elex

Release 2.4.3

Jul 09, 2018

Contents

1 Important links **3**

2 Disclaimer **5**
- 2.1 Elex projects and implementations . . . 5
- 2.2 News . . . 5
- 2.3 Using the FTP system? . . . 6
- 2.4 Features . . . 6
- 2.5 Table of contents . . . 6

Python Module Index **41**

Get database-ready election results from the Associated Press Election API v2.0.

Elex is designed to be fast, friendly, and largely agnostic to stack/language/database choice. Basic usage is as simple as:

```
elex results 2016-03-01 > results.csv
```

CHAPTER 1

Important links

- Documentation: http://elex.readthedocs.org/
- Repository: https://github.com/newsdev/elex/
- Issues: https://github.com/newsdev/elex/issues
- Roadmap: https://github.com/newsdev/elex/milestones

CHAPTER 2

Disclaimer

Elex was developed by The New York Times and NPR and not in concert with the Associated Press. Though we plan on using Elex for the 2016 cycle, there is no guarantee that this software will work for you. If you're thinking about using Elex, check out the license and contact the authors.

2.1 Elex projects and implementations

NPR

- NPR loader: A simple reference data loader for PostgreSQL.

New York Times

- New York Times Elex loader: A more sophisticated data loader for PostgreSQL.
- New York Times AP Deja Vu: A webservice to replay JSON captured during an election.
- New York Times Elex admin: An admin interface for Elex data loaded with the New York Times loader written in Flask.

Experimental

- node-elex-admin: Incomplete node-based admin interface.
- elex-webVideoTextCrawler: Convert Elex data into HTML5 text track for live video streaming.

2.2 News

- Introducing Elex, A Tool To Make Election Coverage Better For Everyone, Jeremy Bowers and David Eads, Source
- NPR and The New York Times teamed up to make election reporting faster, Benjamin Mullin, Poynter

2.3 Using the FTP system?

Use the Los Angeles Times' python-elections library.

The New York Times has a sample implementation that demonstrates how you might integrate the FTP loader with your Elex-based system.

2.4 Features

- Uses v2.1 of the Associated Press Election API **NOTE**: Requires a (paid) account with the AP.
- Intuitive command line interface: Get data as CSV or JSON and pipe to the data tool of your choice.
- Friendly Python API for use as a library.
- Simple election recording (to MongoDB).
- Comprehensive tests.
- Extensive documentation.
- Fast and correct.

2.5 Table of contents

2.5.1 Installation

Quick install

Install the Python library:

```
pip install elex
```

Set your AP API key:

```
export AP_API_KEY=<MY_AP_API_KEY>
```

On Windows machines, use `setx` instead

```
setx AP_API_KEY=<MY_AP_API_KEY>
```

Note: `Setx` sets a permanent user level environment variable. To set a machine level variable use `\m` option

Optional requirements

- MongoDB (for recording raw results during tests and elections)

Install walkthrough with virtualenv

If you've set up and run Python projects before, you may have your own process, and the *Quick Install* instructions can get you going. But if you're fairly new to Python development, or if you're not familiar with the benefits of using a virtual environment, these tips are for you.

Set up some base tools

The NPR Visuals Team's guide to setting up a development environment is wonderful. Walking through the entire guide is highly recommended; your machine will be much happier for it, and you'll feel prepared for a lot of things beyond just Elex.

For now, though, the most important piece is "Chapter 2: Install Virtualenv." At the very least, step through that section and install `virtualenv` and `virtualenvwrapper`, two tools that help you use virtual environments for your Python projects.

Note: Virtual environments let you compartmentalize projects and the Python tools you install to work on them. You can create as many virtual environments as you like. When you "activate" one of them, you can feel comfortable installing new libraries, because if things break, no problem. Delete that environment and start again; your global settings haven't been touched. When you have things working just right, you can "freeze" the environment to create a list of installed packages so someone else can replicate it. Learning to love virtual environments makes you more efficient _and_ less stressed.

Once you've installed `virtualenv` and `virtualenvwrapper`, then added the appropriate trigger to your `.bash_profile` as described in the NPR Visuals guide, you're ready to set up a pristine Elex environment.

Install Elex

The `virtualenvwrapper` tool gives you access to several commands for creating and managing virtual environments. To create a fresh environment for Elex, run this from your command line:

```
mkvirtualenv elex
```

Your new environment won't know about or have access to any Python tools you've installed elsewhere, which is exactly what you want. The `mkvirtualenv` command will automatically activate your new environment for you, and your command prompt should reflect it. You should see something like:

```
(elex) username@host: ~/your/path $
```

For reference, to turn off an active environment, run the `deactivate` command:

```
deactivate
```

And to enable an environment, run `workon` followed by the environment's name:

```
workon elex
```

With your new "elex" environment activated, installing the Elex library itself is easy:

```
pip install elex
```

That will download Elex and add it to your virtual environment, along with all the libraries it depends on. Just for fun, you can print to screen everything that was installed:

```
pip freeze
```

Now the Elex code will be available to you any time you activate your "elex" environment. You'll still need a project API key to actually run commands, so with "elex" active, add the key you should have received from AP:

```
export AP_API_KEY=your_api_key_string
```

And with that in place, Elex should work as expected. You can test with any of the tutorial commands, like:

```
elex races 11-03-2015 -o json
```

Some extra tricks

Automatically set your API key

If you've followed the instructions above, you should already have your `AP_API_KEY` set. When you `export` a variable, however, it's only available until your session ends. It's tedious to set something like that manually every time you start a new project session, though. Thankfully `virtualenvwrapper` provides an easy way to automatically load variables each time you activate an environment.

Open a new tab in your terminal, and:

```
workon elex
cdvirtualenv
open bin/postactivate
```

This will activate your "elex" environment, navigate to its internal directory on your machine, then use your text editor to open a file called `postactivate`. Any code you put in this file will be run immediately after you activate that environment. So just add:

```
export AP_API_KEY=your_api_key_string
echo "AP_API_KEY set"
```

Then save and close. From now on, every time you activate a new session of your "elex" environment, your API key will automatically be available (and you'll get a little "AP_API_KEY set" reminder printed to screen).

Make human-readable JSON

You might notice that generating JSON with an Elex command like `elex races 11-03-2015 -o json` will put all the results on one line. This is great for keeping file sizes smaller, and it's perfectly readable by other machines. But if you're trying to see what properties are available in the JSON generated by different Elex commands, it's not particularly human-friendly. Fortunately, Elex provides a shortcut to display human-formatted json, the `--format-json` flag.

```
elex races 11-03-2015 -o json --format-json
```

Or to save to a flat file you can inspect later:

```
elex races 11-03-2015 -o json --format-json > races.json
```

2.5.2 Tutorial

Command Line Interface

This tool is primarily designed for use on the command line using standard *NIX operations like pipes and output redirection.

To write a stream of races in CSV format to your terminal, run:

```
elex races '11-03-2015'
```

To write this data to a file:

```
elex races '11-03-2015' > races.csv
```

To pipe it into PostgreSQL:

```
elex races 11-03-2015 | psql elections -c "COPY races FROM stdin DELIMITER ',' CSV␣
↪HEADER;"```
```

To get JSON output:

```
elex races 11-03-2015 -o json
```

Output can be piped to tools like sed, awk, jq, or csvkit for further processing.

