
PyBEL-CX Documentation

Release 0.1.4-dev

Charles Tapley Hoyt

Oct 04, 2019

CONTENTS

1	Requirements	3
2	Installation	5
3	Command Line Usage	7
3.1	CX to BEL	7
3.2	BEL to CX	7
4	Command Line Interface Usage	9
5	Programmatic Usage	11
5.1	CX Conversion	11
5.2	NDEx Client	12
6	Indices and tables	15
	Python Module Index	17
	Index	19

A PyBEL extension for interconversion with CX.

REQUIREMENTS

To support this feature, we need 2 command line scripts:

1. Export script: takes a CX document from STDIN stream and write a BEL network out to STDOUT. Writes error message to STDERR when error occurs.
2. Import script: takes a BEL network from STDIN and write a CX network to STDOUT

Here are some other specs for these scripts:

- Relatively easy to deploy.
- Have minimum memory footprint possible.
- The scripts will be run as a task from the server so it is OK if runtime is long.
- They can take extra command line arguments if necessary
- The scripts return 0 when succeeded and returns a non-zero exit code when it failed.

INSTALLATION

PyBEL-CX can be installed easily from [PyPI](#) with the following code in your favorite terminal:

```
$ python3 -m pip install pybel-cx
```

or from the latest code on [GitHub](#) with:

```
$ python3 -m pip install git+https://github.com/pybel/pybel-cx.git
```


COMMAND LINE USAGE

PyBEL-CX installs two command line utilities: `bel_to_cx` and `cx_to_bel`.

3.1 CX to BEL

Running this script has the caveat that the CX document should conform to the schema created by PyBEL-CX.

```
$ cat my_network.cx | cx_to_bel > my_network.bel
```

3.2 BEL to CX

```
$ cat my_network.bel | bel_to_cx > my_network.cx
```


COMMAND LINE INTERFACE USAGE

CLI for PyBEL-CX.

PROGRAMMATIC USAGE

5.1 CX Conversion

This module wraps conversion between `pybel.BELGraph` and CX JSON.

CX is an aspect-oriented network interchange format encoded in JSON with a format inspired by the JSON-LD encoding of Resource Description Framework (RDF). It is primarily used by the Network Data Exchange (NDEx) and more recent versions of Cytoscape.

See also:

- The NDEx Data Model [Specification](#)
- [Cytoscape.js](#)
- CX Support for Cytoscape.js on the Cytoscape [App Store](#)

`pybel_cx.to_cx(graph: pybel.struct.graph.BELGraph) → List[Dict]`
Convert a BEL Graph to a CX JSON object for use with [NDEx](#).

See also:

- [NDEx Python Client](#)
- [PyBEL / NDEx Python Client Wrapper](#)

`pybel_cx.to_cx_file(graph: pybel.struct.graph.BELGraph, file: TextIO, indent: Optional[int] = 2, **kwargs) → None`

Write a BEL graph to a JSON file in CX format.

Parameters

- **graph** – A BEL graph
- **file** – A writable file or file-like
- **indent** – How many spaces to use to pretty print. Change to None for no pretty printing

Example: `>>> from pybel.examples import sialic_acid_graph >>> from pybel_cx import to_cx_file >>> with open('graph.cx', 'w') as f: >>> ... to_cx_file(sialic_acid_graph, f)`

`pybel_cx.to_cx_jsons(graph: pybel.struct.graph.BELGraph, **kwargs) → str`
Dump a BEL graph as CX JSON to a string.

Returns CX JSON string

`pybel_cx.from_cx(cx: List[Dict]) → pybel.struct.graph.BELGraph`
Rebuild a BELGraph from CX JSON output from PyBEL.

Parameters **cx** – The CX JSON for this graph

`pybel_cx.from_cx_file` (*file: TextIO*) → `pybel.struct.graph.BELGraph`

Read a file containing CX JSON and converts to a BEL graph.

Parameters `file` (*file*) – A readable file or file-like containing the CX JSON for this graph

Returns A BEL Graph representing the CX graph contained in the file

`pybel_cx.from_cx_jsons` (*graph_cx_json_str: str*) → `pybel.struct.graph.BELGraph`

Reconstitute a BEL graph from a CX JSON string.

Parameters `graph_cx_json_str` – CX JSON string

Returns A BEL graph representing the CX graph contained in the string

5.2 NDEx Client

Integration with NDEx.

This module provides a wrapper around CX import/export and the NDEx [client](#) to allow for easy upload and download of BEL documents to the [NDEx](#) database.

The programmatic API also provides options for specifying username and password. By default, it checks the environment variables: `NDEX_USERNAME` and `NDEX_PASSWORD`.

`pybel_cx.to_ndex` (*graph, username=None, password=None, debug=False*)

Upload a BEL graph to NDEx.

Parameters

- **graph** (*pybel.BELGraph*) – A BEL graph
- **username** (*Optional[str]*) – NDEx username
- **password** (*Optional[str]*) – NDEx password
- **debug** (*bool*) – If true, turn on NDEx client debugging

Returns The UUID assigned to the network by NDEx

Return type `str`

Example Usage:

```
>>> from pybel.examples import sialic_acid_graph
>>> from pybel_cx import to_ndex
>>> to_ndex(sialic_acid_graph)
```

`pybel_cx.from_ndex` (*network_id, username=None, password=None, debug=False*)

Download a BEL Graph from NDEx.

Warning: This function only will work for CX documents that have been originally exported from PyBEL

Parameters

- **network_id** (*str*) – The UUID assigned to the network by NDEx
- **username** (*Optional[str]*) – NDEx username
- **password** (*Optional[str]*) – NDEx password
- **debug** (*bool*) – If true, turn on NDEx client debugging

Return type `pybel.BELGraph`

Example Usage:

```
>>> from pybel_cx import from_ndex
>>> network_id = '1709e6f3-04a1-11e7-aba2-0ac135e8bacf'
>>> graph = from_ndex(network_id)
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`pybel_cx, ??`

`pybel_cx.cli, 9`

`pybel_cx.cx, 11`

`pybel_cx.ndex_utils, 12`

INDEX

F

`from_cx()` (*in module pybel_cx*), 11
`from_cx_file()` (*in module pybel_cx*), 11
`from_cx_jsons()` (*in module pybel_cx*), 12
`from_ndex()` (*in module pybel_cx*), 12

P

`pybel_cx` (*module*), 1
`pybel_cx.cli` (*module*), 9
`pybel_cx.cx` (*module*), 11
`pybel_cx.ndex_utils` (*module*), 12

T

`to_cx()` (*in module pybel_cx*), 11
`to_cx_file()` (*in module pybel_cx*), 11
`to_cx_jsons()` (*in module pybel_cx*), 11
`to_ndex()` (*in module pybel_cx*), 12