Getting Started with libSBOLj 2.0

This beginner's guide introduces the creation and use of `TopLevel` SBOL 2.0 objects using libSBOLj 2.0. This guide is not meant to be a comprehensive documentation of the library but rather a simple introduction to the basic methods available. In particular, this document demonstrates the usage of the library's API through the creation of `ComponentDefinition` and `Sequence` entities for part of a genetic inverter. A similar approach can be taken for the other SBOL 2.0 `TopLevel` objects, such as `ModuleDefinition`, `Model`, `Collection`, and `GenericTopLevel`. Readers who are not familiar with Java are recommended to use other resources to learn about its basics. The example used throughout this tutorial is listed as "GettingStartedExample.java" in the libSBOLj 2.0 examples directory.

Creating an SBOL Document

An `SBOLDocument` object is a container for `TopLevel` data objects. Therefore, the first step is to create an `SBOLDocument` to put our `TopLevel` objects into. This can be done by calling the `SBOLDocument` constructor as shown below.

```
String prURI = "http://partsregistry.org";
SBOLDocument document = new SBOLDocument();
document.setDefaultURIprefix(prURI);
document.setTypesInURIs(true);
document.setComplete(true);
document.setCreateDefaults(true);
```

The method `setDefaultURIprefix` sets the default URI prefix to the value defined by the `prURI` string. All data objects created following this statement carry this default URI prefix. If we want to allow all top-level identity URIs to include their types, then we can call `document.setTypesInURIs(true)`. The `document.setComplete(true)` statement sets the "complete" flag to `true` for the given `document`. It means that any URIs that cannot be dereferenced to a valid object in the current `SBOLDocument` object cause an exception to be thrown. Finally, the `document.setCreateDefaults(true)` statement sets the "createDefaults" flag to `true` for the given `document`. It means that when an object that must reference a `ComponentInstance` object cannot find the component, but it can find a `ComponentDefinition` object with the same `displayId`, it creates a default `ComponentInstance` object with this `displayId` and references it.

Creating SBOL Data Objects

The next step is to create `TopLevel` objects. All create methods take as parameters an optional URI prefix `String`, a display id `String`, and an optional version `String`. Each create method also has parameters for each required element of a `TopLevel` object. For a `ComponentDefinition`, the only required element is a set of type URIs. The code below creates `ComponentDefinition` objects for a promoter repressed by TetR, a coding sequence for the LacI repressor protein, and a cassette that includes them both. The `types` property creates a set of URIs to specify the biochemical or physical entity for each `ComponentDefinition`, which, in this case, is a double-stranded DNA. The first URI `ComponentDefinition.DNA` is a BioPAX ontology term defined as a constant in the `ComponentDefinition` class. The second URI is a *Chemical Entities of Biological Interest* (ChEBI) term specified by the given URI. Since only the display id is provided to the create functions below, the default URI prefix is used and no version is set.

```
HashSet<URI> types =  new HashSet<URI>(Arrays.asList(ComponentDefinition.DNA,
                        URI.create("http://identifiers.org/chebi/CHEBI:4705")));
ComponentDefinition TetR_promoter = document.createComponentDefinition(
                                  "BBa_R0040",
                                  types);
ComponentDefinition LacI_repressor = document.createComponentDefinition(
                                "BBa_C0012",
                                types);
ComponentDefinition pIKELeftCassette = document.createComponentDefinition(
                                "pIKELeftCassette",
                                types);
```

