to create an application on Twitter and configure these credentials. The option you want is to create an app. Set the callback url to `http://<app-url>/settings`. If you're running it on a port other than 80, put that in the url.
- **twitter_api_secret:**
  - See Twitter's developer docs for for details on how to create an application with Twitter and configure these credentials.
- **google_analytics_client_id:**
  - See Google's developer docs for for details on how to create an application with Google Analytics and configure these credentials. You'll want to create an application and enable the Analytics API. Then click on "Credentials" on the left and create an oAuth 2.0 for a "web application." Set the callback URI to `http://<api-url>:5000/api/v1/auths/google-analytics/callback`
- **google_analytics_client_secret:**
  - See Google's developer docs for for details on how to create an application with Google Analytics and configure these credentials. You'll want to create an application and enable the Analytics API. Then click on "Credentials" on the left and create an oAuth 2.0 for a "web application." Set the callback URI to `http://<api-url>:5000/api/v1/auths/google-analytics/callback`.
- **facebook_app_id:**
  - See Facebook's developer docs for for details on how to create an application on facebook and configure these credentials.
- **facebook_app_secret:**
  - See Facebook's developer docs for for details on how to create an application on facebook and configure these credentials.
- **reddit_user_agent:**
  - A reddit-friendly User Agent.
- **embedly_api_key:**
  - An API key for using Embedly for content extraction. If not provided, we fall back on internal methods.

1.1.3 Default Tags and Recipes

In addition to these settings, you can also configure a list of default Tags and Recipes to create for every new Organization. This is useful for creating applications on top of the API which onboard new Organizations and Users without too much hassle.

These files live, by default, in `~/.newslynx/defaults`. Each file should have the name of the items it contains and consist of a list of objects:

For example, `~/.newslynx/defaults/tags.yaml` specifies a list of tags to apply to every new organization.

```
- name: Politics
  slug: politics
  color: '#a6cee3'
  type: subject

- name: Reprint / pickup
  slug: reprint-pickup
  category: citation
  level: media
  color: '#6699cc'
  type: impact
```

If you'd like to save these files elsewhere, you can modify the following configurations

- **default_tags:**
  - A path to a `yaml` file with a list of default Tags.
  - default = `~/.newslynx/defaults/tags.yaml`
- **default_recipes:**
  - A path to a `yaml` file with a list of default Recipes.
  - default = `~/.newslynx/defaults/recipes.yaml`

By default, NewsLynx Core installs the default Tags and Recipes needed to run the Application. If you'd like to install NewsLynx core without these defaults, make sure to use the `--bare` flat when you run `newslynx init` (more details on this below).

1.1.4 Additional Options

In addition, there are numerous optional configurations you can tweak to modify the performance of NewsLynx. You can also read through them in the source code.

Super user

- **super_user_org:**
  - The Super User's Organization. The name of the organization to create on initialization. Will default to `admin`.
- **super_user_org_timezone:**
  - The timezone for the above organization. Will default to `UTC`.

Postgres

- **sqlalchemy_database_uri**
  - A valid SQLAlchemy Database URI.
  - **NOTE** This configuration is required when installing `newslynx-core` locally.
  - default = `postgresql://localhost:5432/newslynx`
- **sqlalchemy_pool_size**
  - the maximum number of concurrent database connecitons

  - default = `1000`

- **`sqlalchemy_pool_max_overflow`**
  - the maximum number of concurrent database connections over sqlalchemy_pool_size before an error is thrown.
  - default = `100`
- **`sqlalchemy_pool_timeout`**
  - the number of seconds to wait on a database transaction before throwing an error.
  - default = `60`
- **`sqlalchemy_echo`**
  - whether or not to log all sql queries. Recommended only for debugging purposes.
  - default = `false`

Redis

- **`redis_url`**
  - the URL of the redis connection
  - default = `redis://localhost:6379/0`

Caching

- **`url_cache_prefix`**
  - The key prefix of the Redis cache for URL extraction (the process of reconciling raw URLs to their canonical form)
  - default = `newslynx-url-cache`
- **`url_cache_ttl`**
  - The number of seconds before an extracted URL expires.
  - default = `1209600` (14 days)
- **`url_cache_pool_size`**
  - the number of URLs to extract conccurrently when ingesting Events
  - default = `5`
- **`extract_cache_prefix`**
  - The key prefix of the Redis cache for Article extraction (the process of extracting metadata from URLs)
  - default = `newslynx-extract-cache`
- **“extract_cache_ttl “**
  - The number of seconds before metadata extracted from a URL expires.
  - default = `259200` (3 days)
- **`thumbnail_cache_prefix`**

  - The key prefix of the Redis cache for Article extraction (the process of extracting metadata from URLs)
  - default = `newslynx-thumbnail-cache`
- **`thumbnail_cache_ttl`**
  - The number of seconds before metadata extracted from a URL expires.
  - default = `259200` (3 days)
- **`thumbnail_size`**
  - The size of thumbnails to generate. (These are stored on Events and Articles when an Image URL is present.)
  - default = `[150, 150]`
- **`thumbnail_default_format`**
  - The default format to render Thumbnails as. When we can identify the proper original format, we will render it as that format.
  - default = `png`
- **`comparison_cache_prefix`**
  - The key prefix of the Redis cache for Comparison metrics
  - default = `newslynx-comparison-cache`
- **`comparison_cache_ttl`**
  - The number of seconds before metadata extracted from a URL expires.
  - default = `86400` (1 day)
- **`comparison_percentiles`**
  - The percentiles to return in the Comparison API.
  - default = `[2.5, 5.0, 10.0, 20.0, 30.0, 40.0, 60.0, 70.0, 80.0, 90.0, 95.0, 97.5]`

Recipe Queue

- **`merlynne_kwargs_prefix`**
  - The key prefix for recipe configuraion we pass into Sous Chefs.
  - default = `newslynx-merlynne-kwargs`
- **`merlynne_kwargs_ttl`**
  - The number of seconds we'll keep these configuration in redis before they expire.
  - default = `60`
- **`merlynne_results_ttl`**
  - The number of seconds we'll keep the outputs of SousChefs in Redis before they expire.
  - default = `60`

Recipe Scheduler

- **scheduler_refresh_interval**
  - The frequency in seconds with which we'll check for updates to recipe schedules.
  - default = `45`
- **scheduler_reset_pause_range**
  - The range in seconds within which we'll reset Recipes when their schedule / configurations have changed.
  - default = `[20, 200]`

Network

- **network_user_agent**
  - The User Agent to use in the header of all outgoing network requests.
  - default = `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10; rv:33.0) Gecko/20100101 Firefox/33.0`
- **network_timeout**
  - The timout range for all network requests.
  - default = `[7, 27]`
- **network_wait**
  - How long to wait in between network retiries.
  - default = `0.8`
- **network_backoff**
  - The factor with which to multiply `network_wait` on each subsequent retry.
  - default = `2`
- **network_max_retries**
  - The maximum number of retries before failing.
  - default = `2`

Notifications

- **notify_methods**
  - A list of notification methods to utilize. Currently `email` and `slack`.
  - default = `[]`
- **notify_email_recipients**
  - A list of emaill addresses to send notifications to.
  - default = `[]`
- **notify_email_subject_prefix**
  - The prefix to insert into the subject of all email notifications.

  - default = `[ Merlynne ]`

- **notify_slack_webhook**
  - A slack webhook url for posting notifications to.
  - see: https://api.slack.com/incoming-webhooks
- **notify_slack_channel**
  - The slack channel to post to.
  - default = `#general`
- **notify_slack_username**
  - The slack username to post as.
  - default = `Merlynne`
- **notify_slack_emoji**
  - The slack emoji to post with.
  - default = `:-1:`

Email

These configurations are only currenly required for installs where `notify_methods` includes `email`. In the future, there will be more email integrations and these configurations will stay the same.

- **mail_username**
  - The username of the account to use for sending and recieving emails.
- **mail_password**
  - The password of the account to use for sending and recieving emails.
- **mail_server**
  - The domain of the account's server (e.g. mail.google.come)
- **mail_smtp_port**
  - The server's smtp port for sending messages
- **mail_imap_port**
  - The server's imap port for receiving messages

1.1.5 Intialization

Once you have setup your configurations, follow the installation docs.

1.2 Development Installation Guide

The following installation guide is oriented towards people interested in running NewsLynx locally on a Mac OS X computer. For deploying the software, please refer to our automation docs.

1.2.1 NewsLynx Core

NOTE - For most applications, we recommend following our automation guide. If you'd like to setup a development environment, following the instructions below for MacOS X. At this time, NewsLynx Core is not supported on Windows.

Install the dependencies

`newslynx` depends on Postgres and Redis. If you're on Mac OS X, the easiest way to run Postgres is with the Postgres.app. However, if you prefer the Homebrew distribution, make sure to install it with plpython.

```
$ brew install postgresql --build-from-source --with-python
```

If you already have the Hombrew version installed, run this command:

```
$ brew reinstall postgresql --build-from-source --with-python
```

Finally, install `redis` via Homebrew

```
$ brew install redis
```

If `redis` does not automatically start, open another tab and run

```
$ redis-server
```

Installation

OPTIONAL - First set your configurations. If you don't do this, we will fallback on the app's default configuration file.

Install `newslynx`, initialize the database, super user, and install default sous chefs, tags, and recipes.

```
$ git clone https://github.com/newslynx/newslynx-core.git
$ cd newslynx-core
$ make app_install
```

EXPERT MODE - Don't install the app's default sous chefs, tags, or recipes.

```
$ git clone https://github.com/newslynx/newslynx-core.git
$ cd newslynx-core
$ make bare_install
```

For the next steps, refer to the getting started docs.

1.2.2 NewsLynx App

NOTE - For most applications, we recommend following our automation guide. If you'd like to setup a development environment, following the instructions below for MacOS X.

Download the git repository and install dependencies:

```
$ git clone https://github.com/newslynx/newslynx-app && cd newslynx-app
$ npm install
```

1.2.3 NewsLynx Desktop

We also have a beta desktop version of NewsLynx App, which brings all of the same functionality of the web interface to a native environment, which can be easier for people to use. You can download the latest release for Mac OS X on the Releases page of the project repo.

Follow that project's issue tracker for progress on the first 1.0 release. If you would like to try it in the mean time please do. If you would like a Windows or Linux version, let us know by filing an issue and we can bake one out.

1.3 Running the API and Application

1.3.1 Running Newslynx Core

Once you've installed and initialized NewsLynx Core, you can start a debug server with the following command:

```
$ newslynx debug
```

If you'd like to start a multi-theaded production server (some Sous Chefs may not work without this), run this command inside the root directory of `newslynx-core`:

```
$ bin/run
```

To start the task queue, run this command inside the root directory of `newslynx-core`:

```
$ bin/start_workers
```

To stop the task queue, run this command inside the root directory of `newslynx-core`:

```
$ bin/stop_workers
```

To start the Recipe scheduler, run this command:

```
$ newslynx cron
```

1.3.2 Running NewsLynx App

To start the server, in the `newslynx-app` folder, run the following:

```
$ npm start
```

This compiles your CSS and JS and runs the server with Forever.

When you see the following, it's done and you can visit http://localhost:3000.

Note: If you are running this in production, you want to run it in behind https and tell the app you are doing so one of two ways:

1. Run it with the environment variable `NEWSLYNX_ENV=https`
2. Set `https: true` in your `~/.newslynx/config.yaml` file

This will make sure your cookies are set securely.

```
#####################################
# HTTP listening on 0.0.0.0:3000... #
#####################################
```

Other App start up commands

Alternate commands are in package.json under "scripts". These are for **developing locally.**

If you want to modify files and have the CSS and JS re-compiled automatically and the server restarted if necessary, do:

```
$ npm run dev
```

If you just want to watch the CSS and JS and re-compile when on change, do:

```
$ npm run watch-files
```

If you just want to watch the Express server and restart when its files change (templates, server js files), do:

```
$ npm run watch-server
```

These last two commands are best run in tandem in two separate shell windows. *npm run dev* does them both in one window for convenience.

The final command listed is `npm test`, which will run a simple test to make sure the server can launch.

1.4 Sous Chefs

NewsLynx is powered by Sous Chefs. Sous Chefs are small `python` scripts that can do almost anything to modify NewsLynx's data via the API. We implemented NewsLynx this way for the following reasons:

1. Different Organizations value different Metrics and different Events. Locking all Organizations into a single framework ultimately inhibits them. By making NewsLynx fully modular, we can account for many different potential use cases.
2. Social networks and analytic providers come in and out of fashion. The landscape of online media is constantly in flux. While we might assume that most every media organization has a Google Analytics account and a Facebook page, this may not be the case next year. By not assuming the primacy of any particular data source, Sous Chefs allows NewsLynx to grow and adapt to shfiting conditions.
3. Certain Organizations may have internal CMS's or Analytics tools which they have created internally. Sous Chefs enables these organizations to import these sources into NewsLynx.

1.4.1 Writing Sous Chefs

All Sous Chefs must inherit from the core `newslynx.sc.SousChef` class. You can see the source code for this class here. Every Sous Chef can define the following methods:

- **`setup`**
  - Configures the Sous Chef before executing it. This method can be used for authenticating wtih APIs or setting custom properties of the Sous Chef.
- **`run`**
  - Executes the Sous Chef. If this method is designed to bring data into NewsLynx, it should return a list or generator of dictionaries.
- **`load`**
  - Loads data output from `run` into NewsLynx. If this SousChef yields more than one record, it should utilize the *Bulk API*

- **teardown**
  - Do something after the Sous Chef finishes loading data.

