BEL.bio API Documentation

Release 0.4.28

William Hayes

Oct 02, 2019

Contents:

The BEL.bio API provides a REST API for the BEL Language and BEL Content.

Functionality provided:

- BEL Namespace (Term) queries
- BEL language parsing and validation
- BEL Nanopub validation
- BEL Edge creation from BEL Nanopubs

API Documentation (also found at each API endpoint at /swaggerui)

CHAPTER 1

API Documentation

Demo API Documentation

BEL API documentation using OpenAPI (Swagger) 3+. The documentation is hosted inside the BEL API service.

Every BEL API endpoint comes with Swagger UI pointing to the OpenAPI/Swagger doc. Just got to the root url of the API and add '/swaggerui' - e.g. https://api.bel.bio/swaggerui as one example.

In the Swagger UI, you can use either of the Servers: the Demo server (https://api.bel.bio/swaggerui) or the API service local to the API endpoint demonstrated by the Server '/' which is the default or 'https://api.bel.bio'.

CHAPTER 2

Installation

The recommended way to use the BEL.bio API is to use the docker image. We publish the released BEL.bio API releases as docker images on Dockerhub at https://hub.docker.com/r/belbio/bel_api/.

2.1 Configuration

Configuration is described in the BEL Python package documentation

Configuration documentation

2.2 Docker Compose

```
# docker-compose build && docker-compose up -d
version: "3.2"

volumes:
  bel_arango_data:
    driver: local
  bel_elasticsearch_data:
    driver: local
  bel_kibana_data:
    driver: local

services:

  # BEL API - core requirement
  # belbio_conf api_url: http://localhost:8000
  bel_api:
    container_name: bel_api
    image: belbio/bel_api
```

```
    ports:
      - "8000:8000"
    volumes:
      - ./belbio_conf.yml:/app/belbio_conf.yml
      - ./belbio_secrets.yml:/app/belbio_secrets.yml
    environment:
      - SERVER_MODE=DEV
      - traefik.enable=true
      - traefik.backend=bel_api
      - traefik.frontend.rule=Host:belapi.test;PathPrefixStrip:/v1
      - traefik.port=8000
    # restart: always

  # ArangoDB - document/graph store - core requirement
  # belbio_conf arangodb_host: localhost  port: 8529
  bel_arangodb:
    container_name: bel_arangodb
    image: arangodb:3.3.4
    ports:
      - "8529:8529"
    volumes:
      - bel_arango_data:/var/lib/arangodb3
    environment:
      - ARANGO_NO_AUTH=1
    healthcheck:
         test: curl -f bel_arangodb:8529/_api/version || exit 1
    # restart: always

  # Elasticsearch - terminology search - core requirement
  # belbio_conf elasticsearch: http://localhost:9200
  bel_elasticsearch:
    container_name: bel_elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:6.1.0
    ports:
      - "9200:9200"
    volumes:
      - bel_elasticsearch_data:/usr/share/elasticsearch/data
      - ./elasticsearch.yml:/conf/elasticsearch.yml
    # logging:
    #   driver: none
    environment:
      - cluster.name=docker-cluster
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - xpack.security.enabled=false
      - xpack.watcher.enabled=false
    ulimits:
      memlock:
        soft: -1
        hard: -1
    # deploy:
    #   resources:
    #     limits:
    #       memory: 1g
```

```
    # restart: always
```

2.3 Development Installation

Please read *Contributing* to understand how to contribute to this project.

1. Fork the bel_api project at https://github.com/belbio/bel_api
2. *git clone <your project fork>*
3. *make dev_install*
4. To make bel python package editable *pip install -e <directory of bel python package>*

CHAPTER 3

Contributing

Important: First off, thanks for taking the time to contribute!

The following is a set of guidelines for contributing to BEL.bio and its repositories, which are hosted in BELbio on GitHub. These are mostly guidelines, not rules. Use your best judgment, and feel free to propose changes to this document in a pull request. If you have any questions, please let us know.

You will need to sign a Contributer's License Agreement (CLA) the first time you submit a pull request and anytime afterwards that the CLA agreement is updated (very rarely). This is signed by clicking on the request as part of the pull request and digitally signed with your Github ID automatically.

3.1 Code of Conduct

This project and everyone participating in it is governed by the *Code of Conduct* *Code of Conduct*. By participating, you are expected to uphold this code. Please report unacceptable behavior to one of the project maintainers, William Hayes or Anselmo Di Fabio.