The create sequence methods take an optional URI prefix `String`, a display id `String`, an optional version `String`, a `String` providing the elements of the sequence, and a URI specifying the encoding used by the sequence. The code below creates sequences for the TetR repressible promoter and LacI coding sequence.

```
Sequence seq_187 = document.createSequence(
                   "seq_187",
                   "tccctatcagtgatagagattgacatccctatcagtgatagagatactgagcac",
                   Sequence.IUPAC_DNA
                   );
String element2 = "atggtgaatgtgaaaccagtaacgttatacgatgtcgcagagtatgccggtgtc"
           + "tcttatcagaccgtttcccgcgtggtgaaccaggccagccacgtttctgcgaaaacgcggga"
           + "aaaagtggaagcggcgatggcggagctgaattacattcccaaccgcgtggcacaacaactgg"
           + "cgggcaaacagtcgttgctgattggcgttgccacctccagtctggccctgcacgcgccgtcg"
           + "caaattgtcgcggcgattaaatctcgcgccgatcaactgggtgccagcgtggtggtgtcgat"
           + "ggtagaacgaagcggcgtcgaagcctgtaaagcggcggtgcacaatcttctcgcgcaacgcg"
           + "tcagtgggctgatcattaactatccgctggatgaccaggatgccattgctgtggaagctgcc"
           + "tgcactaatgttccggcgttatttcttgatgtctctgaccagacacccatcaacagtattat"
           + "tttctcccatgaagacggtacgcgactgggcgtggagcatctggtcgcattgggtcaccagc"
           + "aaatcgcgctgttagcgggcccattaagttctgtctcggcgcgtctgcgtctggctggctgg"
           + "cataaatatctcactcgcaatcaaattcagccgatagcggaacgggaaggcgactggagtgc"
           + "catgtccggttttcaacaaaccatgcaaatgctgaatgagggcatcgttcccactgcgatgc"
           + "tggttgccaacgatcagatggcgctgggcgcaatgcgcgccattaccgagtccgggctgcgc"
           + "gttggtgcggatatctcggtagtgggatacgacgataccgaagacagctcatgttatatccc"
           + "gccgttaaccaccatcaaacaggattttcgcctgctggggcaaaccagcgtggaccgcttgc"
           + "tgcaactctctcagggccaggcggtgaagggcaatcagctgttgcccgtctcactggtgaaa"
           + "agaaaaaccaccctggcgcccaatacgcaaaccgcctctccccgcgcgttggccgattcatt"
           + "aatgcagctggcacgacaggtttcccgactggaaagcgggcaggctgcaaacgacgaaaact"
           + "acgctttagtagcttaataa";
Sequence seq_153 = document.createSequence(
                   "seq_153",
                   element2,
                   Sequence.IUPAC_DNA
                   );
```

Setting and Editing Optional Fields

Next, we set and edit the optional fields for the `TopLevel` objects. For any optional field that is not a set or list, the library provides methods to set its value, unset its value, and check its value. The only exceptions where these methods are not available are the following three fields in the `Identified` class: `persistentIdentity`, `displayId`, and `version`. These fields cannot be edited, since they are crucial to maintaining compliant SBOL objects (see Section 11.2 "Compliant SBOL Objects" of the Specification (Data Model 2.0) for more details). The example code below first sets the name and description of the TetR repressible promoter and LacI repressor. Next, it checks if the name is set for `TetR_promoter`, then unsets it, and sets it to a new name.

```
TetR_promoter.setName("p tetR");
LacI_repressor.setName("lacI");
TetR_promoter.setDescription("TetR repressible promoter");
LacI_repressor.setDescription("lacI repressor from E. coli (+LVA)");
if (TetR_promoter.isSetName()) {
      TetR_promoter.unsetName();
      TetR_promoter.setName("p(tetR)");
}
```

For an optional field that is either a list or a set, the library provides methods for adding, removing, and checking if an element is contained in the list or set. The code below first adds roles to the `TetR_promoter` and `LacI_repressor` using constants in the `SequenceOntology` class that indicates that they are a promoter and coding sequence respectively. Next, it adds a second role to `TetR_promoter`, checks if the role is contained in the set of roles, then removes the role.

```
TetR_promoter.addRole(SequenceOntology.PROMOTER);
LacI_repressor.addRole(SequenceOntology.CDS);
URI TetR_promoter_role2 = URI.create("http://identifiers.org/so/SO:0000613");
TetR_promoter.addRole(TetR_promoter_role2);
if (TetR_promoter.containsRole(TetR_promoter_role2)) {
      TetR_promoter.removeRole(TetR_promoter_role2);
}
```