You can see examples of our internal Sous Chefs here.

1.4.2 Configuration

All Sous Chefs are configured via a `yaml` or `json` file. These files follow a JSON schema which is defined here. Every Sous Chef consists of the following configurations:

- **name**
  - The display name of the Sous Chef.
  - **Required**
- **slug**
  - A unique slug for the Sous Chef. This can double as it's ID in API calls. Must only contain letters and dashes. In `regex`, this means: `'^[a-z][a-z\-]+[a-z]$'`.
  - **Required**
- **description**
  - A description of what this Sous Chef does.
  - **Required**
- **runs**
  - The `python` import path for the Sous Chef. For instance, if you built a `python` module named `my_sous_chef`, this might take the value of `my_sous_chef.SousChef`. The class at the end of the import path must inherit from the core `newslynx.sc.SousChef` class. You can see the source code for this class here. You can see an example of one of these modules here or below.
  - **Required**
- **creates**
  - What type of collection does this Sous Chef create?
  - **Required**
  - **Can be one of:**
    * `events`
    * `content`
    * `tags`
    * `metrics`
    * `report`
    * `external`
    * `internal`
- **option_order**
  - The order in which `options` are rendered. This parameter should take a list of option names. This parameter primarily exists to aid in the dynamic rendering of Sous Chef input forms.
  - *Optional*

- **includes**
  - A list of relative paths to other Sous Chef config files to inherit for this Sous Chef. This prevents rewriting option definitions for Sous Chefs with similar behaviors. If this file contains a partial Sous Chef config, it's filename should begin with an underscore. You can see an example of such a file here and it's include statement here.
  - *Optional*
- **requires_auths**
  - A list of <endpoints-auths> `names` that must exist for an Organization before this Sous Chef can run.
  - *Optional*
- **requires_settings**
  - A list of <endpoints-settings> `names` that must exist for an Organization before this Sous Chef can run.
  - *Optional*
- **options**
  - The options this Sous Chef takes.
  - *Optional* (if this Sous Chef's behavior should not be modified).
- **metrics**
  - The Metrics this Sous Chef creates.
  - *Optional* (unless the Sous Chef creates *Metrics*).

1.4.3 Options

Sous Chefs can specify as many options as they require. These options can be passed into the Sous Chef to change it's behavior. At their core, *Recipes* are simply a populated list of these options. Each option follows the schema sepecified below:

- **input_type**
  - What type of input form should this option render?
  - This options enables `newslynx-app` to dynamically render forms to populate Sous Chef options.
  - **This option can accept the following values**
    - **search**
      - Render a search form to aid in populating the option's value. In practice, this is used to add items to an option which can be searched for via the API.
    - **radio**
      - Render a radio form.
    - **select**
      - Render a drop-down form.
    - **checkbox**
      - Render a checkbox form.
    - **checkbox-single**

        - Render a single-checkbox. This would be used, for example, in the case that you want an non-required option to take a `default` value of `true` while enabling a user to override this default with a value of `false`.

    * **number**
        - Render a numeric form.

    * **datepicker**
        - Render a form which enables a user to select a date.

    * **text**
        - Render an open-ended text form.

    * **paragraph**
        - Render an open-ended text form with more room to fill in details. Primarily for use with `description` fields.

- **input_options**
    - If the `input_type` is `radio`, `select`, `checkbox`, or `checkbox-single`, a list of possible options to populate the form.
    - This parameter enables `newslynx-app` to dynamically render dropdowns, checkbox, or radio options.
- **value_types**
    - What value types does this option accept?
    - This parameter enables `newslynx-core` to exhaustively validate options before executing Sous Chefs.
    - **This option can accept the following values:**

        * **datetime**
            - An ISO-8601 date.

        * **crontab**
            - A cron string.

        * **json**
            - A complex option, usually a dictionary, which only needs to be json-serializable.

        * **regex**
            - Valid input to `re.compile()`.

        * **boolean**
            - Truish (true, t, yes, y, 1, on) or Falsish Values (false, f, no, n, off).

        * **numeric**
            - Valid input to `float()` or `int()`.

        * **string**
            - Valid input ot `str()`

        * **nulltype**
            - Any of (None, null, N/A, NaN)

        * **url**
            - A valid URL (determined by regular expresion.).
        * **email**
            - A valid email address (determined by regular expresion.)
        * **searchstring**
            - A custom type which we use to provide basic search capabilities to Sous Chefs (explained below.)

- **accepts_list**
    - Does this option accept a list of values?
    - Defaults to `false`.
- **default**
    - What is the default value for this options?
    - Defaults to `null`.
- **required**
    - Is this option required for the Sous Chef to properly run?
    - Defaults to `false`.
- **help**
    - Parameters to help users properly fill out options.
    - `help` is a dictionary of the following values:
    - **placeholder**
        * The placeholder/example text for this option.
    - **link**
        * A link for more details about this option.
    - **description**
        * A description of this option to display on form hover.

1.4.4 Default Options

All Sous Chefs come with the following default options. These exist for aiding in the creation and scheduling of *Recipes*. For more information on how these options affect when a Sous Chef is executed, refer to the *Scheduling* docs.

```
name:
    input_type: text
    value_types:
        - string
    required: true
    help:
        description: The name of the Recipe.

slug:
    input_type: text
```

```
    value_types:
        - string
    help:
        placeholder: (optional)
        description: The recipe slug. Lowercase and separated with '-'.

description:
    input_type: paragraph
    value_types:
        - string
        - nulltype
    help:
        placeholder: (optional)
        description: A description of what this recipe does.

schedule_by:
    input_type: select
    input_options:
        - minutes
        - time_of_day
        - crontab
        - unscheduled
    value_types:
        - string
        - nulltype
    default: null
    help:
        placeholder: minutes
        description: The method for scheduling the recipe.

crontab:
    input_type: text
    value_types:
        - crontab
        - nulltype
    default: null
    help:
        placeholder: "*/30 * * * *"
        description: A crontab string to use for scheduling this recipe.
        link: "https://en.wikipedia.org/wiki/Cron"

minutes:
    input_type: number
    value_types:
        - numeric
        - nulltype
    default: null
    help:
        placeholder: 60
        description: The frequency with which this Recipe should run (in minutes).

time_of_day:
    input_type: select
    input_options:
        - '12:00 AM'
        - '12:30 AM'
        ...
        - '11:30 PM'
```

```
    value_types:
        - string
        - nulltype
    default: null
    help:
        placeholder: '4:30 PM'
        description: The time of day at which this Recipe should run daily.

timeout:
    input_type: number
    value_types:
        - numeric
    default: 240
    help:
        placeholder: 240
        description: |
            The number of seconds after which this Recipe will time out.
```

1.4.5 Search

As mentioned above, Sous Chefs can accept options with a value type of `searchstring`. A search string is a built-in type that has the following capabilities (described through examples):

- **`term`**
  - match on a term
- **`~term`**
  - fuzzy match on a term using jaro-winkler distance
- **`/.*term.*/`**
  - apply a regex
- **`"term1 term2"`**
  - match on a phrase
- **`~"term1 term2"`**
  - fuzzy match on a phrase
- **You can chain two search strings together with the following operators:**
  - `AND` => must match both searchstrings
  - `&` => must match both searchstrings
  - `OR` => can match either term
  - `|` => can match either term
- There is not yet support of parenthetical grouping of chained terms.

1.4.6 Configuring Metrics

Sous Chefs which create metrics must also specify the schema of the metrics they create. This schema is specified in the *Metrics* docs.

1.4.7 Examples

The best way to understand how Sous Chef's work is to look at the Source Code for the built-in modules here. You can see an example of a custom Sous Chef module here. If you're interested in writing your own Sous Chefs, please refer to the writing-sous-chefs docs.

1.5 Metrics

In our white paper (PDF, released June 2015), which had originally been focused on impact measurement, we were surprised to learn how important quantitative metrics were. More importantly, we came to understand how newsrooms were getting their numbers from a nearly infinite range of sources. Many newsrooms like NPR have even developed metrics that best reflect their unique understanding of their audience. A principal challenge in building NewsLynx was determining how best to collect, store, analayze, and present metrics in a way that allowed Newsrooms to focus on the metrics that mattered most to them.

The system we devised adheres to the following standards:

1. While NewsLynx is an analytics platform, it is not a service for tracking events on sites. NewsLynx exists in between analytics providers and their audiences as a tool which enables the curation and customization of metrics.
2. For the reasons above, **we do not store timeseries data more granular than hour**. While this is configurable by setting `metrics_min_date_unit` and `metrics_min_date_value` in your `config.yaml`, it's not encouraged.
3. Metrics should be stored in the their rawest state but should be automatically summarized and compared. The process from ingestion => summary => comparison should be both invisible to the user and effortlessly undone.
4. While we should make no assumptions about the data source a user prefers, we should not have to constantly compromise the above standards to accomodate one source. With that in mind, we should have a schema for Metrics which, while rigid, is flexible enough to handle most use cases. In the cases which this schema does not prove adequate, we should consider whether this particular use case is general enough before modifying the schema.

1.6 Creating Metrics

As described in the Sous Chef docs, Metrics are exclusively defined by the Sous Chefs which create them. All Sous Chefs that create metrics must have the following section in their configuration:

```
metrics:
    metric_name:
        **options
    anoter_metric_name:
        **options
    ...
```

1.6.1 Options

All metrics have access to the following options:

- **`display_name`**
    - How should this metric be displated?

- **description**
  - An informative explanation of what this metric represents.
- **type**
  - Helps in determining how metric should be interpreted / summarized / presented.
  - **Can be one of:**
    - **count**
      - A number of something,
      - e.g. pageviews, time on page, attention minutes, etc.
      - `count` metrics are also fill-ins for Binary or Boolean fields. Simply store these values as `0` or `1`. If you'd like to summarize them as Booleans, set the `agg` to `max`.
      - Metrics with this type will be assumed to have an `agg` of `sum` unless overridden.
    - **cumulative**
      - A count that increases over time.
      - This is a special type of Metric in that we should like to capture its differences over time periods. However, we also want to avoid alteration of the original source of the data. As a result, a `cumulative` metric is stored as cumulative sum, but when queried, is transformed into a `count`.
      - e.g. twitter shares, facebook comments, followers, etc.
      - Metrics with this type will be assumed to have an `agg` of `sum` unless overridden.
    - **median**
      - The median of a list of metrics.
      - e.g. median time on page.
      - Metrics with this type will be assumed to have an `agg` of `median` unless overridden.
    - **average**
      - The average of a list of metrics.
      - e.g. average time on page.
      - Metrics with this type will be assumed to have an `agg` of `average` unless overridden.
    - **percentile**
      - Usually a number beween 0 - 100.
      - e.g. percent internal traffic
      - Metrics with this type will be assumed to have an `agg` of `avg` unless overridden.
    - **min_rank**
      - A number which should be interpreted as "a lower number is good."
      - e.g. position on homepage.
      - Metrics with this type will be assumed to have an `agg` of `min` unless overridden.
    - **max_rank**
      - A number which should be interpreted as "a higher number is good."

            - e.g. position on homepage.
            - Metrics with this type will be assumed to have an `agg` of `max` unless overridden.

- **agg**
    - The function to use when aggregating this metric.
    - In practice, these map directly onto `postgres` functions.
    - **Can be one of (for now):**
        * `sum`
        * `avg`
        * `median`
        * `max`
        * `min`
- **content_levels**
    - This field lets us know that the metric is related to content items and should be stored at the specified level. For more on what this means see metrics-how-does-this-work
    - **Can be one of:**
        * **timeseries**
            - Accessible via the Content Timeseries API.
        * **summary**
            - Accessible via the Content Search API endpoints-content-items-search and when retrieving *individual Content Items*.
        * **comparison**
            - Accessible via endpoints-content-metrics-get-comparisons
- **org_levels**
    - This field lets us know that the metric is linked to an organization as a whole. For more on what this means see metrics-how-does-this-work
    - **Can be one of:**
        * **timeseries**
            - Accessible via endpoints-org-metrics-get-timeseries
        * **summary**
            - Accessible via endpoints-org-metrics-get-summary

For instance, the Sous Chef `google-analytics-to-content-timeseries` lists this metric configuration:

```
metrics:
    ga_pageviews:
        display_name: Pageviews
        description: |
            The number of times this page was opened,
            as reported by Google Analytics.
        type: count
        content_levels:
            - timeseries
```

```
            - summary
            - comparison
        org_levels:
            - timeseries
            - summary

    ga_total_time_on_page:
        display_name: Total Time on Page
        description: |
            The total time visitors spent on this page,
            as reported by Google Analytics.
        type: count
        content_levels:
            - timeseries
            - summary
            - comparison
        org_levels:
            - timeseries
            - summary

    ga_avg_time_on_page:
        display_name: Average Time on Page
        type: computed
        content_levels:
            - timeseries
            - summary
            - comparison
        org_levels:
            - timeseries
            - summary
        formula: '{ga_total_time_on_page} / NULLIF({ga_pageviews}, 0)'
        agg: avg
```

1.7 Recipes

Documentation in progress...