Python Modules

Perhaps you'd like to use Python objects in your application. This is how you would call the Elex modules directly without using the command line tool.

```
from elex import api

# Setup and call the AP API.
e = api.Election(electiondate='2015-11-03', datafile=None, testresults=False,␣
↪liveresults=True, is_test=False)
raw_races = e.get_raw_races()
race_objs = e.get_race_objects(raw_races)

# Get lists of Python objects for each of the core models.
ballot_measures = e.ballot_measures
candidate_reporting_units = e.candidate_reporting_units
candidates = e.candidates
races = e.races
reporting_units = e.reporting_units
results = e.results
```

2.5.3 Command line interface

Commands and flags

```
commands:

  ballot-measures
```

(continues on next page)

(continued from previous page)

```
    Get ballot measures

  candidate-reporting-units
    Get candidate reporting units (without results)

  candidates
    Get candidates

  clear-cache
    Clear the elex response cache

  delegates
    Get all delegate reports

  elections
    Get list of available elections

  governor-trends
    Get governor trend report

  house-trends
    Get US House trend report

  next-election
    Get the next election (if date is specified, will be relative to that date,␣
↪otherwise will use today's date)

  races
    Get races

  reporting-units
    Get reporting units

  results
    Get results

  senate-trends
    Get US Senate trend report

positional arguments:
  date                  Election date (e.g. "2015-11-03"; most common date
                        formats accepted).

optional arguments:
  -h, --help            show this help message and exit
  --debug               toggle debug output
  --quiet               suppress all output
  -o {json,csv}         output format (default: csv)
  -t, --test            Use testing API calls
  -n, --not-live        Do not use live data API calls
  -d DATA_FILE, --data-file DATA_FILE
                        Specify data file instead of making HTTP request when
                        using election commands like `elex results` and `elex
                        races`.
  --delegate-sum-file DELEGATE_SUM_FILE
                        Specify delegate sum report file instead of making
                        HTTP request when using `elex delegates`
```

(continues on next page)

(continued from previous page)

```
  --delegate-super-file DELEGATE_SUPER_FILE
                        Specify delegate super report file instead of making
                        HTTP request when using `elex delegates`
  --trend-file TREND_FILE
                        Specify trend file instead of making HTTP request when
                        when using `elex [gov/house/senate]-trends`
  --format-json         Pretty print JSON when using `-o json`.
  -v, --version         show program's version number and exit
  --results-level RESULTS_LEVEL
                        Specify reporting level for results.
  --officeids OFFICEIDS Specify officeids to parse.
  --raceids RACEIDS     Specify raceids to parse.
  --set-zero-counts     Override results with zeros; omits the winner
                        indicator.Sets the vote, delegate, and reporting
                        precinct counts to zero.
  --national-only       Limit results to national-level results only.
  --local-only          Limit results to local-level results only.
  --with-timestamp      Append a `timestamp` column to each row of data output
                        with current system timestamp.
  --batch-name BATCH_NAME
                        Specify a value for a `batchname` column to append to
                        each row.
```

Command reference

class elex.cli.app.**ElexBaseController**(**args*, ***kw*)

races()

elex races <electiondate>

Returns race data for a given election date.

Command:

```
elex races 2016-03-26
```

Example output:

id	raceid	racetype	racetypeid	description	electiondate	initialization_data	is_ballot_measure	lastupdated	national	officeid	officename	party	seatname	seatnum	statename	statepostal	uncontested
2919	2919	Caucus	E		2016-03-26	True	False	2016-03-27T03:03:54Z	True	P	President	Dem			AK	False	False
12975	12975	Caucus	E		2016-03-26	True	False	2016-03-29T17:17:41Z	True	P	President	Dem			HI	False	False
...																	

reporting_units()

elex reporting-units <electiondate>

Returns reporting unit data for a given election date.

Reporting units represent geographic aggregation of voting data at the national, state, county, and district level.

Command:

```
elex reporting-units 2016-03-26
```

candidate_reporting_units()
`elex candidate-reporting-units <electiondate>`

Returns candidate reporting unit data for a given election date.

A candidate reporting unit is a container for the results of a voting in a specific reporting unit. This command is a close cousin of *elex results <electiondate>*.

This command does not return results.

Command:

```
elex candidate-reporting-units 2016-03-26
```

Notes:

This command can be used to quickly create schemas.

```
pip install csvkit
elex candidate-reporting-units 03-26-16 | csvsql -i mysql
```

Will output:

```
CREATE TABLE stdin (
    id VARCHAR(23) NOT NULL,
    raceid INTEGER NOT NULL,
    racetype VARCHAR(6) NOT NULL,
    racetypeid VARCHAR(1) NOT NULL,
    ...
);
```

candidates()
`elex candidates <electiondate>`

Returns candidate data for a given election date.

Command:

```
elex candidates 2016-03-26
```

Example output:

id	candidateid	ballotorder	first	last	party	polid	polnum
polid-1445	6527	2	Bernie	Sanders	Dem	1445	4262
polid-1746	6526	1	Hillary	Clinton	Dem	1746	4261
...							

ballot_measures()
`elex ballot-measures <electiondate>`

Returns ballot measure data for a given election date.

Command:

```
elex ballot-measures 2016-03-15
```

Example output:

id	candidateid	ballotorder	description	electiondate	last	polid	polnum	seatname
2016-03-15-43697	43697	1		2016-03-15	For		37229	Public Improvement Bonds
2016-03-15-43698	43698	2		2016-03-15	Against		37230	Public Improvement Bonds
...								

results()

```
elex results <electiondate>
```

Returns result data.

Each row in the output represents a fully flattened and denormalized version of a result for specific candidate in a specific race.

Command:

```
elex results 2016-03-01
```

Example output:

id	uncontested	raceid	racetype	racetypeid	ballotorder	candidateid	description	electiondate	electtype	statepostal	statename	test	winner	initialization_data	lastupdated	level	officeid	officename	party	polid	polnum	precinctsreporting	precinctsreportingpct	precincttotal	reportingunitid	reportingunitname	runoff	seatname	statefips	statepostal	test	votecount	votepct
302810-polid-6181-state-1	2	2	Democratic	D	2	6528	0	2016-03-01	Ted Cruz	False	False	False	True	2016-03-02T17:05:46Z	True	Primary	6	State	Republican	GOP	18425	72	1.072	state-1	False	Alaska	AK	False	37	02	False	36566	36.566
302810-polid-8639-state-1	5	2	Democratic	D	5	6548	0	2016-03-01	Donald	False	False	False	True	2016-03-02T17:05:46Z	True	Primary	6	State	Republican	GOP	8639	72	1.072	state-1	False	Alaska	AK	False	37	02	False	36475	36.475
...																																	

elections()

```
elex elections
```

Returns all elections known to the API.

Command:

```
elex elections
```

Example output:

2016-02-09	2016-02-09	True	False
2016-02-16	2016-02-16	True	False
...			

delegates()

`elex delegates`

Returns delegate report data.

Command:

```
elex delegates
```

Example output:

level	party_total	superdelegates_count	last	state	candidateid	party_needed	party	delegates_count	id	d1	d7	d30
state	2472	0	Bush	MN	1239	1237	GOP	0	MN-1239	0	0	0
state	2472	0	Bush	OR	1239	1237	GOP	0	OR-1239	0	0	0

governor_trends()

`elex governor-trends`

Governor balance of power/trend report.

Command:

```
elex governor-trends
```

Example output:

party	office	won	leading	holdovers	winning_trend	current	insufficient_vote	net_winners	net_leaders
Dem	Governor	7	7	12	19	20	0	-1	0

house_trends()

`elex house-trends`

House balance of power/trend report.

Command:

```
elex house-trends
```

Example output:

party	office	won	leading	holdovers	winning_trend	current	insufficient_vote	net_winners	net_leaders
Dem	U.S. House	201	201	0	201	193	0	+8	0

senate_trends()
elex senate-trends

Senate balance of power/trend report.

Command:

```
elex senate-trends
```

Example output:

party	office	won	leading	holdovers	winning_trend	current	insufficient_vote	net_winners	net_leaders
Dem	U.S. Senate	23	23	30	53	51	0	+2	0

next_election()
elex next-election <date-after>

Returns data about the next election with an optional date to start searching.

Command:

```
elex next-election
```

Example output:

id	electiondate	liveresults	testresults
2016-04-19	2016-04-19	False	True

You can also specify the date to find the next election after, e.g.:

```
elex next-election 2016-04-15
```

This will find the first election after April 15, 2016.

clear_cache()
elex clear-cache

Returns data about the next election with an optional date to start searching.

