User Documentation

Release 3.0.0

Selventa Inc.

January 28, 2017

Contents

This is the user documentation for the OpenBEL Framework, release 3.0.0, last updated January 28, 2017.

The OpenBEL Framework is an open-platform technology specifically designed to overcome many of the challenges associated with capturing, integrating, and storing biological knowledge within an organization, and sharing the knowledge across the organization and between business partners.

GitHub: https://github.com/OpenBEL/openbel-framework#readme

BEL Portal: http://openbel.org

Selventa: http://www.selventa.com/

Contents:

CHAPTER 1

Overview

CHAPTER 2

Getting Started

2.1 Visualizing a Knowledge Network

1. Download Cytoscape (2.8) at cytoscape.org
2. Install KAM Navigator Cytoscape plugin by following the installation guide
3. Connect the KAM Navigator to the public demo server
   - Set the Webservice Location (WSDL URL) to: http://demo.openbel.org/openbel-ws/belframework.wsdl
4. Load up our example Knowledge Networks
   - Select your target Knowledge Network
   - Explore the KAM Navigator plugin

For further information go to the KAM Navigator user manual

CHAPTER 3

Tutorials

3.1 Building Custom Namespaces

This tutorial walks through the steps required to build a custom namespace and integrate it into your OpenBEL Framework.

View tutorial at: Building Custom Namespaces

3.2 Bootstrapping the Java API

This tutorial walks through the process of bootstrapping the framework's Java API.

View tutorial at: Bootstrapping the Java API

Contents:

3.2.1 Building Custom Namespaces

Sections

- *Definition* defines the importance of a namespace. A format template is also given.
- *Field Format* defines the format of each field.
- *Understanding Entity Encoding* explores entity encoding to enforce functional semantics in BEL.
- *Quick Start Steps* defines the steps to build and integrate the namespace with your OpenBEL Framework instance.

Definition

A namespace defines a catalog of biological entities useful within the BEL language. Namespace files end in the *.belns* extension. The file can be optionally compressed with gzip, but it must use the *.belns.gz* extension.

The namespace allows a user to track their own vocabularies or reconstitute existing vocabularies like EntrezGene or SwissProt. The file consists of two parts:

- Header
  - Contains Namespace section that captures the namespace definition and where it applies.

  - Contains Author section capturing who is responsible for creating the namespace.
  - Contains Citation section that captures the external vocabulary this namespace is based on.
  - Contains Process section that determines how the namespace file is processed by the OpenBEL Framework.
- Values
  - The Values section contains all biological entities in value/encoding pairs. The value and encoding is delimited by the namespace's DelimiterString. Each pair is separated by a newline.
  - Entity encodings allow you to enforce BEL function semantics on a per-entity basis. Jump to *Understanding Entity Encoding*.
  - Format
    * [ENTITY][DelimiterString][ENCODING]
    * Examples
      · AKT1|GRP
      · Abdominal Pain|O
      · difenoxin hydrochloride|A
    * Supports UTF-8 encoded entities.

File Format

The following is the structure of a namespace file:

```
[Namespace]
Keyword=KEYWORD (aka regex: \w+)
NameString=STRING
DomainString=STRING
SpeciesString=STRING
DescriptionString=STRING
VersionString=STRING
CreatedDateTime=ISO8601 DATE/TIME
QueryValueURL=URL

[Author]
NameString=STRING
CopyrightString=STRING
ContactInfoString=STRING

[Citation]
NameString=STRING
DescriptionString=STRING
PublishedVersionString=STRING
PublishedDate=ISO8601 DATE
ReferenceURL=URL

[Processing]
CaseSensitiveFlag=no|yes
DelimiterString=STRING
CacheableFlag=yes

[Values]
```

```
# Single-line comment
{[ENTITY][DelimiterString][ENCODING]
```

Each field name ends with the type information for it.

- KEYWORD
  - Represents a string containing Word Characters only, max length of 8.
  - Preferred namespace prefix when used in BEL language.
- String
  - Represents a user-defined, UTF-8 encoded string (represented as a Java string).
- Date
  - Represents a date in ISO 8601 format.
- DateTime
  - Represents a date/time in ISO 8601 format.
- URL
  - Represents a valid parsable URL.
- Flag
  - Represents a boolean using the values “no” (false) and “yes” (true).