1.8 Scheduling

1.9 Taxonomy

Documentation in progress...

1.9.1 Tags

Tag Type

Subject Tag

Impact Tag

Impact Tag Category

Impact Tag Level

1.10 Content Items

Documentation in progress...

1.11 Events

Documentation in progress...

1.12 NewsLynx API

All of **NewsLynx**'s functionality is exposed through a RESTful API. This API is mirrored through a built-in command line interface – `newslynx` (or if you prefer to be 1337, `nlx`) – and `python` client – `newslynx.client.API` – as well. The following guide will walk you through all of the the API's functionalities with accompanying examples for `python`, `newslynx`, and `CuRL`.

1.12.1 Authentication

API Keys

All endpoints except for: *POST /login*, require an `apikey` passed in as a query-string parameter. The `apikey` is used to ensure that the requesting user is registered and has access to the the requested resource. When a user is created, his/her `apikey` is randomly generated. It can be refreshed at any time.

Organizations

All endpoints except for those associated with endpoints-users and *Organizations* require an `org` query-string parameter. Since users can belong to multiple organizations, we use this parameter to ensure:

1. That the requesting user is accessing resources related to his/her associated organization.
2. That the requesting user has permission to access the requested resources.

1.12.2 Status Codes

Successes

Successful responses from the API will only return the following status codes:

HTTP status	Description
200 Success	Standard response for successful HTTP requests. The actual response will depend on the request method used.
202 Queued	The server successfully processed a request, and is returning a pointer to a url to check on it's statues
204 No Content	The server successfully processed the request, but is not returning any content. This response is only used for certain `DELETE` methods.

Errors

All errors share this common `json` schema:

```
{
  "status_code": 401,
  "error": "AuthError",
  "message": "Invalid API Key."
}
```

Various status codes connote different categories of errors:

HTTP status	Description
400 Bad Request	The server rejected the request as invalid.
401 Unauthorized	The authentication associated with the request is invalid
403 Forbidden	Authentication is valid, but user is not allowed to perform this operation
404 Not Found	A required object for this operation was not found
405 Invalid	This method does not exist
409 Conflict	A unique constraint was violated
422 Unprocessable Entity	The request was formatted correctly but could not be processed
500 Internal Server Error	The server encountered an unexpected error while processing the request

However, every error will contain a specific message. In the case of Internal Server Errors, the server will respond with the error message generated by the responsible code. You should probably report these to our

1.12.3 Namespacing

All endpoints are prefaced by `/api/:version`. The current (and only version) is `v1`.

1.12.4 Endpoints

All endpoints, unless ortherwise noted, return `json`. If you'd like to follow along with these examples, go ahead and set your `apikey` as an environment variable:

```
export NEWSLYNX_APIKEY='djljahflsdkfhasldkfhasldfa'
export NEWSLYNX_ORG=1 # The org id/slug you wish to access.
                      # Setting this will prevent you from having
                      # to repetitively pass this argument to the
                      # API client / command line interface.
```

User

The **User** API enables login and management of basic user settings (password, name, email, apikey, etc).

User JSON

All methods (unless otherwise specified) return the following `json` object:

```
{
  "organizations": [
    {
      "id": 1,
      "name": "Xosy Media",
      "timezone": "UTC"
    }
  ],
  "apikey": "djljahflsdkfhasldkfhasldfa",
  "name": "Merlynne Jones",
  "created": "2015-05-03T16:21:41.995821-04:00",
  "super_user": true,
  "admin": true,
  "id": 1,
  "email": "merlynne@newslynx.org"
}
```

POST `/login`

Enables authentication via `email` and `password`.

Body

```
{
  "email": "merylnne@newlsynx.org",
  "password": "a-secure-p4ssw0rd"
}
```

Returns A *User JSON* object.

Example Via `CuRL`

```
$ curl --data "email=merlynne@newslynx.org&password=a-secure-password" \
http://localhost:5000/api/v1/login
```

Via `newslynx`

```
$ newslynx api me login email=merlynne@newslynx.org password=a-secure-password
```

Via `python`

```
from newslynx.client import API

api = API()
api.me.login(email=merlynne@newslynx.org, password=a-secure-password)
```

GET /me

Retrieves your user profile.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true

Returns A *User JSON* object.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/me\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api me get
```

Via `python`

```
from newslynx.client import API

api = API()
api.me.get()
```

PUT | PATCH /me

Update your `name`, `email`, `apikey` and/or `password`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`refresh_apikey`	`true` / `false`. If `true`, your `apikey` will be refreshed.	false	false

Body

```
{
  "email": "merylnne2@newlsynx.org",
  "old_password": "a-secure-p4ssw0rd",
  "new_password": "a-more-secure-p4ssw0rd",
  "name": "Meryl Jones"
}
```

Returns An updated *User JSON* object.

Examples Change your `name` and `email`.

Via `CuRL`

```
$ curl -X PUT -d email=merlynne2@newslynx.org -d name="Meryl Jones" \
http://localhost:5000/api/v1/me\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api me update email=merlynne2@newslynx.org name="Meryl Jones"
```

Via `python`

```
from newslynx.client import API

api = API()
api.me.update(email="merlynne2@newslynx.org", name="Meryl Jones")
```

Change your `password`.

Via `CuRL`

```
$ curl -X PUT -d old_password="a-secure-p4ssw0rd" -d new_password="a-more-secure-p4ssw0rd" \
http://localhost:5000/api/v1/me\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api me update old_password="a-secure-p4ssw0rd" new_password="a-more-secure-p4ssw0rd"
```

Via `python`

```
from newslynx.client import API

api = API()
api.me.update(
  old_password="a-secure-p4ssw0rd",
  new_password="a-more-secure-p4ssw0rd"
)
```

Refresh your `apikey`

Via `CuRL`

```
$ curl -X PUT http://localhost:5000/api/v1/me\?apikey=$NEWSLYNX_APIKEY\&refresh_apikey=true
```

Via `newslynx`

```
$ newslynx api me update --refresh_apikey
```

Via `python`

```
from newslynx.client import API

api = API()
api.me.update(refresh_apikey=True)
```

DELETE /me

Delete your account.

NOTE - Will re-assign your Recipes to the Super User.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true

Returns Status: `204`

Example Via `CuRL`

```
$ curl -X DELETE http://localhost:5000/api/v1/me\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api me delete
```

Via `python`

```
from newslynx.client import API

api = API()
api.me.delete()
```

Organizations

The **Organizations** API enables the creation / updating / deleting of organizations, users and settings, as well as assigning / removing users from organizations. All **PUT**, **PATCH**, **POST**, and **DELETE** methods require that the requesting user is an admin or super user.

Organization JSON

All methods, unless otherwise specified, will return one or many organization objects of the following `json` schema:

```
{
  "id": 1,
  "name": "liveqa",
  "timezone": "America/New_York"
  "users": [...],
  "settings": {...},
  "auths": {...}
}
```

GET `/orgs`

Fetch a list of organizations you have access to.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true

Returns A list of *Organization JSON* objects.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/orgs\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api orgs list
```

Via `python`

```
from newslynx.client import API

api = API()
api.orgs.list()
```

POST `/orgs`

Create an organization. This will also add the requesting user to that organization.

NOTE

- Requires admin/super user privileges.
- Will simultaneously create all built-in SousChefs, Metrics, default Recipes and Tags for this organization. Check out the *Configuration* docs for more details.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body

```
{
  "name": "ProPalpatine",
  "slug": "pro-palpatine",
  "timezone": "UTC"
}
```

Returns An *Organization JSON* object.

Example Via `CuRL`

```
$ curl --data "name=ProPalpatine&timezone=UTC&slug=pro-palpatine" \
http://localhost:5000/api/v1/orgs\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api orgs create name=ProPalpatine timezone=UTC slug=pro-palpatine
```

Via `python`

```
from newslynx.client import API

api = API()
api.orgs.create(name="ProPalpatine", timezone="UTC", slug="pro-palpatine")
```

GET `/orgs/:org_id`

Fetch an organization object.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Organization JSON* object.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/orgs/1\?apikey=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api orgs get id=1
```

Via python

```
from newslynx.client import API

api = API()
api.orgs.get(1)
```

PUT | PATCH /orgs/:org_id

Change an organization's name, slug and/or timezone.

NOTE

- Requires admin privileges.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Organization JSON* object.

Body

```
{
  "name": "ProPalpatine2",
  "slug": "pro-palpatine-2",
  "timezone": "US/Western"
}
```

Example Via CuRL

```
$ curl -X PUT -d name=ProPalpatine2 -d name=pro-palpatine-2 -d timezone=US/Western \
http://localhost:5000/api/v1/orgs/1\?apikey=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api orgs update id=1 name=ProPalpatine2 timezone=US/Western slug=pro-palpatine-2
```

Via python

```
from newslynx.client import API

api = API()
api.orgs.update(1,
    name="ProPalpatine2",
    timezone="US/Western",
    slug="pro-palpatine-2")
```

DELETE /orgs/:org_id

Delete an organization and all of it's associated collections.

NOTE

- Requires admin privileges.

WARNING:

- This method will delete all data associated with this organization, except for users.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns STATUS_CODE: 204

Example

```
$ curl -X DELETE http://localhost:5000/api/v1/orgs/1\?apikey=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api orgs delete id=1
```

Via python

```
from newslynx.client import API

api = API()
api.orgs.delete(1)
```

GET /orgs/:org_id/users

Fetch all users associated with an organization.

NOTE

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A list of *User JSON* object.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/orgs/1/users\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api orgs list-users id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.orgs.list_users(1)
```

POST `/orgs/:org_id/users`

Create a new user under an organization.

NOTE

- Requires admin privileges.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body

```
{
  "name": "Brian Abelson",
  "email": "b@nytimes.cat",
  "password": "l0l4k4t",
  "admin": false
}
```

Returns A *User JSON* object.

Example via CuRL

```
$ curl --data "name=Brian Abelson&email=b@nytimes.cat&password=l0l4k4t&admin=false" \
http://localhost:5000/api/v1/orgs/1/users\?apikey=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api orgs create-user id=1 \
  name='Brian Abelson' email='b@nytimes.cat'\
  password='l0l4k4t' admin=false
```

Via python

```
from newslynx.client import API

api = API()
api.orgs.create_user(1,
    name='Brian Abelson', email=b@nytimes.cat,
    password=l0l4k4t, admin=false)
```

GET /orgs/:org_id/users/:user_id

Fetch a user that belongs to a given organization.

NOTE

- You can pass in either an user's id or his/her email to this endpoint.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A *User JSON* object.

Example via CuRL

```
$ curl http://localhost:5000/api/v1/orgs/1/users/b@nytimes.cat\?apikey=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api orgs get-user id=1 user_id=b@nytimes.cat
```

Via python

```
from newslynx.client import API

api = API()
api.orgs.get_user(1, 'b@nytimes.cat')
```

PUT | PATCH /orgs/:org_id/users/:user_id

Add an existing user to an organization.

NOTE:

- Requires admin privileges.
- You can pass in either an user's `id` or his/her `email` to this endpoint.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A *User JSON* object with a new organization affiliation.

Example Via `CuRL`

```
$ curl -X PUT http://localhost:5000/api/v1/orgs/1/users/m@nytimes.cat\?apikey=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api orgs add-user id=1 user_id=b@nytimes.cat
```

Via `python`

```
from newslynx.client import API

api = API()
api.orgs.add_user(1, 'b@nytimes.cat')
```

DELETE /orgs/:org_id/users/:user_id

Remove a user from an organization.

NOTE:

- Requires admin privileges.
- You can pass in either an user's `id` or his/her `email` to this endpoint.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`force`	`true` / `false`. If `true`, the user will be permanently deleted. This option is only relevant if a user does not have other organizational affiliations.	false	false

Returns Status: `204`

Example Via CuRL

```
$ curl -X DELETE http://localhost:5000/api/v1/orgs/1/users/m@nytimes.cat\?apikey=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api orgs remove-user id=1 user_id=b@nytimes.cat
```

Via python

```
from newslynx.client import API

api = API()
api.orgs.remove_user(1, 'b@nytimes.cat')
```

Settings

The **Settings** API enables the creation / updating / deleting of arbitrary settings associated with an organization or user. The settings collection is key/value store which can grow with the complexity of the application.

** NOTE ** : Settings can be nested under the *Organizations* and endpoints-users collections. To route a setting to it's proper collection, just place its `level` in the url. These are the current collections which can have settings:

- `/orgs/settings` Handles organization-level settings.
- `/me/settings` Handles settings for the authenticated user.