Command:

```
elex clear-cache
```

If no cache entries exist, elex will close with exit code 65.

2.5.4 Output and errors

Output handling

In the command line interface, all data is written to stdout. All messages are written to stderr.

```
# Direct data to a file and print messages to console
elex results 2016-02-01 > data.csv
```

(continues on next page)

(continued from previous page)

```
# Direct messages to a file and print data to console
elex results 2016-02-01 2> messages.csv

# Direct messages and data to individual files
elex results 2016-02-01 > data.csv 2> elex-log.txt
```

URLs (which typically contain the API key as a parameter), are only output when the `--debug` flag is specified.

Exit codes

If the `elex` command is successful, it closes with exit code 0.

In the command line interface, common errors are caught, logged, and the `elex` command exits with exit code 1.

Unknown / unexpected errors will continue to raise the normal Python exceptions with a full stacktrace. When this happens, the `elex` command exits with exit code 1.

Important note about future compatibility: Elex v2.1 will integrate a results caching mechanism. When results are returned from the cache and not from the API, the `elex` command will exit with exit code 64. To ensure future compatibility, **only check for exit code 1 when trapping errors** from your scripts.

Common errors

APAPIKeyError

```
2016-04-13 10:18:03,298 (ERROR) elex (v2.0.0) : APAPIKeyError: AP_API_KEY environment␣
↪variable is not set.
```

This means the AP API key is not set. Set the `AP_API_KEY` environment variable.

If using Elex as a Python library, you will need to pass `api_key` to the constructor, e.g.:

```
election = Election(api_key='<APIKEY>', ...)
```

The API key is not required when calling Elex with the `--data-file` flag.

ConnectionError

```
2016-04-12 10:47:59,928 (ERROR) elex (v2.0.0) : Connection error (<requests.packages.
↪urllib3.connection.HTTPConnection object at 0x108525588>: Failed to establish a new␣
↪connection: [Errno 8] nodename nor servname provided, or not known)
```

This happens when the elex client cannot connect to the API. Make sure the `AP_API_BASE_URL` environment variable is correct and that you have network connectivity.

HTTP Error 401 - Forbidden

```
2016-04-12 14:37:37,470 (ERROR) elex (v2.0.0) : HTTP Error 401 - Forbidden (Invalid␣
↪API key.)
```

These errors represent an authentication error. Typically, this is a problem with your AP API key. Make sure the `AP_API_KEY` environment variable is set correctly. If the problem persists, contact AP customer support.

HTTP Error 403 - Over Quota Limit

```
2016-04-12 10:24:01,904 (ERROR) elex (v2.0.0) : HTTP Error 403 - Over quota limit.
```

This means it is time to cool it and make less requests. Most AP clients have a quota of 10 requests a second.

HTTP Error 404 - Not found

```
2016-04-12 14:19:41,279 (ERROR) elex (v2.0.0) : HTTP Error 404 - Not Found.
```

This means the network connection was fine but the endpoint URL does not exist. Check `AP_API_BASE_URL` to make sure the URL is correct.

2.5.5 Configuration

The following environment variables may be set:

```
export API_VERSION='v2'
export BASE_URL='http://api.ap.org/v2'
export AP_API_KEY='<<YOURAPIKEY>>'
export ELEX_RECORDING='flat'
export ELEX_RECORDING_DIR='/tmp/elex-recording'
export ELEX_CACHE_DIRECTORY='/tmp/elex-cache'
```

API_VERSION

The AP API version. You should never need to change this.

BASE_URL

Use a different base url for API requests. Helpful if running a mirror or archive of raw AP data like Elex Deja Vu.

AP_API_KEY

Your API key. Must be set.

ELEX_CACHE_DIRECTORY

Path to the Elex cache directory. If not set, defaults to `<tempdir>/elex-cache` where `<tempdir>` is whatever Python's `tempfile.gettempdir()` returns.

ELEX_RECORDING, ELEX_RECORDING_DIR

Configure full data recording. See *Recording results*.

2.5.6 Recording results

Flat files

Will record timestamped and namespaced files to the `ELEX_RECORDING_DIR` before parsing.

```
export ELEX_RECORDING=flat
export ELEX_RECORDING_DIR=/tmp
```

MongoDB

Will record a timestamped record to MongoDB, connecting via `ELEX_RECORDING_MONGO_URL` and writing to the `ELEX_RECORDING_MONGO_DB` database.

```
export ELEX_RECORDING=mongodb
export ELEX_RECORDING_MONGO_URL=mongodb://localhost:27017/  # Or your own connection␣
↪string.
export ELEX_RECORDING_MONGO_DB=ap_elections_loader
```

2.5.7 Caching

Elex uses a simple file-based caching system based using CacheControl.

Each request to the AP Election API is cached. Each subsequent API request sends the etag. If the API returns a 304 not modified response, the cached version of the request is used.

Exit codes

If the underlying API call is returned from the cache, Elex exits with exit code 64.

For example, the first time you run an Elex results command, the exit code will be `0`.

```
elex results '02-01-2016'
echo $?
0
```

The next time you run the command, the exit code will be `64`.

```
elex results '02-01-2016'
echo $?
64
```

Clearing the cache

To clear the cache, run:

```
elex clear-cache
```

If the cache is empty, the command will return with exit code `65`. This is unlikely to be helpful to end users, but helps with automated testing.

Configuring the cache

To set the cache directory, set the `ELEX_CACHE_DIRECTORY` environment variable.

If `ELEX_CACHE_DIRECTORY` is not set, the default temp directory as determined by Python's tempfile module will be used.

2.5.8 Recipes

Useful Elex patterns. *Contribute your own*.

All examples specify a data file instead of a live or test election date so that all examples can be followed even if you don't have an AP API key. For real election data, replace `-d FILENAME` in these examples with an election date.

Get results at a specific level

Get only state level results:

```
elex results --results-level state -d "${VIRTUAL_ENV}/src/elex/tests/data/20160301_
↪super_tuesday.json"
```

id	raceid	racetype	racetypeid	ballotorder	candidateid	description	delegatecount	electiondate	fipscode	first	incumbent	initialization_data	is_ballot_measure	last	lastupdated	level	officeid	officename	of-party	polid	polnum	precinctsreporting	precinctsreportingpct	precinctstotal	reportingunitid	reportingunitname	runoff	seatname	statename	statepostal	test	uncontested	votecount	votepct	winner
24524747	47R	Republican	2	333600	polid-8639-state-1			2016-03-01		Donald	False		False	Trump	2016-03-02T20:35:54Z	state	P	President	GOP	8639	2	2	100.0	2	49	Massachusetts	False		MA	False	False	1249056	49.3	True
24524747	47R	Republican	13	333660	polid-36679-state-1			2016-03-01		John	False		False	Kasich	2016-03-02T20:35:54Z	state	P	President	GOP	36679	2	2	100.0	2	49	Massachusetts	False		MA	False	False	378809	14.9	False
...																																		

To cut down on load and bandwidth use and speed up loading when you are repeatedly loading results, specify only the level(s) you need to display.

Add timestamp or batch name column to any data command

You can add a timestamp column to track results (or any other data output by elex) with the `--with-timestamp` flag).

```
elex elections --with-timestamp -d "${VIRTUAL_ENV}/elex-dev/src/elex/tests/data/
↪00000000_elections.json"
```

id	electiondate	liveresults	testresults	timestamp
2012-03-13	2012-03-13	True	False	1460438301
2012-11-06	2012-11-06	True	False	1460438301
...				

If you prefer, you can set a batch name. This is useful when executing multiple commands that need a single grouping column.

```
elex elections --batch-name batch-031 -d "${VIRTUAL_ENV}/elex-dev/src/elex/tests/data/
↪00000000_elections.json"
```

id	electiondate	liveresults	testresults	timestamp
2016-02-23	2016-02-23	True	False	batch-031
2016-02-27	2016-02-27	True	False	batch-031
...				