3.2 Code Contributions

This project uses the Git Forking workflow as discussed here https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow

Tip: It may be helpful to review your plans with the community before starting work.

1. Please add an issue to the repository prior to working on a feature or bug.
2. Fork and clone the repository
3. Create a topic branch

4. After completing your changes, please ensure that the code style outlined in Editorconfig is followed.
5. Submit a pull request referencing the issue being resolved

- Other examples of Git Forking workflow http://www.asmeurer.com/git-workflow/ or http://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/

CHAPTER 4

Code of Conduct

4.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

4.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

4.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at William Hayes or Anselmo Di Fabio. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

4.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

CHAPTER 5

DevOps

All administrative tasks are managed by running make tasks using the top-level Makefile in the project folder.

5.1 Builds and Testing

We will use TravisCI for Open Source to run builds and tests.

5.2 Documentation

We use OpenAPI (Swagger) for the API documentation. Due to having to host the SwaggerUI code, we keep the documentation source in *bel_api/make_docs* to hold both the SwaggerUI and Sphinx source. The generated documentation is created in *bel_api/docs* which is deployed using Github Pages.

5.3 Dependabot

We use https://app.dependabot.com/accounts/belbio/repos to keep the python module requirements up to date. It uses the *belbio* user id.

5.4 Code Quality

We are using Code Climate for code quality assessments.

We are using CodeCov for code test coverage assessments.

5.5 Changelog

We are using [Github Changelog Generator](https://github.com/skywinder/github-changelog-generator) for updating the Changelog.

5.6 Contributor Licensing Agreements

All pull requests require signing the [CLA Assistant](https://cla-assistant.io/) Contributor's License Agreement.

CHAPTER 6

Troubleshooting

6.1 Results not up to date

Note: Caching REST API Requests to external servers including Elasticsearch and ArangoDB using Requests CacheControl up to 1 day.

This means that terms, orthologies, pubmed content, etc is cached and may be stale. This can be reset by restarting the BEL.bio API server.

CHAPTER 7

Dependencies and Licensing

7.1 Dependencies

- Python 3.6+
- Falcon - BEL REST API framework python module
- gUnicorn - python WSGI server
- Traefik - Docker friendly reverse proxy (not required if you use your own)
- Docker - for Traefik, gUnicorn/Falcon, ArangoDB, ElasticSearch
- ArangoDB
- ElasticSearch
- bel_lang python package
- bel_nanopub python package

7.2 Licensing

- Apache 2 - BEL.bio API
- Apache 2 - BEL Python Package
- Apache 2 - Elasticsearch https://github.com/elastic/elasticsearch/blob/master/LICENSE.txt
- Apache 2 - ArangoDB - https://www.arangodb.com/documentation/faq/
- MIT - Traefik - https://github.com/containous/traefik/blob/master/LICENSE.md
- Apache 2 - Docker - https://www.docker.com/components-licenses

CHAPTER 8

Glossary

API BEL.bio API – BEL language, BEL Nanopub, BEL Namespaces (and equivalences and orthology) and BEL Edge REST API services

Argument An Argument is a BEL function argument which may be another BEL function, a BEL entity, or a BEL function modifier

> act(p(HGNC:VHL)) directlyIncreases deg(p(HGNC:HIF1A)) – the 'act' function has a BEL function as a parameter p()
>
> p(HGNC:AKT1,pmod(P,T,308)) – the p() function has a BEL entity (HGNC:AKT1) and a pmod() function modifier

AST Abstract Syntax Tree of BEL Assertion or BEL Triple. A hierarchically organized data structure of the BEL Assertion which is very useful for performing transforms such as canonicalization, orthologization or generating computed edges.

AST Function BEL function, e.g. p() or modifier function, e.g. var()

AST NSArg or NSArg Namespace argument, e.g. p(HGNC:AKT1), HGNC:AKT1 is the namespace argument where HGNC is the namespace prefix and AKT1 is the Namespace value.

AST StrArg or StrArg String argument, e.g. pmod(Ph, T, 22), Ph, T and 22 are string arguments.

BEL BEL stands for Biological Expression Language. BEL is a means of capturing biological knowledge in a manner that is human friendly and convertible into computable formats for supporting knowledge-driven analytics. It also serves as a format to share biological knowledge using an open standard.