Setting JSON

All methods, unless otherwise specified, will return one or many setting objects of the following `json` schema:

```
{
    "id": 1,
    "name": "logo_image",
    "level": "orgs",
    "value": "http://example.com/mylogo.png",
    "json_value": false
}
```

If a setting has been declared as having a `json_value`, it will be parsed as such in the response:

```
{
    "id": 1,
    "name": "short_domains",
    "level": "orgs",
    "value": ["prplp.tn", "3mpi.re"],
    "json_value": true
}
```

GET `/:level/settings`

Get a list of an user / organziation settings.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A list of *Setting JSON* objects.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/orgs/settings\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api settings list
```

Via `python`

```
from newslynx.client import API

api = API()
api.settings.list()
```

POST `/:level/settings`

Add a setting to an organization.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body A *Setting JSON* object without an `id`.

Returns A newly-created *Setting JSON* object with an `id`.

Examples Create a simple setting.

Via `CuRL`

```
$ curl --data "name=icon&value=http://example.com/mylogo.png" \
http://localhost:5000/api/v1/orgs/settings\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api settings create name=icon value=http://example.com/mylogo.png
```

Via python

```
from newslynx.client import API

api = API()
api.settings.create(name="icon" value="http://example.com/mylogo.png")
```

Create a json setting.

Via CuRL

```
$ curl --data "name=short_urls&value=[\"prplt.in\"]&json_value=true" \
http://localhost:5000/api/v1/orgs/settings\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api settings create name=short_urls value='["prplt.in"]' json_value=true
```

Via python

```
from newslynx.client import API

api = API()
api.settings.create(name="short_urls", value=["prplt.in"], json_value=True)
```

GET /:level/settings/:setting_id

Get an organization's setting by it's name.

NOTE:

- You can pass in either a setting's id or it's name to this endpoint.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A *Setting JSON* object.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/orgs/settings/short_urls\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api settings get id=short_urls
```

Via python

```
from newslynx.client import API

api = API()
api.settings.get("short_urls")
```

PUT | PATCH /:level/settings/:setting_id

Update a setting for an organization.

NOTE:

- You can pass in either a setting's id or it's name to this endpoint.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body A partial or complete *Setting JSON* object.

Returns A modified *Setting JSON* object.

Examples Update a setting.

Via CuRL

```
$ curl -X PUT -d "value=[\"zzzz.in\"]" -d "json_value=true" \
http://localhost:5000/api/v1/orgs/settings/short_urls\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api settings update id=short_urls value='["zzzz.in"]' json_value=true
```

Via python

```
from newslynx.client import API

api = API()
api.settings.get("short_urls")
```

DELETE /:level/settings/:setting_id

Delete an organization's setting by it's name.

NOTE:

- You can pass in either a setting's `id` or it's `name` to this endpoint.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns Status: `204`

Example

```
$ curl -X DELETE http://localhost:5000/api/v1/orgs/settings/short_urls\?apikey=$NEWSLYNX_APIKEY\&org=
```

Via `newslynx`

```
$ newslynx api settings delete id=short_urls
```

Via `python`

```
from newslynx.client import API

api = API()
api.settings.delete("short_urls")
```

Authorizations

The **Authorizations** API enables authorization with external platforms. Currently it supports

- Google Analytics - Track site traffic
- Twitter - Access tweets from individual users or lists.
- Facebook - Access a organization's Facebook page and, depending on the configuration of the associated Facebook application, collect Insights data.

Authorization JSON

All methods (unless otherwise specified) return the following `json` object:

```
{
  id: 2,
  value: {
    "oauth_token_secret": "ldsfkasdlfjasdlfa380257234",
    "oauth_token": "2419585021-fdfdskadfjakjsdafjd"
  },
  name: "twitter"
}
```

GET /auths

Fetches a list of an organization's authorizations.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A list of *Authorization JSON* objects.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/auths\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api auths list
```

Via python

```
from newslynx.client import API

api = API()
api.settings.delete("short_urls")
```

GET /auths/google-analytics

Fetches an organization's Google Analytics authorization.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Authorization JSON* object

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/auths/google-analytics\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api auths get service=google-analytics
```

Via `python`

```
from newslynx.client import API

api = API()
api.auths.get("google-analytics")
```

GET /auths/google-analytics/grant

Authorizes NewsLynx to access an organization's Google Analytics.

NOTE

This method will prompt a user to authenticate with Google Analytics. Upon successful authentication it will direct them to a form which they can use to select which properties and profiles they would like to grant NewsLynx access to. If at any point a user would like to change these properties, he/she simply needs to access this method again - it's not necessary to revoke access first.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`redirect_uri`	The url which you would like to send a user back to after an authentication attempt	null	false

Returns An *Authorization JSON* object or, if a `redirect_uri` is provided, a redirection to that location with the added query string parameter `auth_success` to indicate whether or not the authorization request was successful.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/auths/google-analytics/grant\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api auths grant service=google-analytics
```

NOTE Will open up a browser to the specified url and run you through the authorization process.

Via `python`

```
from newslynx.client import API

api = API()
api.auths.grant("google-analytics")
```

NOTE Will open up a browser to the specified url and run you through the authorization process.

GET /auths/google-analytics/revoke

Revokes an organization's Google Analytics authorization.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true

Returns Status: 204

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/auths/google-analytics/revoke\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api auths revoke service=google-analytics
```

Via python

```
from newslynx.client import API

api = API()
api.auths.revoke("google-analytics")
```

GET /auths/twitter

Fetches an organization's Twitter authorization.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Authorization JSON* object

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/auths/twitter\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api auths get service=twitter
```

Via python

```
from newslynx.client import API

api = API()
api.auths.get("twitter")
```

GET /auths/twitter/grant

Authorizes NewsLynx to access an organization's Twitter profile.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
redirect_uri	The url which you would like to send a user back to after an authentication attempt	null	false

Returns An *Authorization JSON* object or, if a redirect_uri is provided, a redirection to that location with the added query string parameter auth_success to indicate whether or not the authorization request was successful.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/auths/twitter/grant\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api auths grant service=twitter
```

NOTE Will open up a browser to the specified url and run you through the authorization process.

Via python

```
from newslynx.client import API

api = API()
api.auths.grant("twitter")
```

NOTE Will open up a browser to the specified url and run you through the authorization process.

GET /auths/twitter/revoke

Revokes an organization's Twitter authorization.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true

Returns Status: `204`

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/auths/twitter/revoke\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api auths revoke service=twitter
```

Via `python`

```
from newslynx.client import API

api = API()
api.auths.revoke("twitter")
```

GET `/auths/facebook`

Fetches an organization's Facebook authorization.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Authorization JSON* object

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/auths/facebook\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api auths get service=facebook
```

Via `python`

```
from newslynx.client import API

api = API()
api.auths.get("facebook")
```

GET `/auths/facebook/grant`

Authorizes NewsLynx to access an organization's Facebook profile.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`redirect_uri`	The url which you would like to send a user back to after an authentication attempt	null	false

Returns An *Authorization JSON* object or, if a `redirect_uri` is provided, a redirection to that location with the added query string parameter `auth_success` to indicate whether or not the authorization request was successful.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/auths/facebook/grant\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api auths grant service=facebook
```

NOTE Will open up a browser to the specified url and run you through the authorization process.

Via `python`

```
from newslynx.client import API

api = API()
api.auths.grant("facebook")
```

NOTE Will open up a browser to the specified url and run you through the authorization process.

GET `/auths/facebook/revoke`

Revokes an organization's Facebook authorization.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns Status: `204`

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/auths/facebook/revoke\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api auths revoke service=facebook
```

Via `python`

```
from newslynx.client import API

api = API()
api.auths.revoke("facebook")
```

Tags

The **Tags** API enables the listing, creating, updating, and deleting of tags.

Tag JSON

All methods, unless otherwise specified, will return one or many tag objects of the following `json` schema. Refer to the *Taxonomy docs* for an explanation of this collection.

A *Subject Tag* takes the following schema:

```
{
    "id": 1,
    "org_id": 1,
    "name": "Politics",
    "slug": "politics",
    "type": "subject",
    "color": "#fc0"
}
```

An *Impact Tag* takes the following schema:

```
{
    "id": 1,
    "org_id": 1,
    "name": "Social Media Share",
    "slug": "social-media-share",
    "type": "impact",
    "color": "#0cf",
    "category": "citation",
    "level": "media"
}
```

GET `/tags`

List all tags associated with an organization, as well as helpful faceted counts.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`type`	A *Tag Type* to filter by.	null	false
`category`	An *Impact Tag Category* to filter by.	null	false
`level`	An *Impact Tag Level* to filter by.	null	false
`sort`	Sort results by a tag field. preface with **-** to sort descending	-created	false

Returns

```
{
    "facets": {
        "levels": {
            "institution": 2,
            "media": 3,
            "individual": 1,
            "internal": 2,
            "community": 2
        },
        "categories": {
            "other": 5,
            "citation": 1,
            "change": 2,
            "achievement": 2
        },
        "types": {
            "impact": 10,
            "subject": 10
        }
    },
    "tags": [
        {
            "content_item_count": 10,
            "name": "Politics",
            "slug": "politics"
            "color": "#13962A",
            "org_id": 1,
            "type": "subject",
            "id": 14
        },
        {
            "category": "change",
            "name": "Legislative Change",
            "slug": "legislative-change",
            "level": "individual",
            "color": "#43E1D8",
            "event_count": 15,
            "org_id": 1,
            "type": "impact",
            "id": 1
        },
        ...
```

```
    ]
}
```

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/tags\?sort=-name\&type=subject\&apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api tags list sort=-name type=subject
```

Via python

```
from newslynx.client import API

api = API()
api.tags.list(sort='-name', type='subject')
```

POST /tags

Create a tag.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body A *Tag JSON* object.

Returns A newly-created *Tag JSON* object.

Example Create a subject tag.

Via CuRL

```
$ curl --data "name=foo&type=subject&color=#fc0" \
http://localhost:5000/api/v1/tags\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api tags create name=foo type=subject color=#fc0
```

Via python

```
from newslynx.client import API

api = API()
api.tags.create(name='foo', type='subject' color='#fc0')
```

Create an impact tag.

Via CuRL

```
$ curl --data "name=bar&type=impact&color=#0cf&category=change&level=institution" \
http://localhost:5000/api/v1/tags\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api tags create name=bar type=impact color=#0cf category=change level=institution
```

Via python

```
from newslynx.client import API

api = API()
api.tags.create(name='bar', type='impact' color='#0cf', category='change', level='institution')
```

GET /tags/:tag_id

Get an individual tag.

NOTE

- This endpoint can accept either a tag id or slug.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A *Tag JSON* object.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/tags/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api tags get id=1
```

Via python

```
from newslynx.client import API

api = API()
api.tags.get(1)
```

PUT | PATCH /tags/:tag_id

Update a tag.

NOTE

- This endpoint can accept either a tag `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body A complete or partial *Tag JSON* object.

Returns An updated *Tag JSON* object.

Example Via `CuRL`

```
$ curl -X PUT -d "color=#fc0aaa" \
  http://localhost:5000/api/v1/tags/21\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api tags update id=1 color=#fc0aaa
```

Via `python`

```
from newslynx.client import API

api = API()
api.tags.update(1, color='#fc0aaa')
```

DELETE /tags/:tag_id

Delete a tag.

NOTE

- This endpoint can accept either a tag `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns Status: `204`

Example Via `CuRL`

```
$ curl -X DELETE http://localhost:5000/api/v1/tags/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api tags delete id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.tags.delete(1)
```

PUT `/tags/:from_tag_id/merge/:to_tag_id`

Merges two tags of the same type. This endpoing will delete the `from_tag_id` and re-associate, depending on the type, all of it's Events or Content Items with the `to_tag_id`.

NOTE

- This endpoint can accept either a tag `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Tag JSON* object for the `to_tag_id`.

Example Via `CuRL`

```
$ curl -X PUT http://localhost:5000/api/v1/tags/1/merge/2\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api tags merge from_id=1 to_id_2
```

Via `python`

```
from newslynx.client import API

api = API()
api.tags.merge(1, 2)
```

GET /tags/categories

Get a list of every *Impact Tag Category*. This endpoint exists to aid in creating dynamic UIs.

Returns A list of every *Impact Tag Category*.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/tags/categories
```

Via newslynx

```
$ newslynx api tags categories
```

Via python

```
from newslynx.client import API

api = API()
api.tags.categories()
```

GET /tags/levels

Get a list of every *Impact Tag Level*. This endpoint exists to aid in creating dynamic UIs.

Returns A list of every *Impact Tag Level*.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/tags/levels
```

Via newslynx

```
$ newslynx api tags levels
```

Via python

```
from newslynx.client import API

api = API()
api.tags.levels()
```

SousChefs

The **SousChefs** API enables the listing / creating / updating / deleting of modules for ingesting and modifying data in NewsLynx. Refer to the *SousChef docs* for more details.