Get local election results

Get only local races:

```
elex races 03-15-16 --local-only -d "${VIRTUAL_ENV}/src/elex/tests/data/20160301_
↪super_tuesday.json"
```

id	raceid	race-type	race-typeid	de-scription	elec-tion-date	ini-tial-iza-tion_data	is_ballot_measure	last-up-dated	na-tional	of-ficeid	of-fice-name	party	seat-name	seat-num	state-name	statepostal	total	un-contested
1489714897		Pri-mary	R		2016-03-15	True	False	2016-03-18T12:29:42Z	True	0	State's At-tor-ney	GOP	Cook County			IL	False	True
1532915329		Pri-mary	D		2016-03-15	True	False	2016-03-18T12:29:42Z	True	0	Recorder of Deeds	Dem	Cook County			IL	False	True
...																		

Get only local results:

```
elex results --local-only
```

Get AP zero count data

AP's set zero count parameter is a special server feature that only makes sense to query live.

```
elex results 03-15-16 --set-zero-counts
```

id	raceid	racetype	racetypeid	ballotorder	candidateid	description	delegatecount	electiondate	fipscode	first	incumbent	initialization_data	is_ballot_measure	last	lastupdated	level	national	officeid	officename	party	polid	polnum	precinctsreporting	precinctsreportingpct	precinctstotal	reportingunitid	reportingunitname	runoff	seatname	seatnum	statepostal	statename	test	uncontested	votecount	winner
10667 polid-8639-state-1	6673	Primary	R	13	204280		2016-03-15	Donald		False	False	False	True	Trump	2016-03-16T21:05:09Z	state	True	P	President	GOP	8639	45	740	0.058	10	state-1	False		Florida	FL	False	False	0.0	False
...																																				

Auto-generate SQL schemas with csvkit

Install csvkit, a handy tool for working with CSVs.

Now build the results schema by using the `candidate-reporting-units` command.

```
elex candidate-reporting-units -d "${VIRTUAL_ENV}/src/elex/tests/data/20160301_super_
↪tuesday.json" | csvsql --tables results -i sqlite
```

```
2016-04-14 00:51:07,675 (INFO) elex (v2.0.0) : Getting candidate reporting units for␣
↪election 2016-03-26
CREATE TABLE results (
  id VARCHAR(23) NOT NULL,
  raceid INTEGER NOT NULL,
  racetype VARCHAR(6) NOT NULL,
  racetypeid VARCHAR(1) NOT NULL,
  ballotorder INTEGER NOT NULL,
  candidateid INTEGER NOT NULL,
  description VARCHAR(32),
  delegatecount INTEGER NOT NULL,
  electiondate DATE NOT NULL,
  fipscode VARCHAR(32),
  first VARCHAR(7),
  incumbent BOOLEAN NOT NULL,
  initialization_data BOOLEAN NOT NULL,
  is_ballot_measure BOOLEAN NOT NULL,
  last VARCHAR(12) NOT NULL,
  lastupdated DATETIME NOT NULL,
  level VARCHAR(32),
  national BOOLEAN NOT NULL,
  officeid VARCHAR(1) NOT NULL,
  officename VARCHAR(9) NOT NULL,
  party VARCHAR(3) NOT NULL,
  polid INTEGER NOT NULL,
  polnum INTEGER NOT NULL,
  precinctsreporting INTEGER NOT NULL,
  precinctsreportingpct FLOAT NOT NULL,
  precinctstotal INTEGER NOT NULL,
  reportingunitid VARCHAR(32),
  reportingunitname VARCHAR(32),
  runoff BOOLEAN NOT NULL,
  seatname VARCHAR(32),
  seatnum VARCHAR(32),
```

(continues on next page)

(continued from previous page)

```
  statename VARCHAR(10) NOT NULL,
  statepostal VARCHAR(2) NOT NULL,
  test BOOLEAN NOT NULL,
  uncontested BOOLEAN NOT NULL,
  votecount INTEGER NOT NULL,
  votepct FLOAT NOT NULL,
  winner BOOLEAN NOT NULL,
  CHECK (incumbent IN (0, 1)),
  CHECK (initialization_data IN (0, 1)),
  CHECK (is_ballot_measure IN (0, 1)),
  CHECK (national IN (0, 1)),
  CHECK (runoff IN (0, 1)),
  CHECK (test IN (0, 1)),
  CHECK (uncontested IN (0, 1)),
  CHECK (winner IN (0, 1))
);
```

Insert results with csvkit + sqlite

This is not a wildly efficient way to get results into a database, but it is lightweight.

```
elex candidate-reporting-units -d "${VIRTUAL_ENV}/src/elex/tests/data/20160301_super_
↪tuesday.json" | csvsql --tables results --db sqlite:///db.sqlite --insert
```

Filter with jq and upload to S3

This recipe uses the jq json filtering tool to create a national results json data file with a limited set of data fields and the AWS cli tools to upload the filtered json to S3.

Requirements:

- Amazon web services account
- jq
- AWS cli tools

```
#!/bin/bash

# S3 url: MUST be set to your bucket and path.
ELEX_S3_URL='mybucket.tld/output/path.json'

# Get results and upload to S3
elex results 2012-11-06 --results-level state -o json \
| jq -c '[
            .[] |
            select(.level == "state" ) |
            select(.officename == "President") |
            {
              officename: .officename,
              statepostal: .statepostal,
              first: .first,
              last: .last,
              party: .party,
              votecount: .votecount,
```

(continues on next page)

(continued from previous page)

```
            votepct: .votepct,
            winner: .winner,
            level: .level
          }
        ]' \
| gzip -vc \
| aws s3 cp - s3://$ELEX_S3_URL \
    --acl public-read \
    --content-type=application/json \
    --content-encoding gzip

# Check response headers
curl -I $ELEX_S3_URL

# Get first entry of uploaded json
curl -s --compressed $ELEX_S3_URL | jq '[.[]][0]'
```

`ELEX_S3_URL` **must** be set to your s3 bucket and path.

Steps:

- Get election results in json format with `elex`
- Pipe results to `jq` for filtering
- Pipe filtered results to `gzip` to compress
- Pipe gzipped results to `aws s3 cp` to send to S3.

Inspect with an ORM using Flask and Peewee

This recipe uses the Flask web framework and the Peewee Python ORM to model, query and update data that `elex` provides.

Requirements:

- Elex loader, an NYT project that calls `elex` to load data into a Postgres database with CSV and the Postgres `COPY` command.
- Elex admin, an NYT project that is a simple, web-based admin for creating and editing data to override AP election results, including candidate names, race descriptions, and race calls.

Steps:

- Install `elex-loader` using the loader instructions.
- Install `elex-admin` using the admin instructions.

Extra steps:

- Use the `models.py` that come with `elex-admin` to query data.

2.5.9 Python library reference

Elex can be used as a Python library that provides wrapper objects around AP election data.

Data models / API wrapper (elex.api)

elex.api

class elex.api.**APElection**
Base class for most objects.

Includes handy methods for transformation of data and AP connections

set_state_fields_from_reportingunits()
Set state fields.

set_reportingunits()
Set reporting units.

If this race has reportingunits, serialize them into objects.

set_polid()
Set politication id.

If *polid* is zero, set to *None*.

set_reportingunitids()
Set reporting unit ID.

Per Tracy / AP developers, if the level is "state", the reportingunitid is always 1.

set_candidates()
Set candidates.

If this thing (race, reportingunit) has candidates, serialize them into objects.

serialize()
Serialize the object. Should be implemented in all classes that inherit from *APElection*.

Should return an OrderedDict.

class elex.api.**BallotMeasure**(***kwargs*)
Canonical representation of a ballot measure.