Fields that are a list or set of objects also include operations to clear, get, and set them. The example code below removes all roles at once by calling `clearRoles()`, gets the set of roles for the `TetR_promoter`, and checks if it is empty. Finally, it sets the entire set of roles (replacing any existing set) by calling `setRoles(Set<URI> roles)`.

```
TetR_promoter.clearRoles();
if (!TetR_promoter.getRoles().isEmpty()) {
      System.out.println("TetR_promoter set is not empty");
}
TetR_promoter.setRoles(new HashSet<URI>(Arrays.asList(
                              SequenceOntology.PROMOTER)));
```

Creating and Editing References

`TopLevel` objects can refer to other `TopLevel` objects. For example, a `ComponentDefinition` object can refer to one or more sequences. This reference is created by calling the `addSequence(URI)` method. Note that since the "complete" flag is set to `true`, the library verifies that all objects referenced are present in this `document`. Methods available for manipulating references are similar to those for the optional fields. The code below adds one sequence for each of the `ComponentDefinition` objects, and then deletes all references for `pIKELeftCassette`. Finally, if we include the last line, an exception is thrown because a sequence with the specified URI cannot be found.

```
TetR_promoter.addSequence(seq_187);
LacI_repressor.addSequence(seq_153);
pIKELeftCassette.addSequence(seq_187);
pIKELeftCassette.clearSequences();
//pIKELeftCassette.addSequence(
//   URI.create("http://partsregistry.org/seq/partseq_154"));
```

Creating Annotations

In order to allow representation of data that can not currently be represented by the SBOL data model or data that are outside the scope of SBOL, SBOL offers developers the ability to embed custom data as annotations of SBOL objects and as generic top-level objects. These data are exchanged unmodified between software tools that adopt libSBOLj 2.0.

Each object in SBOL 2.0 can be annotated by having any number of `Annotation` objects that store data in the form of name/value property pairs. The name of an annotation must be a `QName` object, which is composed of a namespace, a local name, and an optional prefix. The value of an annotation must contain a literal (i.e., a `String`, `int`, `double`, `boolean`), URI, or NestedAnnotations object. The code snippet below creates an annotation for `TetR_promoter`. First, a new namespace is added to `document`. It creates a short name "pr" for the `prURI` previously defined as the default URI prefix for this SBOL document. This annotation is named "pr:experience", and is composed of the `prURI` namespace, a local name "experience" and its prefix "pr". It contains a URI value that can be resolved to the information web page on the Parts Registry for promoter "BBa_R0040".

```
String prPrefix = "pr";
document.addNamespace(URI.create(prURI) , prPrefix);
TetR_promoter.createAnnotation(new QName(prURI, "experience", prPrefix),
      URI.create("http://parts.igem.org/Part:BBa_R0040"));
```

Creating Generic TopLevel Object

To embed custom data directly in an SBOL document, we can store them using `GenericTopLevel` objects. The example code below first creates such an object `datasheet`, whose display ID is "datasheet" and version is "1.0". Its required RDF type property is a `QName` object named "myersLab:datasheet". Then we set its name to "Datasheet for Custom Parameters". Custom data are encoded as annotations of this generic top-level object. The first one is the characterization data with a URI value that can be resolved to a location where measurement data can be found. The next anntation stores the value of the transcription rate, which is 0.75. The last three lines create an annotation for `TetR_promoter` that refers to the `datasheet` object.

```
String myersLabURI = "http://www.async.ece.utah.edu";
String myersLabPrefix = "myersLab";
GenericTopLevel datasheet=document.createGenericTopLevel(
      "datasheet",
      "1.0",
      new QName(myersLabURI, "datasheet", myersLabPrefix));
datasheet.setName("Datasheet for Custom Parameters");
datasheet.createAnnotation(
     new QName(myersLabURI, "characterizationData", myersLabPrefix),
     URI.create(myersLabURI + "/measurement/Part:BBa_R0040"));
datasheet.createAnnotation(
     new QName(myersLabURI, "transcriptionRate", myersLabPrefix),
     0.75);
TetR_promoter.createAnnotation(
     new QName(myersLabURI, "datasheet", myersLabPrefix),
     datasheet.getIdentity());
```