BEL Assertion The key assertion(s) being made in a BEL Nanopub. It is an expression that represents knowledge of the existence of biological entities and relationships between them that are known to be observed within a particular experiment context (i.e. Experiment Context), based on some source of prior knowledge such as a scientific publication or newly generated experimental data.

It can also refer to the single string version of a BEL triple (combining the subject, relation and object of a BEL triple).

BEL Edge BEL triples, computed BEL canonicalized to standard BEL Namespace IDs and potentially orthologized which are stored in an EdgeStore (a graph database).

BEL Entity A BEL entity is a biological/chemical entity or concept found as a parameter of a BEL function. The BEL entity may be composed of a BEL namespace and an identifier, but it does not have to have a namespace which may lead to an ambiguous identifier for the BEL entity. Example BEL entity: HGNC:AKT1 which is the AKT1 identifier in the HGNC namespace (Human Gene Nomenclature Committee).

BEL Namespaces A terminology used to unambiguously identify a gene/protein, biological process, pathology, cell line, etc. The terms in BEL Namespace have a namespace prefix such as **HGNC** for Human Gene Nomenclature Committee human gene symbols or GO for Gene Ontology. The Namespace definition may contain equivalents to other namespaces; it may also contain hierarchical structure such as an Anatomy namespace or the Gene Ontology.

BEL Nanopub A Nanopub is defined by http://nanopub.org. Quoting their definition:

> A nanopublication is the smallest unit of publishable information: an assertion about anything that can be uniquely identified and attributed to its author.
>
> Individual nanopublications can be cited by others and tracked for their impact on the community.
>
> Nanopublications are a natural response to the explosion of high-quality contextual information that overwhelms the capacity of conventional research articles in scholarly communication.

A BEL Nanopub is an atomic instance of BEL knowledge to represent a biological interaction or fact with an experimental context and provenance. A BEL Nanopub consists of the following parts:

Citation

The identification of the scientific literature or database where the interaction was originally asserted.

BEL Assertion

The biological interaction curated from the Citation.

Evidence

Extracted text that supports the BEL Assertion. For example, this may be a text quotation, or link to a figure, or table within the Citation. This has been through some name changes (called 'Evidence' and 'Support' in BEL Scripts and 'Support' in OpenBEL Nanopub format).

Annotations

The biological context within the experiment where the BEL Statement is observed. For example, if the experiment sample was a biopsy on Human, Lung tissue then you might provide an Annotation of Ncbi Taxonomy with value Homo sapiens and Uberon with value lung epithelium. Additional annotations may be included such as actual gene expression values or timepoints.

Annotations are a managed as a list of (AnnotationType, ID and label). The AnnotationType indicates what type of annotation it is such as Species, Disease, CellLine, etc. The Annotation ID is now managed as a BEL Namespace which will allow for hierarchical queries in the future.

Metadata

Additional data about the Nanopub itself. For example if the Nanopub was curated using a text mining approach we may provide a CurationMethod Annotation with value Text Mining.

BEL Network A subset of the BEL Edges in the EdgeStore that are selected for connectivity and environmental context, e.g. BEL Edges connected to p(HGNC:EGFR) for lung cancer in humans.

BEL Triple Discrete Subject, Relation, Object (SRO) version of BEL assertion

Edge See BEL Edge

EdgeStore A database for storing BEL Edges which is search-able via the BEL.bio API. This is a graph database which allows for network neighborhood and shortest path queries between nodes.

Namespaces see BEL Namespaces

NanopubStore A database for storing Nanopubs with a separate CRUD REST API interface and some basic Web Administration.

Nested BEL Assertion Where the object of a BEL triple is another BEL triple, e.g. *subject relation (subject relation object)*.

NetworkStore A database for Networks with a separate CRUD REST API interface and some basic Web Administration.

CHAPTER 9

Concept Types

9.1 Entity types

Used to define BEL Entities - e.g. HGNC:EGF - based on what function it is in g(), r(), p() could be a Gene, RNA or Protein entity. Chemical compounds and any other substance that is not already described by an entity type are Abundance entity types. The canonical list of entity types are stored in each version of the BEL Specification YAML document.

- Abundance
- Protein
- RNA
- Micro_RNA
- Gene
- Complex
- BiologicalProcess
- Pathology
- Activity
- Variant
- ProteinModification
- AminoAcid
- Location

9.2 Annotation Types

These terms are used as the preferred Annotation type in BEL Nanopubs. They provide a broad classification for terms in the various BEL Namespaces.