Ballot measures are similar to *Candidate* objects, but represent a position on a ballot such as "In favor of" or "Against" for ballot measures such as a referendum.

serialize()
Implements *APElection.serialize()*.

set_unique_id()
Generate and set unique id.

Candidate IDs are not globally unique. AP National Politian IDs (NPIDs or polid) are unique, but only national-level candidates have them; everyone else gets '0'. The unique key, then, is the NAME of the ID we're using and then the ID itself. Verified this is globally unique with Tracy.

set_id_field()
Set id to <*unique_id*>.

class elex.api.**Candidate**(***kwargs*)
Canonical representation of a candidate. Should be globally unique for this election, across races.

serialize()
Implements *APElection.serialize()*.

set_unique_id()
Generate and set unique id.

Candidate IDs are not globally unique. AP National Politian IDs (NPIDs or polid) are unique, but only national-level candidates have them; everyone else gets '0'. The unique key, then, is the NAME of the ID we're using and then the ID itself. Verified this is globally unique with Tracy.

set_id_field()
Set id to <*unique_id*>.

class elex.api.**CandidateDelegateReport**(***kwargs*)
'level': 'state', 'party_total': 4762, 'superdelegates_count': 0, 'last': u'Steinberg', 'state': u'SD', 'candidateid': u'11291', 'party_need': 2382, 'party': u'Dem', 'delegates_count': 0, 'id': u'SD-11291', 'd1': -1, 'd7': 8, 'd30': 10

serialize()
Implements *APElection.serialize()*.

class elex.api.**CandidateReportingUnit**(***kwargs*)
Canonical reporesentation of an AP candidate. Note: A candidate can be a person OR a ballot measure.

set_id_field()
Set id to <*raceid*>-<*uniqueid*>-<*reportingunitid*>.

set_unique_id()
Generate and set unique id.

Candidate IDs are not globally unique. AP National Politian IDs (NPIDs or polid) are unique, but only national-level candidates have them; everyone else gets '0'. The unique key, then, is the NAME of the ID we're using and then the ID itself. Verified this is globally unique with Tracy Lewis.

serialize()
Implements *APElection.serialize()*.

class elex.api.**DelegateReport**(***kwargs*)
Base class for a single load of AP delegate counts. d = DelegateReport() [z.__dict__ for z in d.candidates]

output_candidates()
Transforms our multi-layered dict of candidates / states into a single list of candidates at each reporting level.

parse_sum()
Parses the delsum JSON produced by the AP.

parse_super()
Parses the delsuper JSON produced by the AP.

load_raw_data(*delsuper_datafile*, *delsum_datafile*)
Gets underlying data lists we need for parsing.

get_ap_file(*path*, *key*)
Get raw data file.

get_ap_report(*key*, *params={}*)
Given a report number and a key for indexing, returns a list of delegate counts by party. Makes a request from the AP using requests. Formats that request with env vars.

get_report_id(*reports*, *key*)
Takes a delSuper or delSum as the argument and returns organization-specific report ID.

class elex.api.**Election**(***kwargs*)
Canonical representation of an election on a single date.

set_id_field()
Set id to <*electiondate*>.

get(*path*, ***params*)
Farms out request to api_request. Could possibly handle choosing which parser backend to use – API-only right now. Also the entry point for recording, which is set via environment variable.

Parameters

- **path** – API url path.
- ****params** – A dict of optional parameters to be included in API request.

get_uniques(*candidate_reporting_units*)
Parses out unique candidates and ballot measures from a list of CandidateReportingUnit objects.

get_raw_races(***params*)
Convenience method for fetching races by election date. Accepts an AP formatting date string, e.g., YYYY-MM-DD. Accepts any number of URL params as kwargs.

If datafile passed to constructor, the file will be used instead of making an HTTP request.

Parameters ****params** – A dict of additional parameters to pass to API. Ignored if *datafile* was passed to the constructor.

get_race_objects(*parsed_json*)
Get parsed race objects.

Parameters **parsed_json** – Dict of parsed AP election JSON.

get_units(*race_objs*)
Parses out races, reporting_units, and candidate_reporting_units in a single loop over the race objects.

Parameters **race_objs** – A list of top-level Race objects.

serialize()
Implements *APElection.serialize()*.

races
Return list of race objects.

reporting_units
Return list of reporting unit objects.

candidate_reporting_units
Return list of candidate reporting unit objects.

results
Return list of candidate reporting unit objects with results.

candidates
Return list of candidate objects with results.

ballot_measures
Return list of ballot measure objects with results.

class elex.api.**Elections**
Holds a collection of election objects

get_elections(*datafile=None*)
Get election data from API or cached file.

Parameters **datafile** – If datafile is specified, use instead of making an API call.

get_next_election(*datafile=None*, *electiondate=None*)
Get next election. By default, will be relative to the current date.

Parameters

- **datafile** – If datafile is specified, use instead of making an API call.
- **electiondate** – If electiondate is specified, gets the next election after the specified date.

class elex.api.**Race**(***kwargs*)
Canonical representation of a single race, which is a seat in a political geography within a certain election.

set_id_field()
Set id to <*raceid*>.

serialize()
Implements *APElection.serialize()*.

class elex.api.**ReportingUnit**(***kwargs*)
Canonical representation of a single level of reporting.

set_level()
New England states report at the township level. Every other state reports at the county level. So, change the level from 'subunit' to the actual level name, either 'state' or 'township'.

set_id_field()
Set id to <*reportingunitid*>.

set_votecount()
Set vote count.

set_candidate_votepct()
Set vote percentage for each candidate.

serialize()
Implements *APElection.serialize()*.

class elex.api.**BaseTrendReport**(*trend_file=None*, *testresults=False*)
A base class for retrieving trend reports from the AP API.

load_raw_data(*office_code*, *trend_file=None*)
Gets underlying data lists we need for parsing.

get_ap_file(*path*)
Get raw data file.

get_ap_report(*key*, *params={}*)
Given a report number and a key for indexing, returns a list of delegate counts by party. Makes a request from the AP using requests. Formats that request with env vars.

get_report_id(*reports*, *key*)
Takes a delSuper or delSum as the argument and returns organization-specific report ID.

output_parties()
Parse the raw data on political parties returned by the API, converts them into objects and assigns them to the object's `parties` attribute.

class elex.api.**USGovernorTrendReport**(*trend_file=None*, *testresults=False*)
A trend report on U.S. governors.

class elex.api.**USSenateTrendReport**(*trend_file=None*, *testresults=False*)
A trend report on the U.S. Senate.

class elex.api.**USHouseTrendReport**(*trend_file=None*, *testresults=False*)
A trend report on U.S. House.

elex.api.utils

Utility functions to record raw election results and handle low-level HTTP interaction with the Associated Press Election API.

class elex.api.utils.**UnicodeMixin**
Python 2 + 3 compatibility for __unicode__

elex.api.utils.**write_recording**(*payload*)
Record a timestamped version of an Associated Press Elections API data download.

Presumes JSON at the moment. Would have to refactor if using XML or FTP. FACTOR FOR USE; REFACTOR FOR REUSE.

Parameters **payload** – JSON payload from Associated Press Elections API.

elex.api.utils.**api_request**(*path*, ***params*)
Function wrapping Python-requests for making a request to the AP's elections API.

A properly formatted request: * Modifies the BASE_URL with a path. * Contains an API_KEY. * Returns a response object.

Parameters ****params** – Extra parameters to pass to *requests*. For example, *apiKey="<YOUR API KEY>*, your AP API key, or *national=True*, for national-only results.

elex.api.utils.**get_reports**(*params={}*)
Get data from *reports* endpoints.

elex.api.maps

Defines `FIPS_TO_STATE`, `STATE_ABBR`, `OFFICE_NAMES` and `PARTY_NAMES` look-up constants.