Creating and Editing Child Objects

The `ModuleDefinition` and `ComponentDefinition` objects have child objects that can be created and edited. The code below creates a child object for `pIKELeftCassette`. Namely, it creates a child `SequenceConstraint` object that says the `TetR_promoter` precedes the `LacI_repressor`. A sequence constraint has a "subject" and an "object" reference that point to two different `Component` objects. They do not currently exist in our document and need to be created. With the "createDefaults" flag set to `true`, creation of these components is done

automatically. Each component created has the same `displayId` as its corresponding `ComponentDefinition` object. They are added to `pIKELeftCassette`'s set of components. This can be verified by calling the `getComponent` method. Finally, if we include the last line, it causes an exception, since the component being removed is used by a sequence constraint.

```
pIKELeftCassette.createSequenceConstraint(
       "pIKELeftCassette_sc",
       RestrictionType.PRECEDES,
       TetR_promoter.getDisplayId(),
       LacI_repressor.getDisplayId()
       );
if (pIKELeftCassette.getComponent("BBa_R0040")==null) {
      System.out.println("TetR_promoter component is missing");
}
if (pIKELeftCassette.getComponent("BBa_C0012")==null) {
      System.out.println("LacI_repressor component is missing");
}
//pIKELeftCassette.removeComponent(
//                          pIKELeftCassette.getComponent("BBa_R0040"));
```

Copying Objects

The library can make copies of `TopLevel` objects using the `createCopy` methods. There are several variations of this method. If only the object is provided, an identical copy is created. Note that this will cause an exception if it is copied into the same document, since it will not have a unique identity. Next, a new display id can be provided, which results in a copy with the display ids (and identities) updated accordingly. Similarly, a version can be provided as well, which results in a copy with the version (and identities) updated accordingly. Finally, a new URI prefix can also be provided, once again resulting in updated identities. The example below makes a copy of the `TetR_promoter` object by calling `createCopy` with a new display id, `BBa_K137046`. The identity URI for `TetR_promoter_copy` is changed to include the new display id, and this change also percolates through the identity URIs for any descendent objects. This example gives the new copy a new sequence, but it keeps all other properties the same.

```
ComponentDefinition TetR_promoter_copy =
    (ComponentDefinition)document.createCopy(TetR_promoter, "BBa_K137046");
Sequence seq = document.createSequence(
              "seq_K137046",
              "gtgctcagtatctctatcactgatagggatgtcaatctctatcactgatagggactctagtatat"
              + "aaacgcagaaaggcccacccgaaggtgagccagtgtgactctagtagagagcgttcaccgaca"
              + "aacaacagataaaacgaaaggc",
              Sequence.IUPAC_DNA
              );
TetR_promoter_copy.addSequence(seq);
```

Serialization

The library supports reading and writing data encoded in RDF/XML format. We can produce a serialization output by calling various write methods in the `SBOLWriter` class. These methods write to either an output stream in the form of Java OutputStream object or a file, some of which are demonstrated below. The first method call produces an output stream and the second stores the output to file "GettingStartedExample.rdf". Note that these two method calls do not specify the output file type, and the library's default serialization output format is RDF/XML.

```
SBOLWriter.write(document,(System.out));
SBOLWriter.write(document, "GettingStartedExample.rdf");
```

Reading of a file or an input stream in the form of Java InputStream object is supported by similar read methods in the `SBOLReader` class. The default input format for any of these read methods is also RDF/XML. The method below first writes the `SBOLDocument` "`doc`" to a Java `ByteArrayOutputStream`, and then reads it back to a new `SBOLDocument` object.

```
public static SBOLDocument writeThenRead(SBOLDocument doc)
             throws SBOLValidationException, IOException, XMLStreamException,
                 FactoryConfigurationError, CoreIoException
{
  ByteArrayOutputStream out = new ByteArrayOutputStream();
  SBOLWriter.write(doc, out);
  return SBOLReader.read(new ByteArrayInputStream(out.toByteArray()));
}
```

The RDF serialization for the SBOLDocument object created in this tutorial is shown below. Note that we only show the beginning of the actual sequences for `seq_K137046` and `partseq_153` in the serialization output below. Actual serialization includes the complete sequences.