- Anatomy
- Cell
- CellLine
- CellStructure
- Pathology
- Species

CHAPTER 10

Other BEL Projects

- OpenBEL, Selventa sponsored community open source project for BEL
- PyBEL, PyBEL is a Python software package that parses BEL scripts, validates their semantics, and facilitates data interchange between common formats and database systems like JSON, CSV, Excel, SQL, CX, and Neo4J.
- Bio2BEL, converting bio databases into OpenBEL Namespaces

Please let us know if we have missed any BEL related projects at whayes at adsworks.com or by adding a Github issue.

CHAPTER 11

Windows Dev Notes

Note: David Chen wrote this up for use on his Windows box. Your situation may be different, but hopefully this is useful to you.

11.1 Notes for Windows users

Install Bash: https://msdn.microsoft.com/en-us/commandline/wsl/about

After installing Bash and setting up your user:

```
apt-get install make
```

These instructions may help you get docker working with Bash for Windows:

```
https://blog.jayway.com/2017/04/19/running-docker-on-bash-on-windows/
```

11.2 Further instructions for Windows users (updated July 5, 2017)

Note: These instructions were tested for `Docker version 17.05.0-ce, build 89658be` and `docker-compose version 1.14.0, build c7bdf9e`.

0. Install Bash for Windows using the instructions above
1. Install Docker CE for Windows and run it: https://store.docker.com/editions/community/docker-ce-desktop-windows
2. Right-click on Docker in your system tray, and click on **Settings**.
3. In the **General** tab, check **Expose daemon on tcp://localhost:2375 without TLS**.

4. In the **Shared Drives** tab, check on the local drive (usually drive C) and click **Apply**. If the settings are not saved after clicking apply, see below. Else, continue.
    - If your drive simply refuses to be checked, it may have to do with the sharing permissions allowed on your account (this seems to be the problem for Microsoft Azure AD accounts).
    - A workaround:
        - **Windows Menu > Administrative Tools > Computer Management > System Tools > Local Users and Groups > Users**
        - On the top menu, click **Actions > New User...**. Set both username and password to "docker" (or whatever you'd like)
        - Uncheck **User must change password at next logon** and check **Password never expires**
        - Switch to this new account and try to access your main files in **C:/Users/your-username**, which will prompt you to authenticate with your username and password
        - Once authenticated, switch back to your main account (do not log out of the docker account) and try the step above **but using the credentials of the new account** (see image below):
        - If issue persists, check the Docker logs by clicking on the **Diagnose and Feedback** tab and selecting **log file**, or open an issue here on Github
5. Open a Windows command line and run `bash` - you should now be in a Bash shell
6. Elevate permission to install the newest version of Docker by running `sudo chown -R {$USERNAME} /usr/local/bin` and replace `{$USERNAME}` with your username
7. Install Docker 17.05.0 using `curl -fsSLO https://get.docker.com/builds/Linux/x86_64/docker-17.05.0-ce.tgz && tar --strip-components=1 -xvzf docker-17.05.0-ce.tgz -C /usr/local/bin`
8. Install docker-compose using `sudo apt install docker-compose`
9. Run `docker --version` to check your version is `>= 17.05.0` after the above installation
10. If not already in the Desktop directory, `cd /mnt/{$DRIVE-LETTER}/Users/{$USERNAME}/Desktop/`. For example, mine was `/mnt/c/Users/DavidChen/Desktop/`
11. `git clone git@github.com:belbio/bel_api.git`
12. `cd bel_api/`
13. `cp api/Config.yml.sample api/Config.yml` and edit Config.yml if necessary.
14. `docker-compose start`
15. The services should now be up and ready.
16. Run `docker-compose logs -f` to view logs. Run `docker-compose stop` to stop all services.

CHAPTER 12

Background

BEL.bio is a clean build of a BEL platform using Python 3.6+ and Docker to increase ease of community use and deployment. Some major enhancements are:

- Supports multiple versions of BEL at the same time
- Improved syntactic and semantic validation over OpenBEL API
- Ability to manage BEL Namespaces individually
- Elasticsearch based namespace searching and term completion
- Python libraries designed to support BEL statement, nanopub, edge parsing

CHAPTER 13

Related Documentation

- BEL Python Package - powers most of the BEL language and processing functionality of the API
- BEL Resource Tools - creates and loads resources used by the API (namespaces, orthology, etc)

Index