Command line interface (elex.cli)

elex.cli.app

class elex.cli.app.**ElexBaseController**(**args*, ***kw*)

class Meta

label = 'base'

description = 'Get and process AP elections data'

arguments = [(['date'], {'nargs': '*', 'help': 'Election date (e.g. "2015-11-03";

default()

races()
`elex races <electiondate>`

Returns race data for a given election date.

Command:

```
elex races 2016-03-26
```

Example output:

id	raceid	race-type	race-typeid	de-scrip-tion	elec-tion-date	ini-tial-iza-tion_data	is_ballot_measure	last-up-dated	juris-dic-tional	of-fice-id	of-fice-name	party	seat-name	seat-num	state-name	state	postal	un-con-tested
2919	2919	Cau-cus	E		2016-03-26	True	False	2016-03-27T03:03:54Z	True	P	Pres-i-dent	Dem				AK	False	False
12975	12975	Cau-cus	E		2016-03-26	True	False	2016-03-29T17:17:41Z	True	P	Pres-i-dent	Dem				HI	False	False
...																		

reporting_units()

```
elex reporting-units <electiondate>
```

Returns reporting unit data for a given election date.

Reporting units represent geographic aggregation of voting data at the national, state, county, and district level.

Command:

```
elex reporting-units 2016-03-26
```

candidate_reporting_units()

```
elex candidate-reporting-units <electiondate>
```

Returns candidate reporting unit data for a given election date.

A candidate reporting unit is a container for the results of a voting in a specific reporting unit. This command is a close cousin of *elex results* <*electiondate*>.

This command does not return results.

Command:

```
elex candidate-reporting-units 2016-03-26
```

Notes:

This command can be used to quickly create schemas.

```
pip install csvkit
elex candidate-reporting-units 03-26-16 | csvsql -i mysql
```

Will output:

```
CREATE TABLE stdin (
    id VARCHAR(23) NOT NULL,
    raceid INTEGER NOT NULL,
    racetype VARCHAR(6) NOT NULL,
    racetypeid VARCHAR(1) NOT NULL,
    ...
);
```

candidates()

```
elex candidates <electiondate>
```

Returns candidate data for a given election date.

Command:

```
elex candidates 2016-03-26
```

Example output:

id	candidateid	ballotorder	first	last	party	polid	polnum
polid-1445	6527	2	Bernie	Sanders	Dem	1445	4262
polid-1746	6526	1	Hillary	Clinton	Dem	1746	4261
...							

ballot_measures()

```
elex ballot-measures <electiondate>
```

Returns ballot measure data for a given election date.

Command:

```
elex ballot-measures 2016-03-15
```

Example output:

id	candidateid	ballotorder	description	electiondate	last	polid	polnum	seatname
2016-03-15-43697	43697	1		2016-03-15	For		37229	Public Improvement Bonds
2016-03-15-43698	43698	2		2016-03-15	Against		37230	Public Improvement Bonds
...								

results()

```
elex results <electiondate>
```

Returns result data.

Each row in the output represents a fully flattened and denormalized version of a result for specific candidate in a specific race.

Command:

```
elex results 2016-03-01
```

Example output:

| id | unique_id | race_type | race_id | ballot_order | is_ballot_measure | election_date | delegate_count | is_primary | is_runoff | is_special_election | office_id | office_name | state_postal | state_fips | state_name | last_updated | has_electoral_votes | num_seats | party_id | party_name | polid | polnum | precinct_reporting | precincts_total | precincts_reporting_pct | test | uncontested | national | num_winners | seat_name | seat_num | race_description | race_name | race_party | race_state_id | race_state_name | race_state_postal | race_state_fips | race_office_id | race_office_name | race_office_description | race_seat_name | race_seat_num | race_is_primary | race_is_runoff | race_is_special_election | race_primary_party | race_primary_party_name | race_num_seats | race_num_winners | race_precincts_reporting | race_precincts_total | race_precincts_reporting_pct | race_total_votes | race_total_votes_dem | race_total_votes_gop | race_total_votes_lib | race_total_votes_grn | race_total_votes_ind | race_total_votes_oth | race_test | race_uncontested | race_national | candidate_id | candidate_ballot_order | candidate_first_name | candidate_last_name | candidate_party_id | candidate_party_name | candidate_polid | candidate_polnum | candidate_incumbent | candidate_delegate_count | candidate_uncontested | candidate_test | candidate_national | candidate_winner | candidate_votes | candidate_electoral_votes | candidate_votes_dem | candidate_votes_gop | candidate_votes_lib | candidate_votes_grn | candidate_votes_ind | candidate_votes_oth | candidate_last_updated | candidate_ballot_measure_text | candidate_ballot_measure_yes_votes | candidate_ballot_measure_no_votes | candidate_ballot_measure_pct_yes_votes | candidate_ballot_measure_pct_no_votes |
|---|---|---

```
elex governor-trends
```

Example output:

party	office	won	leading	holdovers	winning_trend	current	insufficient_vote	net_winners	net_leaders
Dem	Governor	7	7	12	19	20	0	-1	0

house_trends()

```
elex house-trends
```

House balance of power/trend report.

Command:

```
elex house-trends
```

Example output:

party	office	won	leading	holdovers	winning_trend	current	insufficient_vote	net_winners	net_leaders
Dem	U.S. House	201	201	0	201	193	0	+8	0

senate_trends()

```
elex senate-trends
```

Senate balance of power/trend report.

Command:

```
elex senate-trends
```

Example output:

party	office	won	leading	holdovers	winning_trend	current	insufficient_vote	net_winners	net_leaders
Dem	U.S. Senate	23	23	30	53	51	0	+2	0

next_election()

```
elex next-election <date-after>
```

Returns data about the next election with an optional date to start searching.

Command:

```
elex next-election
```

Example output:

id	electiondate	liveresults	testresults
2016-04-19	2016-04-19	False	True

You can also specify the date to find the next election after, e.g.:

```
elex next-election 2016-04-15
```

This will find the first election after April 15, 2016.

clear_cache()
elex clear-cache

Returns data about the next election with an optional date to start searching.

Command:

```
elex clear-cache
```

If no cache entries exist, elex will close with exit code 65.

class elex.cli.app.**ElexApp**(*label=None*, ***kw*)

class Meta

label = 'elex'

base_controller
alias of *ElexBaseController*

exit_on_close = True

hooks = [('post_setup', <function cachecontrol_logging_hook>), ('post_argument_pars

extensions = ['elex.cli.ext_csv', 'elex.cli.ext_json']

output_handler = 'csv'

handler_override_options = {'output': (['-o'], {'help': 'output format (default:

log_handler = <cement.ext.ext_logging.LoggingLogHandler object>

elex.cli.app.**main**()

elex.cli.decorators

elex.cli.decorators.**require_date_argument**(*fn*)
Decorator that checks for date argument.

elex.cli.decorators.**require_ap_api_key**(*fn*)
Decorator that checks for Associated Press API key or data-file argument.

elex.cli.ext_csv

class elex.cli.ext_csv.**CSVOutputHandler**(**args*, ***kw*)
A custom CSV output handler

class Meta

label = 'csv'

overridable = True

render(*data*, *template=None*)

elex.cli.ext_csv.**load**(*app*)

elex.cli.ext_json

class elex.cli.ext_json.**ElexJSONOutputHandler**(**args*, ***kw*)
A custom JSON output handler

class Meta

label = 'json'

overridable = True

render(*data*, *template=None*)

elex.cli.ext_json.**load**(*app*)

elex.cli.hooks

elex.cli.hooks.**add_election_hook**(*app*)
Cache election API object reference after parsing args.

elex.cli.hooks.**cachecontrol_logging_hook**(*app*)
Reroute cachecontrol logger to use cement log handlers.

elex.cli.utils

elex.cli.utils.**parse_date**(*datestring*)
Parse many date formats into an AP friendly format.

Exceptions (elex.exceptions)

Elex exceptions

exception elex.exceptions.**APAPIKeyException**
Raise this exception when an AP API key is not set.

message = 'AP API key is not set.'