Sous Chef JSON

All methods, unless otherwise specified, will return one or many sous chef objects of the following `json` schema:

```
{
  "id": 3,
  "name": "Event from twitter user.",
  "slug": "twitter-user-to-event",
  "description": "Extracts events from a twitter user's timeline.",
  "runs": "newslynx.sc.events.twitter.User",
  "creates": "events",
  "is_command": false,
  "option_order": ["name", "slug", "decription", "screen_name", ...],
  "options": {
    "screen_name": {
      "required": true,
      "input_type": "text",
      "value_types": ["string"],
      "help": {
        "placeholder": "cspan"
      }
      ...
    }
  }
}
```

If a SousChef creates *Metrics*, It should also explicitly declare which metrics it creates:

```
{
  "name": "Content Share Counts",
  "slug": "share-counts-to-content-metrics-timeseries"
  "description": "Computes a timeseries of share counts for an organization's content items.",
  "runs": "newslynx.sc.metrics.shares.TimeseriesCounts",
  "is_command": false,
  "creates": "metrics",
  "id": 6,
  "option_order": [],
  "options": {
    ...
    },
  "metrics": {
    "facebook_comments": {
      "display_name": "Facebook Comments",
      "type": "cumulative",
      "content_levels": ["timeseries", "summary", "comparison"],
      "org_levels": ["timeseries", "summary", "comparison"]
    },
    ...
  }
}
```

GET /sous-chefs

List all SousChefs, as well as helpful faceted counts.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
is_command	Whether this is runs a non-python script See sous-chefs-runners	null	false
creates	**The collection this SousChef** creates. See sous-chefs-creates	all	false
sous_chefs	A comma-separated list of sous- chefs to return. Each element can be prefaced by with **-** or **!** to exclude it.	null	false

Returns

```
{
  "facets": {
    "creates": {
      "thing": 1,
      "event": 3
    },
    "runners": {
      "python": 4
    }
  },
  "sous_chefs": [
    ...
  ]
}
```

Example Fetch all SousChefs:

Via CuRL

```
$ curl http://localhost:5000/api/v1/sous-chefs\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api sous-chefs list
```

Via `python`

```
from newslynx.client import API

api = API()
api.sous_chefs.list()
```

Fetch all SousChefs that create `events`:

Via `CuRL`

```
$ curl http://localhost:5000/api/v1/sous-chefs\?apikey=$NEWSLYNX_APIKEY\&org=1\&creates=events
```

Via `newslynx`

```
$ newslynx api sous-chefs list creates=events
```

Via `python`

```
from newslynx.client import API

api = API()
api.sous_chefs.list(creates='events')
```

POST /sous-chefs

Create a SousChef.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body A valid *Sous Chef JSON* object.

Returns A newly-created *Sous Chef JSON* object.

Example Create a file like this and save it as `sous-chef.json`

```
{
    "slug": "event-twitter-user-z",
    "name": "Event from twitter user 2",
    "runs": "newslynx.sc.events.twitter.User",
    "description": "Extracts events from a twitter user's timeline.",
    "creates": "events",
    "options": {
        "screen_name": {
          "required": true,
```

```
            "input_type": "text",
            "value_types": ["string"],
            "help": {
              "placeholder": "cspan"
            }
          }
    }
}
```

Alternatively, you can create a `yaml` file called `sous-chef.yaml` for use with `newslynx` or `python`. (Note: you can use `json` as well with these tools):

```
slug: twitter-user-to-event-b
name: Event from twitter user 2
runs: newslynx.sc.events.twitter.User
description: "Extracts events from a twitter user's timeline."
creates: events
options:
    screen_name:
      required: true
      input_type: text
      value_types:
        - string
      help:
        placeholder: cspan
```

Via `CuRL`

```
$ curl -X POST \
     -H 'Content-Type:application/json' \
     --data-binary @sous-chef.json \
     http://localhost:5000/api/v1/sous-chefs\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api sous-chefs create --data=sous-chef.yaml
```

Via `python`

```
import json

from newslynx.client import API


sc = json.load(open('sous-chef.json'))

api = API()
api.sous_chefs.create(**sc)
```

GET /sous-chefs/:sous_chef_id

Fetch an individual SousChef.

NOTE

- This endpoint can accept either a Sous Chef `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A *Sous Chef JSON* object.

Example Via `CuRL`

```
$ curl -X PUT -d http://localhost:5000/api/v1/sous-chefs/twitter-user-to-event\?apikey=$NEWSLYNX_APIK
```

Via `newslynx`

```
$ newslynx api sous-chefs get id=twitter-user-to-event
```

Via `python`

```
from newslynx.client import API

api = API()
api.sous_chefs.get('twitter-user-to-event')
```

PUT /sous-chefs/:sous_chef_id

Update an individual SousChef.

NOTE

- This endpoint can accept either a Sous Chef `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body A complete or parial *Sous Chef JSON* object.

Returns A newly-updated *Sous Chef JSON* object.

Example Add another option to a Sous Chef:

Via `CuRL`

```
$ curl -X PUT screen_name='newslynx' http://localhost:5000/api/v1/sous-chefs/twitter-user-to-event\?a
```

Via `newslynx`

```
$ newslynx api sous-chefs update id=twitter-user-to-event \
  -d '{
        "options":{
          "must_link":{
            "input_type":"radio",
            "input_options":["true", "false"],
            "value_types": ["boolean"],
            "default": false
          }
        }
      }'
```

Via `python`

```
from newslynx.client import API

api = API()

opt = {
        "options":{
          "must_link":{
            "input_type":"radio",
            "input_options":["true", "false"],
            "value_types": ["boolean"],
            "default": False
          }
        }
      }

api.sous_chefs.update('twitter-user-to-event', **opt)
```

Recipes

The **Recipes** API enables the configuration of SousChefs to be scheduled at regular intervals. Refer to the Recipes docs for more details.

Recipe JSON

All methods, unless otherwise specified, will return one or many Recipe objects of the following `json` schema:

```
{
  "status": "stable",
  "updated": "2015-07-22T23:26:08.476376+00:00",
  "sous_chef": "rss-feed-to-article",
  "name": "Ingest Articles from an RSS Feed",
  "created": "2015-07-22T23:18:24.721358+00:00",
```

```
  "traceback": null,
  "org_id": 1,
  "last_run": "2015-07-22T23:26:08.473112+00:00",
  "options": {
    "feed_url": "http://wisconsinwatch.org/feed/"
  },
  "time_of_day": "12:00 AM",
  "last_job": {
    "max_date_last_run": "2015-07-14T16:04:14+00:00"
  },
  "event_counts": {
    "pending": 2,
    "total": 5,
    "deleted": 3
  },
  "schedule_by": "unscheduled",
  "id": 1,
  "minutes": 30,
  "slug": "rss-feed-to-article",
  "crontab": "*/30 * * * *",
  "description": "Extracts articles from an RSS Feed."
}
```


GET /recipes

List all Recipes, as well as helpful faceted counts.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`status`	Filter recipes by their status. Either running, error, stable, or uninitialized	null	false
`scheduled`	Filter recipes by whether or not they are scheduled	null	false
`sort`	Sort results by a Recipe field. preface with **-** to sort descending	-created	false
`recipes`	A comma-separated list of Recipe ids or slugs to query by. Each element can be prefaced by with **-** or **!** to exclude it.	null	false
`sous_chefs`	A comma-separated list of sous- chefs slugs that recipes belong to. Each element can be prefaced by with **-** or **!** to exclude it.	null	false

Returns

```
{
  "facets": {
    "creates": {
      "metrics": 7,
      "content": 1,
      "internal": 2,
      "events": 7
    },
    "statuses": {
      "uninitialized": 2,
      "stable": 15
    },
    "sous_chefs": {
      "twitter-user-to-org-timeseries": 1,
      "google-analytics-to-content-timeseries": 1,
      ...
    },
    "schedules": {
      "scheduled": 1,
      "unscheduled": 16
    }
  },
  "recipes": [
    ...
```

```
  ]
}
```

Example Fetch all Recipes:

Via `CuRL`

```
$ curl http://localhost:5000/api/v1/recipes\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api recipes list
```

Via `python`

```
from newslynx.client import API

api = API()
api.recipes.list()
```

Fetch all Recipes that are not instances of `rss-feed-to-article` SousChefs:

Via `CuRL`

```
$ curl http://localhost:5000/api/v1/recipes\?apikey=$NEWSLYNX_APIKEY\&org=1\&sous_chefs=-rss-feed-to-
```

Via `newslynx`

```
$ newslynx api recipes list sous_chefs=-rss-feed-to-article
```

Via `python`

```
from newslynx.client import API

api = API()
api.recipes.list(sous_chefs="-rss-feed-to-article")
```

POST /recipes

Create a Recipe.

NOTE - Since SousChef options are explicitly declared, you do not need to nest Recipes options under an `options` key in the body of this request. However, if you do, the API will still handle them properly.

- While you may explicitly add a `name`, `slug`, and `description` for a Recipe, if missing, will inherit these fields from it's associated SousChef. In the case of `slug`, a short random hash will be added to ensure uniqueness.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`sous_chef`	The SousChef this Recipe runs. While not required as a param, you must either pass this in here or in the request body	null	false

Body A partial or complete *Recipe JSON* object **with all required SousChef options filled out**. Optionally include the `sous_chef` in the body if not provided as a query string.

Returns A newly-created *Recipe JSON* object.

Example Via `CuRL`

Create a file like this and save it as `recipe.json`:

```
{
  "sous_chef": "rss-feed-to-article",
  "options": {
    "feed_url": "http://nytimes.cat/feed.xml"
  }
}
```

Now run this command:

```
$ curl -X POST \
     -H 'Content-Type:application/json' \
     --data-binary @recipe.json \
     http://localhost:5000/api/v1/recipes\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api recipes create \
  sous_chef=rss-feed-to-article \
  feed_url=http://nytimes.cat/feed.xml
```

Via `python`

```
from newslynx.client import API

api = API()
api.recipes.create(
  sous_chef="rss-feed-to-article",
  feed_url="http://nytimes.cat/feed.xml"
)
```

GET `/recipes/:recipe_id`

Fetch an individual Recipe.

NOTE

- This endpoint can accept either a Recipe `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A *Recipe JSON* object.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/recipes/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api recipes get id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.recipes.get(1)
```

PUT /recipes/:recipe-id

Update an individual Recipe.

NOTE

- This endpoint can accept either a Recipe `id` or `slug`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`sous_chef`	The Sous Chef this Recipe runs. While not required as a param, you must either pass this in here or in the request body	null	false

Body A complete or partial *Recipe JSON* object.

Returns A newly-updated *Recipe JSON* object.

Example Via CuRL

```
$ curl -X PUT \
     -d "feed_url=http://newslynx.org/feed.xml" \
     http://localhost:5000/api/v1/recipes/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api recipes update id=1 feed_url=http://newslynx.org/feed.xml
```

Via python

```
from newslynx.client import API

api = API()
api.recipes.update(1, feed_url="http://newslynx.org/feed.xml")
```

Metrics

The **Metrics** API enables the creation, querying, faceting, updating, and deleting of Metrics. Refer to the *Metrics docs* for more details on what these are.

NOTE - Metrics are exclusively created by *Recipes*. Their settings are specified by *Sous Chefs*.

Metric JSON

All methods, unless otherwise specified, will return one or many Metric objects of the following json schema:

```
{
  "updated": "2015-07-22T23:43:39.752646+00:00",
  "display_name": "Tablet Pageviews",
  "name": "ga_pageviews_tablet",
  "created": "2015-07-22T23:43:39.752631+00:00",
  "agg": "sum",
  "org_levels": [
    "summary"
  ],
  "org_id": 1,
  "faceted": false,
  "content_levels": [
    "summary",
    "comparison"
  ],
  "recipe_id": 10,
  "type": "count",
  "id": 20
}
```

GET /metrics

Filter all metrics.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`content_levels`	A comma-separated list of `content_levels` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`org_levels`	A comma-separated list of `org_levels` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`recipes`	A comma-separated list of Recipe ‘‘id‘‘s or ‘‘slug‘‘s to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`sous_chefs`	A comma-separated list of Sous Chef ‘‘slug‘‘s to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`faceted`	`true` / `false`. Filter metrics by whether or not they have facets.	null	false
`computed`	`true` / `false`. Filter metrics by whether or not they are computed.	null	false

Returns


```
{
  "facets": {
    "computed": 3,
    "org_levels": {
      "timeseries": 16,
      "summary": 31
    },
    "faceted": 2,
    "content_levels": {
      "comparison": 31,
      "timeseries": 16,
      "summary": 33
    },
    "recipes": {
      "internal-refresh-content-summary-metrics-0d5a13": 12,
      "google-analytics-to-content-timeseries-2ecafa": 7,
      "google-analytics-to-content-device-summaries-295220": 3,
```

```
      "google-analytics-to-content-domain-facets-0e0e8a": 2,
      "share-counts-to-content-timeseries-f36b30": 9
    },
    "types": {
      "count": 21,
      "cumulative": 9,
      "computed": 3
    }
  },
  "metrics": [
    ...
  ]
}
```

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/metrics\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api metrics list
```

Via python

```
from newslynx.client import API

api = API()
api.metrics.list()
```

GET /metrics/:metric_id

Fetch an individual metric.