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:myersLab="http://www.async.ece.utah.edu" xmlns:prov="http://www.w3.org/ns/prov#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:pr="http://partsregistry.org"
     xmlns:sbol="http://sbols.org/v2#" xmlns:dcterms="http://purl.org/dc/terms/">
  <sbol:ComponentDefinition rdf:about="http://partsregistry.org/cd/BBa_R0040">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/cd/BBa_R0040"/>
    <sbol:displayId>BBa_R0040</sbol:displayId>
    <dcterms:title>p(tetR)</dcterms:title>
    <dcterms:description>TetR repressible promoter</dcterms:description>
    <pr:experience rdf:resource="http://parts.igem.org/Part:BBa_R0040"/>
    <myersLab:datasheet rdf:resource="http://partsregistry.org/gen/datasheet/1.0"/>
    <sbol:type rdf:resource="http://identifiers.org/chebi/CHEBI:4705"/>
    <sbol:type rdf:resource="http://www.biopax.org/release/biopax-level3.owl#DnaRegion"/>
    <sbol:role rdf:resource="http://identifiers.org/so/SO:0000167"/>
    <sbol:sequence rdf:resource="http://partsregistry.org/seq/partseq_187"/>
  </sbol:ComponentDefinition>
  <sbol:ComponentDefinition rdf:about="http://partsregistry.org/cd/BBa_C0012">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/cd/BBa_C0012"/>
    <sbol:displayId>BBa_C0012</sbol:displayId>
    <dcterms:title>lacI</dcterms:title>
    <dcterms:description>lacI repressor from E. coli (+LVA)</dcterms:description>
    <sbol:type rdf:resource="http://identifiers.org/chebi/CHEBI:4705"/>
    <sbol:type rdf:resource="http://www.biopax.org/release/biopax-level3.owl#DnaRegion"/>
    <sbol:role rdf:resource="http://identifiers.org/so/SO:0000316"/>
    <sbol:sequence rdf:resource="http://partsregistry.org/seq/partseq_153"/>
  </sbol:ComponentDefinition>
  <sbol:ComponentDefinition rdf:about="http://partsregistry.org/BBa_K137046">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/BBa_K137046"/>
    <sbol:displayId>BBa_K137046</sbol:displayId>
    <prov:wasDerivedFrom rdf:resource="http://partsregistry.org/cd/BBa_R0040"/>
    <dcterms:title>p(tetR)</dcterms:title>
    <dcterms:description>TetR repressible promoter</dcterms:description>
    <pr:experience rdf:resource="http://parts.igem.org/Part:BBa_R0040"/>
    <myersLab:datasheet rdf:resource="http://partsregistry.org/gen/datasheet/1.0"/>
    <sbol:type rdf:resource="http://identifiers.org/chebi/CHEBI:4705"/>
    <sbol:type rdf:resource="http://www.biopax.org/release/biopax-level3.owl#DnaRegion"/>
    <sbol:role rdf:resource="http://identifiers.org/so/SO:0000167"/>
    <sbol:sequence rdf:resource="http://partsregistry.org/seq/partseq_187"/>
    <sbol:sequence rdf:resource="http://partsregistry.org/seq/seq_K137046"/>
  </sbol:ComponentDefinition>
  <sbol:ComponentDefinition rdf:about="http://partsregistry.org/cd/pIKELeftCassette">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/cd/pIKELeftCassette"/>
    <sbol:displayId>pIKELeftCassette</sbol:displayId>
    <sbol:type rdf:resource="http://identifiers.org/chebi/CHEBI:4705"/>
    <sbol:type rdf:resource="http://www.biopax.org/release/biopax-level3.owl#DnaRegion"/>
    <sbol:component>
      <sbol:Component rdf:about="http://partsregistry.org/cd/pIKELeftCassette/BBa_R0040">
        <sbol:persistentIdentity
            rdf:resource="http://partsregistry.org/cd/pIKELeftCassette/BBa_R0040"/>
        <sbol:displayId>BBa_R0040</sbol:displayId>
        <sbol:access rdf:resource="http://sbols.org/v2#public"/>
        <sbol:definition rdf:resource="http://partsregistry.org/cd/BBa_R0040"/>
      </sbol:Component>
    </sbol:component>
    <sbol:component>
      <sbol:Component rdf:about="http://partsregistry.org/cd/pIKELeftCassette/BBa_C0012">
        <sbol:persistentIdentity
            rdf:resource="http://partsregistry.org/cd/pIKELeftCassette/BBa_C0012"/>
        <sbol:displayId>BBa_C0012</sbol:displayId>
        <sbol:access rdf:resource="http://sbols.org/v2#public"/>
        <sbol:definition rdf:resource="http://partsregistry.org/cd/BBa_C0012"/>
      </sbol:Component>
    </sbol:component>
    <sbol:sequenceConstraint>
      <sbol:SequenceConstraint
          rdf:about="http://partsregistry.org/cd/pIKELeftCassette/pIKELeftCassette_sc">
        <sbol:persistentIdentity
            rdf:resource="http://partsregistry.org/cd/pIKELeftCassette/pIKELeftCassette_sc"/>
        <sbol:displayId>pIKELeftCassette_sc</sbol:displayId>
        <sbol:restriction rdf:resource="http://sbols.org/v2#precedes"/>
        <sbol:subject rdf:resource="http://partsregistry.org/cd/pIKELeftCassette/BBa_R0040"/>
        <sbol:object rdf:resource="http://partsregistry.org/cd/pIKELeftCassette/BBa_C0012"/>
      </sbol:SequenceConstraint>
    </sbol:sequenceConstraint>
  </sbol:ComponentDefinition>
```