Official Associated Press API

The official documentation of the Associated Press election data API that this library draws from.

2.5.10 Contributing

We welcome contributions of all sizes. You got this!

Find a task

1. Check out the issue tracker and pick out a task or create a new issue
2. Leave a comment on the ticket so that others know you're working on it.

Install Elex development environment

1. Fork the project on Github.
2. Install a development version of the code with:

```
mkvirtualenv elex-dev
workon elex-dev
git clone git@github.com:<YOUR_GITHUB_USER>/elex.git .``
```

3. Install developer dependencies for tests and docs:

```
pip install -r requirements.txt
pip install -r requirements-dev.txt
```

Running tests

Edit or write the code or docs, taking care to include well=crafted docstrings and generally following the format of the existing code.

Write tests for any new features you add. Add to the tests in the `tests` directory or follow the format of files like `tests/test_election.py`.

Make sure all tests are passing in your environment by running the nose2 tests.

```
make test
```

If you have Python 2.7, 3.6, and pypy installed, run can run `tox` to test in multiple environments.

Writing docs

Write documentation by adding to one of the files in `docs` or adding your own.

To build a local preview, run:

```
make -C docs html
```

The documentation is built in `docs/_build/html`. Use Python's simple HTTP server to view it.

```
cd docs/_build/html
python -m http.server
```

Python 2.7 users should use `SimpleHTTPServer` instead of `http.server`.

Submitting code

Submit a pull request on Github.

Testing performance

To get detailed information about performance, run the tests with the –profile flag:

```
nose2 tests --profile
```

Testing API request limit

You can test the API request limit, but only by setting an environment variable. Use with extreme care.

```
AP_RUN_QUOTA_TEST=1 nose2 tests.test_ap_quota
```

Authors

elex is maintained by Jeremy Bowers <jeremy.bowers@nytimes.com> and David Eads <davideads@gmail.com>.

These individuals have contributed code, tests, documentation, and troubleshooting:

- Jeremy Bowers
- David Eads
- Livia Labate
- Wilson Andrews
- Eric Buth
- Juan Elosua
- Ben Welsh
- Tom Giratikanon
- Ryan Pitts
- Miles Watkins
- Vanessa Qian

2.5.11 Changelog

2.4.0 - October 23, 2016

- Add version number back to log output (#294)
- Restore exit code 64 for cached responses (#302) and document caching (#301)
- Fix csvkit compatibility (#206)
- Add trend reports (#299)

2.3.0 - October 5, 2016

- Fix `__str__` methods (#292)
- Fix bug related to parties in initialization data (#293, #286)
- Add `clear-cache` command to wipe http response cache (#289)

2.2.0 - September 24, 2016

- Add reponse caching (#121, #250)
- Enable gzip on http requests (#273)
- Updated pinned requirements (#279)

2.1.1 - September 12, 2016

Fixes a bug related to national / local flags on races. Running `--local-only` would show all races as `national=true` due to a Elex defaulting `national` to `true` but the AP drops the `national` flag in the API results when the URL specifies `national=false`.

2.1.0 - August 31, 2016

- Breaking change: Adds `electwon` and `electtotal` to `CandidateReportingUnit` and `electtotal` to `ReportingUnit` to represent total electoral votes and number of electoral votes won. As these fields were not in previous releases, we've bumped to 2.1.x to indicate a breaking change to the schema.

2.0.10 - 2.0.11 - August 25, 2016

- Fixes a bug that makes `reportingunitid` not-unique for national races, e.g., president. (#278)

2.0.9 - August 16, 2016

- A variety of transparent speedups. (#277)

2.0.8 - July 25, 2016

- Adds a `raceids` feature. `elex races 2016-03-15 --raceids 10675,14897` still downloads the full JSON file but only parses the races passed in the `raceids` argument. Particularly effective when used with the `local-only` flag to grab a subset of non-national races, e.g., every NY state race.

2.0.5 - 2.0.6 - June 6, 2016

- Fixes a small bug in the ME reporting for the upcoming 6-14 primary.

2.0.1 - 2.0.4 - April 26, 2016

- Fixes for AP API v2.1.
- Fix for missing Rhode Island mail-in ballot (#263).
- Fix for township to county rollups in New England (#264).
- Delegate report cache has been removed; the feature could have negative consequences and will be better addressed by a full caching system.

2.0.0 - April 14, 2016

Remove redundant data fields, introduce breaking data model fixes, organizational report ID caching, and command line cleanup.

The 2.x release is named for Ethel Payne, the "First Lady of the Black Press", whose natural curiosity led her to become a groundbreaking journalist.

- Precincts reporting percent now expressed in normal form (#204). Prior to the 2.0 release, precincts reporting percent was expressed as a number between 0 and 100 while vote percent while percent of votes received was expressed as a number between 0 and 1. Now **all percents in the data are expressed as a number between 0 and 1** and should be multiplied by 100 to display the human-readable percentage.
- Remove `unique_id` field (#256). The unique_id has been superseded by the ID field in all cases and was redundant. The 2.0 release removes this field, and all Elex users should adjust their data models and schemas accordingly.
- Race data now includes an `is_ballot_measure` column for consistency (#238).
- Cache delegate report IDs (#234). Getting delegate reports previously required three API calls which each counted against the API quota limit. Now, on first request, the report IDs are cached until the `elex clear-delegate-cache` command is run. With the introduction of "free" report access in AP API v2.1, getting delegate reports do not count at all against the request quota except the first `elex delegates` is run or after running `elex clear-delegate-cache`.
- Refactor error handling when interacting with the API (#239, #240, #249). All error handling logic has been moved to command line library and out of the Python API. All errors encountered when using Elex as a Python library are raised and must be handled by the developer. The command line library catches common/well-known errors and provides useful feedback.
- Add `--with-timestamp` and `--batch-name` flags to add a timestamp based or arbitrary grouping column to any results (#212).
- `elex next-election` now returns an error when there is no valid next election (#160).
- Election date is automatically determined when using the `--data-file` flag. This means no date argument is required when specifying a data file. (#161)
- Removed dependency on Clint output library (#63).
- Improve documentation (#251).
- Abandon previous caching and daemon efforts (#122, #137). Caching will be a feature of Elex 2.1.

Important note about exit codes:

Elex will be implementing a caching layer in version 2.1 that uses conditional GET requests to decide whether or not to get fresh data. The command line tool will return exit code 64 when getting data from the cache, the normal 0 exit code on a successful full request, and exit code 1 for all errors. If you have code that depends on reading the Elex exit code, ensure that you are checking for exit code 1 and 1 only when trapping for errors.

1.2.0 - Feb. 25, 2016

Many bugfixes and some new fields / id schemes that might break implementations that rely on stable field names / orders.

- Fixes an issue with requests defaulting to national-only (#229, #230).
- Solves an issue with 3/5 and 3/6 Maine results not including townships (#228).
- Supports a `set-zero-counts` argument to the CLI to return zeroed-out data (#227).

- Includes a `delegatecount` field on `CandidateReportingUnit` to store data from district-level results (#225).
- Supports a `results-level` argument to the CLI to return district-level data. (#223)
- Solves an issue with `reportingunitid` not being unique acrosss different result levels (#226).
- Adds an `electiondate` field on `BallotMeasure` to guarantee uniqueness (#210).
- Makes a composite id for `BallotMeasure` that includes `electiondate` (#210).

1.1.0 - Feb. 2, 2016

Documentation and dependency fixes.

- Elex can now be run in the same virtualenv as csvkit (#206).
- Links and copyright notice in documentation updated.
- Added section about virtualenvs to install guide, courtesy of Ryan Pitts.
- Add better tests for AP request quota (#203).

1.0.0 - Jan. 25, 2016

The 1.x release is named for Martha Ellis Gellhorn, one of the greatest war correspondents of the 20th century.