NOTE

- You can pass in a metric's *name* or *id* to this endpoint.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Metric JSON* object.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/metrics/ga_pageviews\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api metrics get id=ga_pageviews
```

Via python

```
from newslynx.client import API

api = API()
api.metrics.get('ga_pageviews')
```

PUT | PATCH /metrics/:metric_id

Update a metric.

NOTE

- You can pass in a metric's name or id to this endpoint.
- You cannot update a metric's name, only it's display_name.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body An partial or complete *Metric JSON* object.

Returns A newly updated *Metric JSON* object.

Example Via CuRL

```
$ curl -X PUT -d 'display_name=Google Analytics Entrances' \
http://localhost:5000/api/v1/metrics/ga_entrances\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api metrics update id=ga_entrances display_name='Google Analytics Entrances'
```

Via python

```
from newslynx.client import API

api = API()
api.metrics.get('ga_entrances', display_name='Google Analytics Entrances')
```

DELETE /metrics/:metric_id

Delete a metric.

NOTE

- You can pass in a metric's `name` or `id` to this endpoint.
- This endpoint will delete all instances of this metric from Timeseries and Summary tables (when applicable).
- If you want to re-create a metric, you'll need to re-create the Recipe which originally created it.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns Status: `204`

Example Via `CuRL`

```
$ curl -X DELETE -d \
http://localhost:5000/api/v1/metrics/ga_entrances\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api metrics delete id=ga_entrances
```

Via `python`

```
from newslynx.client import API

api = API()
api.metrics.delete('ga_entrances')
```

Authors

The **Authors** API enables the creation, update, and deletion of Authors. It also enables programmatic access to the creation and modification of associations between authors and content items.

Author JSON

All methods, unless otherwise specified, will return one or many Metric objects of the following `json` schema:

```
{
  "updated": "2015-06-20T18:15:12.459411+00:00",
  "name": "MERLYNNE JONES",
  "created": "2015-06-20T18:15:12.459397+00:00",
  "org_id": 1,
  "img_url": "http://newslynx.org/merlynne-selfie.jpeg",
  "id": 1,
```

```
  "content_items": [
    ...
  ]
}
```

NOTE All Author ‘‘name‘‘s are stored in ALL CAPS to help prevent duplication.

GET /authors

Fetch all authors for an organization.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
q	A query for an author's name.	null	false
incl_content	Whether or not to include content items associated with the authors.	false	false

Returns A list of endpoint-authors-json objects.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/authors\?q=merlynne&apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api authors list q=merlynne
```

Via python

```
from newslynx.client import API

api = API()
api.authors.list(q=merlynne)
```

POST /authors

Create an Author.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body An endpoint-authors-json object with, at the very minimum, a unique `name`.

Returns A newly-created endpoint-authors-json object.

Example Via `CuRL`

```
$ curl -X POST --data="name=DARTH" \
  http://localhost:5000/api/v1/authors\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api authors create name=DARTH
```

Via `python`

```
from newslynx.client import API

api = API()
api.authors.create(name='DARTH')
```

GET /authors/:author_id

Fetch an individual author.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`incl_content`	Whether or not to include content items associated with the author.	false	false

Returns An endpoint-authors-json object.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/authors/1\?&apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api authors get id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.authors.get(1)
```

PUT | PATCH /authors/:author_id

Update an author.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Body An partial or complete endpoint-authors-json object.

Returns A newly updated endpoint-authors-json object.

Example Via CuRL

```
$ curl -X PUT -d "name=ANNAKIN" \
  http://localhost:5000/api/v1/authors/DARTH\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api authors update id=DARTH name=ANNAKIN
```

Via python

```
from newslynx.client import API

api = API()
api.authors.update('DARTH', name='ANNAKIN')
```

DELETE /authors/:author_id

Delete an author.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true

Returns Status: 204

Example Via CuRL

```
$ curl -X DELETE http://localhost:5000/api/v1/authors/1\&apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api authors delete id=1
```

Via python

```
from newslynx.client import API

api = API()
api.authors.delete(1)
```

PUT /authors/:author_id/content/:content_item_id

Associate an author with a content item.

NOTE

- Will always return the modified list of content items associated with the author.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns A newly updated endpoint-authors-json object with the newly-associated content item included.

Example Via CuRL

```
$ curl -X PUT \
http://localhost:5000/api/v1/authors/1/merge/2\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api authors merge from_id=1 to_id=2
```

Via python

```
from newslynx.client import API

api = API()
api.authors.merge(1, 2)
```

DELETE /authors/:author_id/content/:content_item_id

Remove an association between an author and a content item.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true

Returns Status: 204

Example

```
$ curl -X DELETE \
http://localhost:5000/api/v1/authors/1/content/2\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api authors remove-content-item id=1 content_item_id=2
```

Via python

```
from newslynx.client import API

api = API()
api.authors.remove_content_item(1, 2)
```

PUT /authors/:from_author_id/merge/:to_author_id

Merges an Author with another Author. This method merges the from_author *into* the to_author, transferring all associated content items, and deleting the from_author in the process. This API exists to aid in dealing with duplicate Authors produced by the author extraction process.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true

Returns A newly updated endpoint-authors-json object for the *to_author* with content items from the *from_author* included.

Example Via CuRL

```
$ curl -X PUT \
http://localhost:5000/api/v1/authors/2/merge/3\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api authors remove-content-item id=1 content_item_id=2
```

Via `python`

```
from newslynx.client import API

api = API()
api.authors.remove_content_item(1, 2)
```

Events

The **Events** API enables the creation, querying, faceting, updating, and deleting of Events. Refer to the *Events docs* for more details on what these are.

Event JSON

All methods, unless otherwise specified, will return one or many Event objects of the following `json` schema:

NOTE

- Events with a `status` of `deleted` mean that these Events have been manually deleted by a user or by a recipe. Such events are kept in the database for 7 days and can be restored at any point. After 7 days these events are permanently deleted.


```
{
  "status": "approved",
  "updated": "2015-06-06T22:10:22.137437+00:00",
  "provenance": "recipe",
  "description": "dolores iure eveniet harum dicta totam eos porro sint nisi quasi molestiae sit moll
  "content_items": [
    {
      "url": "http://example.com/d0fe5387-0c98-11e5-963f-6c4008aeb606",
      "id": 39,
      "title": "veritatis eos nisi a"
    }
  ],
  "recipe_id": 1,
  "authors": [
    "Anthony Roob"
  ],
  "id": 173,
  "created": "2015-05-27T23:10:20.852576+00:00",
  "url": "http://example.com/d25cbf6b-0c98-11e5-9e0f-6c4008aeb606",
  "title": "ut odio eos asperior",
  "tag_ids": [
    5
  ],
  "meta": {
    "followers": 78
  },
  "source_id": "facebook-page-to-event:d25cb405-0c98-11e5-b5c0-6c4008aeb606",
  "img_url": "http://example.com/d25cc021-0c98-11e5-b0a9-6c4008aeb606.png",
  "thumbnail": "...",
  "body": "..."
}
```

GET /events

Search and filter all events and return helpful faceted counts.

Params

Returns The Events search endpoint will always return helpful pagination information. Including

- `first` - The first page of the response as a URL.
- `last` - The last page of the response as a URL.
- `next` - The next page of the response (unless the last page is returned) as a URL.
- `prev` - The previous page of the response (unless the first page is returned) as a URL.
- `page` - The current page number.
- `per_page` - The number of results per page.
- `total` - The total number of pages returned.

It will also always return the `total` number of results for all pages.

```
{
  "pagination": {
    "last": "http://localhost:5000/api/v1/events?org=1&apikey=key&page=5&provenance=recipe",
    "total_pages": 5,
    "next": "http://localhost:5000/api/v1/events?status=approved&org=1&apikey=key&page=2&provenance=r
    "per_page": 25,
    "page": 1,
    "first": "http://localhost:5000/api/v1/events?status=approved&org=1&apikey=key&page=1&provenance=
  },
  "total": 104,
  "facets": {
    "categories": {
      ...
    },
    ...
  }
  "events": [
    ...
  ]
}
```

Examples List `approved` events by most recently created.

Via `CuRL`

```
$ curl http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&status\=approved&sort=-
```

Via `newslynx`

```
$ newslynx api events search status=approved sort=-created
```

Via `python`

```
from newslynx.client import API

api = API()
api.events.search(status='approved', sort='-created')
```

Search events created manually.

Via CuRL

```
$ curl http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&provenance\=manual&q=fo
```

Via newslynx

```
$ newslynx api events search q=foobar provenance=manual
```

Via python

```
from newslynx.client import API

api = API()
api.events.search(q='foobar', provenance='manual')
```

List events that only have certain tags and have *not* been created by certain recipes.

Via CuRL

```
$ curl http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&recipe_ids=-1\&tag_ids=
```

Via newslynx

```
$ newslynx api events search recipe_ids='-1' tag_ids='1,2,3'
```

Via python

```
from newslynx.client import API

api = API()
api.events.search(recipe_ids='-1' tag_ids='1,2,3')
```

List events that link to certain content_items and include the Event body in the response:

Via CuRL

```
$ curl http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&content_item_ids=1,-2,
```

Via newslynx

```
$ newslynx api events search recipe_ids='-1' tag_ids='1,2,3'
```

Via python

```
from newslynx.client import API

api = API()
api.events.search(recipe_ids='-1' tag_ids='1,2,3')
```

Facet events by tag levels:

PROTIP: If you just want facets, use `per_page=1` to speed up the request.

```
$ curl http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&per_page=1&facets=level
```

Search Events and only return `id` and `title`:

PROTIP: when submitting a search query , upping the `per_page` limit, limiting the `fields` returned, and limiting the field to search on serves as an effective auto-complete endpoint.

```
$ curl http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&q=foobar&fields=id,titl
```

POST `/events`

Create an event.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`must_link`	Only create if the event contains links to one or more Content Items.	false	false
`recipe_id`	The Recipe that retrieved this Event. If missing, we assume the event is manually created	false	false

Body A *Event JSON* object, but the only required field is `title`. You can also include the following special fields:

- `tag_ids` - An array of tags to assign to this event.
- `content_item_ids` - An array of content items to associate with this event.
- `links` - An array of links you'd like to include when checking for matching Content Items

Source IDs

If you're creating an event that's associated with a `recipe_id`, it's also imperative that you pass in a `source_id`. We use this field to ensure that no duplicate events are created and also to make sure that Events that have been previously `deleted` are not re-created by a Recipe which is continuously polling a data source. If you include a `recipe_id` as a query string, the `source_id` you pass in will be prefixed by the `slug` of this Recipe to ensure that events created by recipes which generate similar source_ids do not conflict.

Dates

If you wish to specify a `created` for an event, just pass it in as ISO 8601 date string. If you include a UTC-Offset, it will be properly convered to UTC. Otherwise it will be assumed to be UTC. If you don't pass in a `created` field, it will be set as the time the Event was ingested.

Provenance

Events created by recipes (AKA: Events that pass a `recipe_id` to the method) will be assigned a `provenance` of `recipe`. All other events are assumed to have been created manually and will be assigned a `provenance` of `manual`

Meta Fields

All fields passed to this method that are not part of the *Event JSON* object will be inserted into the `meta` field.

Returns A newly-created *Event JSON* object. If you specify `must_link=true` and there is no matching ContentItem in the request body, then this method will return `null`.

Examples Create an `approved` event associated with specific `content_item_ids` and `tag_ids`

First, create a file like this and save it as `event.json`

```
{
  "source_id": "fdslakfjdaslkfjasdlkaf",
  "title": "Something else happened.",
  "description": "This was crazy!",
  "body": "<p> This is the transcript of what happened</p>",
  "tag_ids": [1,2],
  "status": "approved",
  "content_item_ids": [1,2]
}
```

Via `CuRL`

```
$ curl -X POST \
     -H 'Content-Type:application/json' \
     --data-binary @event.json \
    http://localhost:5000/api/v1/events\?org\=1\&apikey\=$NEWSLYNX_APIKEY?recipe_id=1
```

Via `newslynx`

```
$ newslynx api events create --data=event.json recipe_id=1
```

Via `python`

```
import json

from newslynx.client import API

api = API()
event = json.load(open('event.json'))
api.events.create(**event)
```

GET /events/:event_id

Fetch an individual event.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Event JSON* object with the `body` included.

Example Via `CuRL`

```
$ curl http://localhost:5000/api/v1/events/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api events get id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.events.get(1)
```

PUT | PATCH /events/:event_id

Update an individual event.

NOTE

- When passing in `tag_ids` and `content_item_ids`, this method will upsert pre-existing associations rather than replacing them.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Event JSON* object.

Examples Update an event's `description`

Via `CuRL`

```
$ curl -X PUT -d "description=This is what happened" \
http://localhost:5000/api/v1/events/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api events update id=1 description='This is what happened'
```

Via `python`

```
from newslynx.client import API

api = API()
api.events.update(1, description='This is what happened')
```

Approve an event by associating it with specific `content_item_ids` and `tag_ids` and setting it's status as `approved`.

First, create a file like this and save it as `event.json`

```
{
  "tag_ids": [3,4],
  "content_item_ids": [1,2]
  "status": "approved"
}
```

Via `CuRL`

```
$ curl -X PUT \
    -H 'Content-Type:application/json' \
    --data-binary @event.json \
    http://localhost:5000/api/v1/events/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api events update id=1 --data=event.json
```

Via `python`

```
import json

from newslynx.client import API

api = API()
event = json.load(open('event.json'))
api.events.update(1, **event)
```

DELETE `/events/:event_id`

Set an Event's `status` as deleted and remove it's associations with Tags and Content Items. Permanently delete an `event` by adding the parameter `force=true`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`force`	Whether or not to permanently delete this event. wish to access.	null	true

Returns Status: `204`

Examples Set an Event's `status` to `deleted`

Via `CuRL`

```
$ curl -X DELETE http://localhost:5000/api/v1/events/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api events delete id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.events.delete(1)
```

Permanently delete an Event.

Via `CuRL`

```
$ curl -X DELETE http://localhost:5000/api/v1/events/1\?org\=1\&force\=true\&apikey\=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api events delete id=1 force=true
```

Via `python`

```
from newslynx.client import API

api = API()
api.events.delete(1, force=True)
```

PUT /events/:event_id/tags/:tag_id

Add a Tag to an Event.

NOTE

- Events must first be "approved" before adding additional Tags.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Event JSON* object.

Example Via CuRL

```
$ curl -X PUT http://localhost:5000/api/v1/events/2/tag/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api events add-tag id=2 tag_id=1
```

Via python

```
from newslynx.client import API

api = API()
api.events.add_tag(2, 1)
```

DELETE /events/:event_id/tags/:tag_id

Remove an associated Tag from an Event.

NOTE

- Events must first be "approved" before removing Tags

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Event JSON* object.

Example Via CuRL

```
$ curl -X DELETE http://localhost:5000/api/v1/events/2/tag/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api events remove-tag id=2 tag_id=1
```

Via python

```
from newslynx.client import API

api = API()
api.events.remove_tag(2, 1)
```

PUT /events/:event_id/content/:content_item_id

Associate an Event with a Content Item.

NOTE

- Events must first be "approved" before adding additional Content Items.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Event JSON* object.

Example Via CuRL

```
$ curl -X PUT http://localhost:5000/api/v1/events/2/content/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api events add-content-item id=2 content_item_id=1
```

Via python

```
from newslynx.client import API

api = API()
api.events.add_content_item(2, 1)
```

DELETE /events/:event_id/content/:content_item_id

Remove an associated Content Item from an Event.

NOTE

- Events must first be "approved" before removing Content Items.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Event JSON* object.

Example Via CuRL

```
$ curl -X DELETE http://localhost:5000/api/v1/events/2/content/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api events remove-content-item id=2 content_item_id=1
```

Via python

```
from newslynx.client import API

api = API()
api.events.remove_content_item(2, 1)
```

Content Items

The **Content Items** API enables the creation, querying, faceting, updating, and deleting of Content Items. Refer to the *Content Items docs* for more details on what these are.

Content Item JSON

All methods, unless otherwise specified, will return one or many Content Item objects of the following json schema:

NOTE

- Events with a status of deleted mean that these Events have been manually deleted by a user or by a recipe. Such events are kept in the database for 7 days and can be restored at any point. After 7 days these events are permanently deleted.