```
  <sbol:Sequence rdf:about="http://partsregistry.org/seq/seq_K137046">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/seq/seq_K137046"/>
    <sbol:displayId>seq_K137046</sbol:displayId>
    <sbol:elements>gtgctcagtatctctatcactgatagggatgtcaatctctatcactgata ...</sbol:elements>
    <sbol:encoding rdf:resource="http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html"/>
  </sbol:Sequence>
  <sbol:Sequence rdf:about="http://partsregistry.org/seq/partseq_187">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/seq/partseq_187"/>
    <sbol:displayId>partseq_187</sbol:displayId>
    <sbol:elements>tccctatcagtgatagagattgacatccctatcagtgatagagatactgagcac</sbol:elements>
    <sbol:encoding rdf:resource="http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html"/>
  </sbol:Sequence>
  <sbol:Sequence rdf:about="http://partsregistry.org/seq/partseq_153">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/seq/partseq_153"/>
    <sbol:displayId>partseq_153</sbol:displayId>
    <sbol:elements>atggtgaatgtgaaaccagtaacgttatacgatgtcgcagagtatgccgg ...</sbol:elements>
    <sbol:encoding rdf:resource="http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html"/>
  </sbol:Sequence>
  <myersLab:datasheet rdf:about="http://partsregistry.org/gen/datasheet/1.0">
    <sbol:persistentIdentity rdf:resource="http://partsregistry.org/gen/datasheet"/>
    <sbol:displayId>datasheet</sbol:displayId>
    <sbol:version>1.0</sbol:version>
    <dcterms:title>Datasheet for Custom Parameters</dcterms:title>
    <myersLab:characterizationData
        rdf:resource="http://www.async.ece.utah.edu/measurement/Part:BBa_R0040"/>
    <myersLab:transcriptionRate>0.75</myersLab:transcriptionRate>
  </myersLab:datasheet>
</rdf:RDF>
```