- Delegate counts (#138, #194). Delegate counts can be accessed with `elex delegates`.
- Rename `elex.api.api` to `elex.api.models` and allow model objects to be imported with statements like `from elex.api import Election` (#146). Python modules directly calling Elex will need to update their import statements accordingly.
- Fix duplicate IDs (#176).
- Handle incorrect null/none values in some cases (#173, #174, #175).
- Expand contributing / developer guide (#151).
- Add recipe for filtering with jq and uploading to s3 in a single command (#131).

0.2.0 - Dec. 24, 2015

- Tag git versions (#170).
- Fix elections command (#167).
- Use correct state code for county level results (#164).
- Use tox to test multiple Python versions (#153).
- Allow API url to be specified in environment variable (#144).
- Don't sort results for performance and stability (#136).
- Capture and log full API request URL in command line debugging mode (#134).
- Python 3 compatibility (#99).

0.1.2 - Dec. 21, 2015

- Fix missing vote percent in results (#152).

0.1.1 - Dec. 10, 2015

- Add Travis CI support (#101).
- Fix packaging.

0.1.0 - Dec. 10, 2015

First major release.

- Decided on *elex* for name (#59).
- Initial tests (#70, #107).
- First draft of docs (#18).
- Set up http://elex.readthedocs.org/ (#60).
- Handle New England states (townships and counties) (#123).
- Remove date parsing (#115) and dynamic field setter (#117) to improve performance.

0.0.0 - 0.0.42

Initial Python API and concept created by Jeremy Bowers; initial command line interface created by David Eads.

Python Module Index

e

elex.api, 24
elex.api.maps, 28
elex.api.utils, 28
elex.cli.decorators, 33
elex.cli.ext_csv, 33
elex.cli.ext_json, 34
elex.cli.hooks, 34
elex.cli.utils, 34
elex.exceptions, 34

Index

A

add_election_hook() (in module elex.cli.hooks), 34
APAPIKeyException, 34
APElection (class in elex.api), 24
api_request() (in module elex.api.utils), 28

B

ballot_measures (elex.api.Election attribute), 26
ballot_measures() (elex.cli.app.ElexBaseController method), 12
BallotMeasure (class in elex.api), 24
BaseTrendReport (class in elex.api), 27

C

cachecontrol_logging_hook() (in module elex.cli.hooks), 34
Candidate (class in elex.api), 24
candidate_reporting_units (elex.api.Election attribute), 26
candidate_reporting_units() (elex.cli.app.ElexBaseController method), 12
CandidateDelegateReport (class in elex.api), 25
CandidateReportingUnit (class in elex.api), 25
candidates (elex.api.Election attribute), 26
candidates() (elex.cli.app.ElexBaseController method), 12
clear_cache() (elex.cli.app.ElexBaseController method), 15
CSVOutputHandler (class in elex.cli.ext_csv), 33
CSVOutputHandler.Meta (class in elex.cli.ext_csv), 33

D

DelegateReport (class in elex.api), 25
delegates() (elex.cli.app.ElexBaseController method), 14

E

Election (class in elex.api), 25
Elections (class in elex.api), 26
elections() (elex.cli.app.ElexBaseController method), 13
elex.api (module), 24
elex.api.maps (module), 28
elex.api.utils (module), 28
elex.cli.decorators (module), 33
elex.cli.ext_csv (module), 33
elex.cli.ext_json (module), 34
elex.cli.hooks (module), 34
elex.cli.utils (module), 34
elex.exceptions (module), 34
ElexBaseController (class in elex.cli.app), 11
ElexJSONOutputHandler (class in elex.cli.ext_json), 34
ElexJSONOutputHandler.Meta (class in elex.cli.ext_json), 34

G

get() (elex.api.Election method), 26
get_ap_file() (elex.api.BaseTrendReport method), 27
get_ap_file() (elex.api.DelegateReport method), 25
get_ap_report() (elex.api.BaseTrendReport method), 27
get_ap_report() (elex.api.DelegateReport method), 25
get_elections() (elex.api.Elections method), 26
get_next_election() (elex.api.Elections method), 26
get_race_objects() (elex.api.Election method), 26
get_raw_races() (elex.api.Election method), 26
get_report_id() (elex.api.BaseTrendReport method), 27
get_report_id() (elex.api.DelegateReport method), 25
get_reports() (in module elex.api.utils), 28
get_uniques() (elex.api.Election method), 26
get_units() (elex.api.Election method), 26
governor_trends() (elex.cli.app.ElexBaseController method), 14

H

house_trends() (elex.cli.app.ElexBaseController method), 14

L

label (elex.cli.ext_csv.CSVOutputHandler.Meta attribute), 33

label (elex.cli.ext_json.ElexJSONOutputHandler.Meta attribute), 34
load() (in module elex.cli.ext_csv), 33
load() (in module elex.cli.ext_json), 34
load_raw_data() (elex.api.BaseTrendReport method), 27
load_raw_data() (elex.api.DelegateReport method), 25

M

message (elex.exceptions.APAPIKeyException attribute), 34

N

next_election() (elex.cli.app.ElexBaseController method), 15

O

output_candidates() (elex.api.DelegateReport method), 25
output_parties() (elex.api.BaseTrendReport method), 27
overridable (elex.cli.ext_csv.CSVOutputHandler.Meta attribute), 33
overridable (elex.cli.ext_json.ElexJSONOutputHandler.Meta attribute), 34

P

parse_date() (in module elex.cli.utils), 34
parse_sum() (elex.api.DelegateReport method), 25
parse_super() (elex.api.DelegateReport method), 25

R

Race (class in elex.api), 27
races (elex.api.Election attribute), 26
races() (elex.cli.app.ElexBaseController method), 11
render() (elex.cli.ext_csv.CSVOutputHandler method), 33
render() (elex.cli.ext_json.ElexJSONOutputHandler method), 34
reporting_units (elex.api.Election attribute), 26
reporting_units() (elex.cli.app.ElexBaseController method), 11
ReportingUnit (class in elex.api), 27
require_ap_api_key() (in module elex.cli.decorators), 33
require_date_argument() (in module elex.cli.decorators), 33
results (elex.api.Election attribute), 26
results() (elex.cli.app.ElexBaseController method), 13

S

senate_trends() (elex.cli.app.ElexBaseController method), 14
serialize() (elex.api.APElection method), 24
serialize() (elex.api.BallotMeasure method), 24
serialize() (elex.api.Candidate method), 24
serialize() (elex.api.CandidateDelegateReport method), 25
serialize() (elex.api.CandidateReportingUnit method), 25
serialize() (elex.api.Election method), 26
serialize() (elex.api.Race method), 27
serialize() (elex.api.ReportingUnit method), 27
set_candidate_votepct() (elex.api.ReportingUnit method), 27
set_candidates() (elex.api.APElection method), 24
set_id_field() (elex.api.BallotMeasure method), 24
set_id_field() (elex.api.Candidate method), 25
set_id_field() (elex.api.CandidateReportingUnit method), 25
set_id_field() (elex.api.Election method), 25
set_id_field() (elex.api.Race method), 27
set_id_field() (elex.api.ReportingUnit method), 27
set_level() (elex.api.ReportingUnit method), 27
set_polid() (elex.api.APElection method), 24
set_reportingunitids() (elex.api.APElection method), 24
set_reportingunits() (elex.api.APElection method), 24
set_state_fields_from_reportingunits() (elex.api.APElection method), 24
set_unique_id() (elex.api.BallotMeasure method), 24
set_unique_id() (elex.api.Candidate method), 24
set_unique_id() (elex.api.CandidateReportingUnit method), 25
set_votecount() (elex.api.ReportingUnit method), 27

U

UnicodeMixin (class in elex.api.utils), 28
USGovernorTrendReport (class in elex.api), 27
USHouseTrendReport (class in elex.api), 27
USSenateTrendReport (class in elex.api), 27

W

write_recording() (in module elex.api.utils), 28