```
{
  "body": "...",
  "domain": "example.com",
  "site_name": "Example News",
  "description": "id voluptas voluptatem ea illum quae nam ab fugiat praesentium non libero quo in no
  "created": "2015-03-17T19:15:14.152661+00:00",
  "url": "http://example.com/4be5aec2-1778-11e5-b940-6c4008aeb606",
  "subject_tag_ids": [
    19
  ],
  "impact_tag_ids": [
    12, 13
  ],
  "provenance": "recipe",
  "org_id": 1,
  "updated": "2015-06-20T18:15:14.156157+00:00",
  "favicon": "http://example.com/favicon.ico",
  "metrics": {
    "facebook_comments": 55508,
    "total_events": 1,
    ...
  },
  "recipe_id": 1,
  "meta": {},
```

```
  "authors": [
    {
      "id": 3,
      "name": "MERLYNNE JONES"
    }
  ],
  "title": "id ab modi unde aliq",
  "type": "article",
  "id": 37,
  "img_url": "http://example.com/d25cc021-0c98-11e5-b0a9-6c4008aeb606.png",
  "thumbnail": "...",
}
```


GET `/content`

Search and filter all content items and return helpful faceted counts.

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`q`	A search query. Will search on `body`, `title`, `description`, `authors` & `meta`. Please refer to the Postgres Search docs for details on query syntax.	null	false
`search`	The field to search on. Either: `body`, `title`, `description`, `authors` `meta`, or `all`.	`all`	false
`fields`	A comma-separated list of fields to include in the response.	null	false
`incl_body`	Whether or not to include the body of the content item in the response.	false	false
`incl_img`	Whether or not to include the img_url and thumbnail in the response.	false	false
`incl_metrics`	Whether or not to include metrics in the response.	true	false
`created_after`	An ISO-8601 date to filter results by.	null	false
`created_before`	An ISO-8601 date to filter results by.	null	false

Continued on next page

Table 1.1 – continued from previous page

Parameter	Description	Default	Required
`updated_after`	An ISO-8601 date to filter results by.	null	false
`updated_before`	An ISO-8601 date to filter results by.	null	false
`sort`	Sort results by an event field. preface with **-** to sort descending. When submitting a search query, use `relevance` to sort by match rank. To sort by metrics, use the syntax `metrics.{metric_name}`	-created.	false
`type`	A type to filter results by. See the *docs* for more details on these.	null	false
`provenance`	A provenance to filter results by. Choose from `manual` or `recipe`.	null	false
`ids`	A comma-separated list of content item `ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`sort_ids`	Whether or not use order of the above `ids` to sort the results by. Will override arguments passed to `sort`.	false	false
`url`	A url to filter results by.	null	false
`url_regex`	A regex to test on urls.	null	false
`domain`	A domain to filter results by	null	false
`author_ids`	A comma-separated list of ‘‘author_ids‘‘to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`impact_tag_ids`	A comma-separated list of `impact_tag_ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`subject_tag_ids`	A comma-separated list of `subject_tag_ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`levels`	A comma-separated list of Tag `levels` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false

Continued on next page

Table 1.1 – continued from previous page

Parameter	Description	Default	Required
`categories`	A comma-separated list of Tag `categories` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`recipe_ids`	A comma-separated list of `recipe_ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`sous_chefs`	A comma-separated list of Sous Chef slugs to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`facets`	A comma-separated list of faceted counts to include in the response. Choose from `subect_tags`, `events`, `levels`, `categories`, `sous_chefs`, `recipes`, `types`, `site_names`, `domains`, `authors`, `event_statuses` `impact_tags` or `all`.	null	false
`page`	The page number of the results.	1	false
`per_page`	**The number of results to return** per page. Max is 100.	25	false

Params

Returns The Content Items search endpoint will always return helpful pagination information. Including

- `first` - The first page of the response as a URL.
- `last` - The last page of the response as a URL.
- `next` - The next page of the response (unless the last page is returned) as a URL.
- `prev` - The previous page of the response (unless the first page is returned) as a URL.
- `page` - The current page number.
- `per_page` - The number of results per page.
- `total` - The total number of pages returned.

```
{
  "pagination": {
    "last": "http://localhost:5000/api/v1/events?org=1&apikey=key&page=5&provenance=recipe",
    "total_pages": 5,
```

```
    "next": "http://localhost:5000/api/v1/events?status=approved&org=1&apikey=key&page=2&provenance=
    "per_page": 25,
    "page": 1,
    "first": "http://localhost:5000/api/v1/events?status=approved&org=1&apikey=key&page=1&provenance=
  },
  "total": 104,
  "facets": {
    "tags": {
      ...
    },
    ...
  }
  "content_items": [
    ...
  ]
}
```

Examples List content items or type `article` by most recently created.

Via CuRL

```
$ curl http://localhost:5000/api/v1/content\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&type\=article&sort=-cr
```

Via newslynx

```
$ newslynx api content search type=article sort=-created
```

Via python

```
from newslynx.client import API

api = API()
api.content.search(type='article', sort='-created')
```

Search content items created manually.

```

```

Via CuRL

```
$ curl http://localhost:5000/api/v1/content\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&provenance\=manual\&q
```

Via newslynx

```
$ newslynx api content search provenance=manual q=uqbar
```

Via python

```
from newslynx.client import API

api = API()
api.content.search(provenance='manual', q='uqbar')
```

List content items that only have certain subject tags and have *not* been created by certain recipes.

Via `CuRL`

```
$ curl http://localhost:5000/api/v1/content\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&recipe_ids=-1\&subject
```

Via `newslynx`

```
$ newslynx api content search recipe_ids=-1 tag_ids=1,2,3
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.search(recipe_ids='-1', tag_ids='1,2,3')
```

Sort content items by their number of Twitter Shares.

Via `CuRL`

```
$ curl http://localhost:5000/api/v1/content\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&sort=-metrics.twitter_
```

Via `newslynx`

```
$ newslynx api content search sort=-metrics.twitter_shares
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.search(sort="-metrics.twitter_shares")
```

Facet content items by the tag levels of associated events:

PROTIP: If you just want facets, use `per_page=1` to speed up the request.

```
$ curl http://localhost:5000/api/v1/content\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&per_page=1&facets=leve
```

Search Content Items and only return `id` and `title`:

PROTIP: when submitting a search query , upping the `per_page` limit, limiting the `fields` returned, and limiting the field to search on serves as an effective auto-complete endpoint.

```
$ curl http://localhost:5000/api/v1/content\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&q=foobar&fields=id,tit
```

POST `/content`

Create an content item.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
"extract"	Use extract API to construct the content item. When `true` you only need to provide a `url` and `type` in the body of the request. All other fields provided will take precedence over extracted ones.	false	false

Body A *Content Item JSON* object, but the only required field is `title`. You can also include the following special fields:

- `tag_ids` - An array of tags to assign to this content item.
- `author_ids` - An array of author ids to assign to this content item.
- `authors` - An array of author names to assign to this content item. The API will attempt to reconcile these names with existing authors, if the author does not exist, the API will create one.

Provenance

Content items created by recipes (AKA: content items that pass a `recipe_id` to the method) will be assigned a `provenance` of `recipe`. All other content items are assumed to have been created manually and will be assigned a `provenance` of `manual`.

Meta Fields

All fields passed to this method that are not part of the *Content Item JSON* object will be inserted into the `meta` field.

Returns A newly-created *Content Item JSON* object.

Examples Via `CuRL`

```
$ curl --data "url=https://projects.propublica.org/killing-the-colorado/story/wasting-water-out-west-
  http://localhost:5000/api/v1/content\?apikey=$NEWSLYNX_APIKEY\&org=1\&extract=true
```

Via `newslynx`

```
$ newslynx api content create \
    url="https://projects.propublica.org/killing-the-colorado/story/wasting-water-out-west-use-it-or-
    type=article
    extract=true
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.create(
    url="https://projects.propublica.org/killing-the-colorado/story/wasting-water-out-west-use-it-or-
    type="article",
    extract=True
  )
```

GET /content/:content_item_id

Fetch an individual content item.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An *Content Item JSON* object with body, img_url, thumbnail, metrics always included.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/content/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via newslynx

```
$ newslynx api content get id=1
```

Via python

```
from newslynx.client import API

api = API()
api.content.get(1)
```

PUT | PATCH /content/:content_item_id

Update an individual content item id.

NOTE

- When passing in tag_ids this method will upsert pre-existing associations rather than replacing them.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Content Item JSON* object.

Examples Update a content item's `description`.

Via `CuRL`

```
$ curl -X PUT -d "description=This is what this story was about in a gist" \
http://localhost:5000/api/v1/content/1\?apikey=$NEWSLYNX_APIKEY\&org=1
```

Via `newslynx`

```
$ newslynx api content update id=1 description='This is what this story was about in a gist'
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.update(1, description='This is what this story was about in a gist')
```

DELETE /content/:content_item_id

Delete a content item and all of it's associate metrics.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns Status: `204`

Example Via `CuRL`

```
$ curl -X DELETE http://localhost:5000/api/v1/content/1\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api content delete id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.delete(1)
```

PUT /content/:content_item_id/tag/:tag_id

Add a subject tag to a content item.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Content Item JSON* object.

Example Via CuRL .. code-block:: bash

$ curl -X PUT http://localhost:5000/api/v1/content/2/tags/15?org=1&apikey=$NEWSLYNX_APIKEY

Via newslynx

```
$ newslynx api content add-tag id=2 tag_id=15
```

Via python

```
from newslynx.client import API

api = API()
api.content.add_tag(2, 15)
```

DELETE /content/:content_item_id/tags/:tag_id

Remove a subject tag from a content item.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false

Returns An updated *Event JSON* object.

Example Via CuRL

```
$ curl -X DELETE http://localhost:5000/api/v1/content/2/tags/15\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content remove-tag id=2 tag_id=15
```

Via python

```
from newslynx.client import API

api = API()
api.content.remove_tag(2, 15)
```

Content Timeseries Metrics

The content timeseries API allows you to query for all metrics that have been set with a `content_level` of `timeseries`. Please refer to the *Metrics docs* for details on how this works.

Content Timeseries JSON

All endpoints unless otherwise will return the following `json` schema. Note that all metrics will have been given the name specified in their accompying *schema*.