Field Format

Block	Field	Description	Required
Namespace	Keyword	Preferred BEL Keyword, Word Characters, max length of 8	Yes
Namespace	NameString	Namespace name, UTF-8 encoded string	Yes
Namespace	DomainString	One of : "BiologicalProcess", "Chemical", "Gene and Gene Products", "Other"	Yes
Namespace	SpeciesString	Comma-separated list of species taxonomy ids	No
Namespace	DescriptionString	Namespace description, UTF-8 encoded string	No
Namespace	VersionString	Namespace version, UTF-8 encoded string	No
Namespace	CreatedDateTime	Namespace publish timestamp, ISO 8601 Date/Time	Yes
Namespace	QueryValueURL	HTTP URL to query for details on namespace values (must be valid URL)	No
Author	NameString	Namespace's authors, UTF-8 encoded string	Yes
Author	CopyrightString	Namespace's copyright/license information, UTF-8 encoded string	No
Author	ContactInfoString	Namespace author's contact info, UTF-8 encoded string	No
Citation	NameString	Citation name, UTF-8 encoded string	Yes
Citation	DescriptionString	Citation description, UTF-8 encoded string	No
Citation	PublishedVersionString	Citation version, UTF-8 encoded string	No
Citation	PublishedDate	Citation publish timestamp, ISO 8601 Date	No
Citation	ReferenceURL	URL to more citation information (must be valid URL)	No
Processing	CaseSensistiveFlag (unused)	no for case-insensitive lookup, yes for case-sensitive lookup	No
Processing	DelimiterString	User-defined delimiter string that splits namespace value from encoding	Yes
Processing	CacheableFlag (unused)	no to never cache namespace, yes to always cache	No

Understanding Entity Encoding

The entity encoding allows the OpenBEL Framework to enforce functional semantics when processing BEL documents. The biological entities can define a set of encoding flags that indicate which functions apply to this entity. For example whether an entity produces a protein or not.

The valid encoding values are:

Encoding Value	Valid BEL Functions
B	bp(), path()
O	path()
R	r(), m()
M	m()
P	p()
G	g()
A	a(), r(), m(), p(), g(), complex()
C	complex()

An example would be the HGNC Gene Symbol GK4P. It can code for a gene and rna abundance, but not a protein. To capture these semantics we would add the GK4P biological entity to the namespace like:

```
GK4P|GR
```

Quick Start Steps

The quick-start to building and integrating your namespace with the OpenBEL Framework.

1. Grab the example template from the *File Format* section.
2. Customize the field values.
3. Build your biological entity values with proper encodings.
4. Deploy your namespace to a local (file) or remote (http / https) location. For file URL format consult File URI Scheme.
5. Record the URL for this namespace for later. This uniquely identified your namespace.
6. Retrieve the stock resource index from the URL: http://resource.belframework.org/belframework/1.0/index.xml
7. Update the resource index with your namespace entry. Use it's URL retrieved in step 5.
8. Store the customized resource index locally using a local file URL. For file URL format consult File URI Scheme.
9. Open the OpenBEL Framework config/belframework.cfg configuration file and change the 'resource_index_url' to this file URL.
10. Start using your namespace URL in BEL Documents!

3.2.2 Bootstrapping the Java API

System Configuration

Creation of the BEL Framework SystemConfiguration can be done using one of the following constructors:

1. **SystemConfiguration.createSystemConfiguration()**
   - The no-arg method uses the `BELFRAMEWORK_HOME` environment variable to setup the system configuration. The BEL compiler uses this variant; the environment variable is configured in the platform `setenv` script.
2. **SystemConfiguration.createSystemConfiguration(java.io.File)**
   - The file-based method uses a java.io.File to setup the system configuration. The BEL Framework tools use this variant with the `-s` option.
3. **SystemConfiguration.createSystemConfiguration(java.util.Map)**

- The map-based method uses a java.util.Map to setup the system configuration. This is the preferred method of programmatically configuring the BEL Framework for use.

Variable Expansion

There are a set of variables that will automatically be expanded in the system configuration when they are seen. These variables can be used in any value of a name-value pair.

{tmp} Expanded to the system temporary directory.

{home} Expanded to the user's home directory.

{name} Expanded to the user's name.

{dir} Expanded to the current working directory.

{belframework_home} Expanded to the `BELFRAMEWORK_HOME` environment variable.

Programmatic Configuration

This is the simplest method of configuring the BEL Framework SystemConfiguration when using the Java API. The SystemConfiguration class exposes a variety of static fields that define the components of the framework's configuration. See the *belframework.cfg* examples available in the *config* directory of the BEL Framework.

Example:

```
Map<String, String> map = new HashMap<String, String>();
map.put(SystemConfiguration.KAMSTORE_URL_DESC, "jdbc:mysql://localhost:3306");
map.put(SystemConfiguration.FRAMEWORK_WORKING_AREA_DESC, "{tmp}/bel_framework");
// map.put(SystemConfiguration...
SystemConfiguration syscfg = SystemConfiguration.createSystemConfiguration(map);
```