```
{
  "content_item_id": 1,
  "datetime": "2015-07-25T10:00:00+00:00",
  "facebook_comments": 47,
  "linkedin_shares": 90,
  "ga_avg_time_on_page": 0.28,
  "ga_total_time_on_page": 202,
  "reddit_upvotes": 40,
  ...
}
```

GET `/content/timeseries`

Query the entire content timeseries store for an organization. This endpoint enables you to query content items by their associated Tags, Events, and Authors. It also allows you to aggregate at a specified date `unit` as well as by `datetime` so as to enable quick construction of aggregate timeseries for a Tag, Author, Event, or even an entire organization.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true
`localize`	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
`unit`	The datetime `unit` to aggreate metrics by. Choose from `hour`, `day`, `month`, or `null` to return full aggregations. Note that metrics will be aggregated according to their specified `agg` function.	day	false
`sparse`	Do not fill in missing dates in the timeseries. If `false`, will set the value for missing v metrics to `0` '.	true	false
`group_by_id`	Whether to aggregate by `content_item_id`. If `false`, will return aggregate for all content items by the specified date `unit`	true	false
`rm_nulls`	Whether to remove `null` metrics from the timeseries.	false	false
`transform`	Apply a transformation to the timeseries. Currently the only transformation is `cumulative`. This will turn all metrics with an `agg` of `sum` into cumulative sums.	false	false
`before`	An ISO-8601 date to filter results by.	null	false
`after`	An ISO-8601 date to filter results by.	5 days ago	false
`author_ids`	A comma-separated list of ''author_ids''to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`impact_tag_ids`	A comma-separated list of `impact_tag_ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`event_ids`	A comma-separated list of `event_ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`ids`	A comma-separated list of `content_item_ids` to filter results by. Preface any element with **!** or **-** to exclude it.	null	false
`select`	A comma-separated list of Metrics to include in the to	•	false

Returns An list of endpoint-content-timeseries-json objects.

Examples Get the totals for three content items.

Via CuRL

```
$ curl http://localhost:5000/api/v1/content/timeseries\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&unit=null\&
```

Via newslynx

```
$ newslynx api content list-timeseries unit=null ids=1,2,3
```

Via python

```
from newslynx.client import API

api = API()
api.content.list_timeseries(unit='null', ids='1,2,3')
```

Get the hourly timeseries for all of an Author's Content Items.

Via CuRL

```
$ curl http://localhost:5000/api/v1/content/timeseries\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&unit=hour\&
```

Via newslynx

```
$ newslynx api content list-timeseries unit=hour author_ids=1 group_by_id=f
```

Via python

```
from newslynx.client import API

api = API()
api.content.list_timeseries(unit='hour', author_ids='1', group_by_id=False)
```

Get the daily cumulative sums for an entire organization.

Via CuRL

```
$ curl http://localhost:5000/api/v1/content/timeseries\?org\=1\&apikey\=$NEWSLYNX_APIKEY\&unit=day\&t
```

Via newslynx

```
$ newslynx api content list-timeseries unit=day group_by_id=f transform=cumulative
```

Via python

```
from newslynx.client import API

api = API()
api.content.list_timeseries(
  unit='day', group_by_id=False,
```

```
  transform='cumulative'
)
```

GET /content/:content_item_id/timeseries

Query timeseries metrics for a single content item.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
localize	Return dates in the org's specified timezone. If *false* dates will be returned in UTC.	false	false
unit	The datetime unit to aggreate metrics by. Choose from hour, day, month, or null to return full aggregations. Note that metrics will be aggregated according to their specified agg function.	hour	false
sparse	Do not fill in missing dates in the timeseries. If false, will set the value for missing v metrics to 0 '.	true	false
group_by_id	Whether to aggregate by content_item_id. If false, will return aggregate for all content items by the specified date unit	true	false
rm_nulls	Whether to remove null metrics from the timeseries.	false	false
transform	Apply a transformation to the timeseries. Currently the only transformation is cumulative. This will turn all metrics with an agg of sum into cumulative sums.	false	false
before	An ISO-8601 date to filter results by.	null	false
after	An ISO-8601 date to filter results by.	30 days ago	false
select	A comma-separated list of Metrics to include in the to response. If * will return all availabled metrics. Preface any element with ! or - to exclude it.	•	false

Returns An list of endpoint-content-timeseries-json objects.

Example Get an individual timeseries.

Via CuRL

```
$ curl http://localhost:5000/api/v1/content/1/timeseries\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content get-timeseries id=1
```

Via python

```
from newslynx.client import API

api = API()
api.content.get_timeseries(1)
```

Content Summary Metrics

Utilities for rolling up content timeseries metrics.

PUT /content/summary

Refresh summaries of content timeseries metrics. These will show up in the *Content Search API* and when retrieving *individual Content Items*.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
since	Refresh content items whose timeseries have been updated in the last X hours.	24	true

Returns

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

Example Via CuRL

```
$ curl -X PUT http://localhost:5000/api/v1/content/summary\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content refresh-summaries
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.refresh_summaries()
```

PUT /content/:content_item_id/summary

Refresh summaries of content timeseries metrics for an invidual content item. These will show up in the *Content Search API* and when retrieving *individual Content Items*.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns

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

Example Get an individual timeseries.

Via `CuRL`

```
$ curl -X PUT http://localhost:5000/api/v1/content/1/summary\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via `newslynx`

```
$ newslynx api content refresh-summary id=1
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.refresh_summary(1)
```

Comparisons

The Comparison API enables the summarization of the Summary metrics for various facets at various levels - `content`, `orgs`. These comparisons power the App's article comparison view which allows a user to compare a metric for a single Content Item against the distribution of this Metrc for all Content Items or a particular subset of Content Items. See the *Metric docs* for more details on how these work.

Comparisons JSON

Unless otherwise indicated, all comparison endpoints wil return the following `json` schema. Here, comparisons are broken out into `types` with the name of the type being the top-level key. In the example below, the comparison type is `all`. Each element of the list represents an individual metrics. Fields prefaced by `per` signify the value of this metric at it's Xth percentile. So, for the below example, you would interpret `per_95` to mean "the 95th percentile of `facebook_comments` for this facet is `8824.7`.

```
{
  "all" : [
    {
      "per_95": 8824.7,
      "per_2_5": 7492.3,
      "per_40": 8134.8,
      "per_70": 8486.2,
      "per_97_5": 8837.98,
      "per_90": 8672.4,
      "per_80": 8584.6,
      "per_30": 8055.4,
      "max": 8883,
      "metric": "facebook_comments",
      "min": 7438,
      "median": 8254,
      "per_60": 8387.8,
      "per_20": 7827.6,
      "per_5": 7524.35,
      "per_10": 7580.4,
      "mean": 8217.58
    },
    ...
  ]
}
```

In the case a comparison type returns multiple sub-facets, the structure will be as described below. In this case, you would interpret `per_95` to mean "the 95th percentile of `facebook_comments` for the Content Items with a Subject Tag of ID `10` is `8540.5`.

```
{
  "subject_tags" : {
      "10": [
        {
          "per_95": 8540.5,
          "per_2_5": 8119.25,
          "per_40": 8222,
          "per_70": 8422,
          "per_97_5": 8559.25,
          "per_90": 8503,
          "per_80": 8428,
          "per_30": 8217,
          "max": 8578,
          "metric": "facebook_comments",
          "min": 8106,
          "median": 8319,
          "per_60": 8416,
          "per_20": 8212,
          "per_5": 8132.5,
          "per_10": 8159,
```

```
            "mean": 8327
          },
          ...
        ]
    }
}
```

GET /content/comparisons

List all content comparison metrics.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
refresh	Whether to refresh the cache	false	false
cache_details	Return details about how long this comparison has been cached	false	false

Returns A endpoint-comparisons-json object.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/content/comparisons\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content list-comparisons
```

Via python

```
from newslynx.client import API

api = API()
api.content.list_comparisons()
```

PUT /content/comparisons

Update all comparison metrics.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true

Returns

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

Example Via CuRL

```
$ curl -X PUT http://localhost:5000/api/v1/content/comparisons\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content refresh-summaries
```

Via python

```
from newslynx.client import API

api = API()
api.content.refresh_summaries()
```

GET /content/comparisons/:comparison_type

Get just this comparison-type. Choose from:

- all - Compare against all content items
- type - Compare against each Content Item type
- impact-tags - Compare against Content Items with specific Impact Tags.
- subject-tags - Compare against Content Items with specific Subject Tags.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
refresh	Whether to refresh the cache.	false	false
cache_details	Return details about how long this comparison has been cached	false	false

Returns A endpoint-comparisons-json object.

Example Via CuRL

```
$ curl http://localhost:5000/api/v1/content/comparisons/impact-tags\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content get-comparison type=impact-tags
```

Via python

```
from newslynx.client import API

api = API()
api.content.get_comparison(type='impact-tags')
```

PUT /content/comparisons/:comparison-type

Refresh just this comparison `type`.

Params

Parameter	Description	Default	Required
`apikey`	Your `apikey`	null	true
`org`	The organization's `id` or `slug` you wish to access.	null	true

Returns

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

Example Via `CuRL`

```
$ curl -X PUT http://localhost:5000/api/v1/content/comparisons/impact-tags\?org\=1\&apikey\=$NEWSLYN
```

Via `newslynx`

```
$ newslynx api content refresh-comparison type=impact-tags
```

Via `python`

```
from newslynx.client import API

api = API()
api.content.refresh_comparison(type='impact-tags')
```

Make Comparisons

This API utilizes the Comparison API to return percentile rankings of Content Items by each metric.

Make Comparisons JSON

This endpoint mirrors the structure of the Comparison API endpoints-comparisons-json. However, instead of a list of distributions for all metrics, it leverages these distributions to return percentile rankings for an individual Content Item by all of its metrics. Here, we would interpret `facebook_comment` as meaning, "this Content Item ranks in the 90% for `facebook_comments` versus all Content Items with a Subject Tag of `10`"

```
{
  "content_item_id": 1,
  "comparisons": {
    "subject_tags" : {
        "10": {
          "facebook_comments": 90,
          "ga_pageviews": 40,
          "twitter_shares": 10,
          ...
```

```
        },
        ...
    },
    ...
  }
}
```

GET /content/:content_item_id/comparisons

Compare this Content Item by all comparison-types.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
refresh	Whether to refresh the underlying comparison cache.	false	false

Returns A endpoint-make-comparisons-json object.

Example Via CuRL

```
$ curl -X PUT http://localhost:5000/api/v1/content/1/comparisons\?org\=1\&apikey\=$NEWSLYNX_APIKEY
```

Via newslynx

```
$ newslynx api content make-comparisons id=1
```

Via python

```
from newslynx.client import API

api = API()
api.content.make_comparisons(1)
```

GET /content/:content_item_id/comparisons/:comparison-type

Compare this Content Item by a particular comparison-type.

Params

Parameter	Description	Default	Required
apikey	Your apikey	null	true
org	The organization's id or slug you wish to access.	null	true
refresh	Whether to refresh the underlying comparison cache.	false	false

Returns A endpoint-make-comparisons-json object.

Example Via CuRL

```
$ curl -X PUT http://localhost:5000/api/v1/content/1/comparisons/impact-tags\?org\=1\&apikey\=$NEWSLY
```

Via newslynx

```
$ newslynx api content make-comparison id=1 type=impact-tags
```

Via python

```
from newslynx.client import API

api = API()
api.content.make_comparison(1, type="impact-tags")
```

Bulk Creation

The **Bulk Creation API** API enables efficient upsert of new Metrics, Content Items, and Events. This endpoint is designed to help SousChefs effectively bring data into NewsLynx. All methods are executed within a Task queue and return references to the :ref: *endpoints-jobs* API.

Jobs

The **Jobs** API enables the monitoring of processes executed in the Task Queue.

Extract

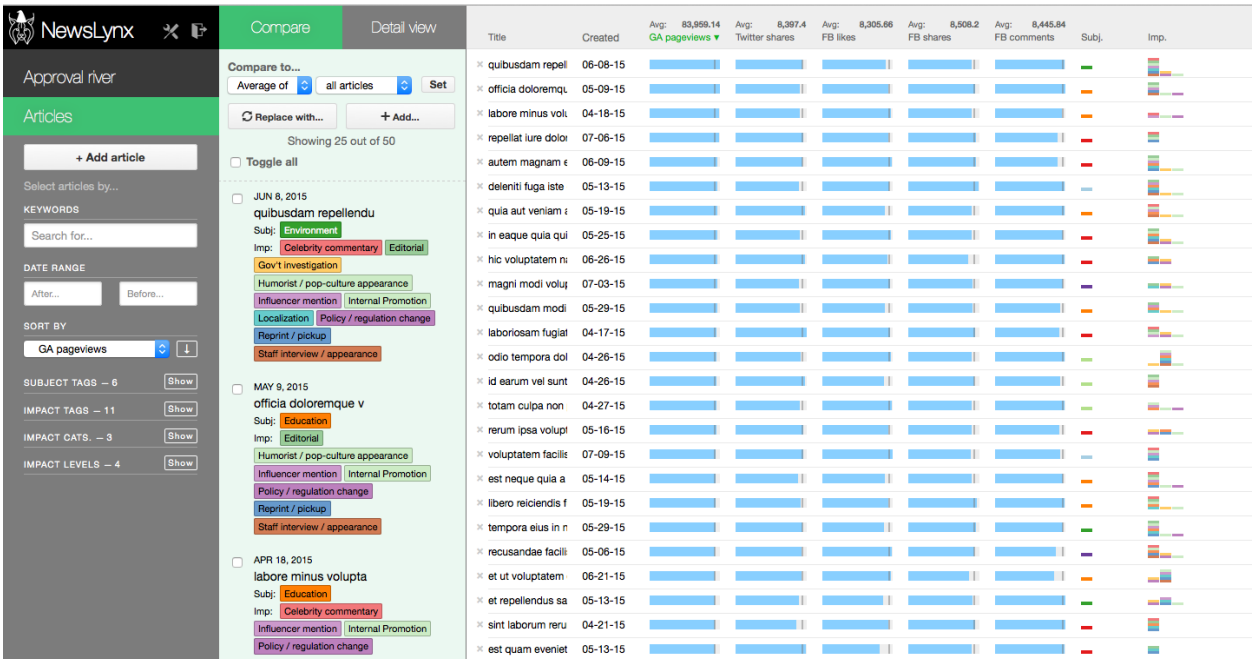
The **Extract** API...

1.12.5 Compression

All endpoints can return responses with gzip compression so long as the Accept-Encoding header of the request is set to gzip.

1.13 NewsLynx App Architecture

To learn how the app works, read the guide on GitHub.



1.14 Command Line Interface

Documentation in progress...