

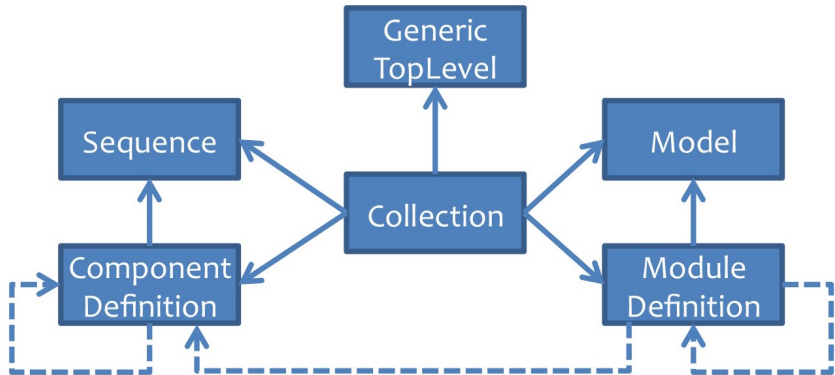
Introduction to libSBOLj and sboljs

Chris J. Myers

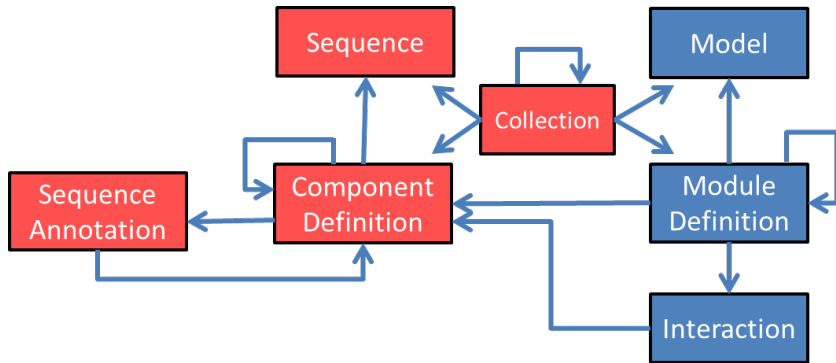
University of Utah

SBOL Workshop
August 11, 2017

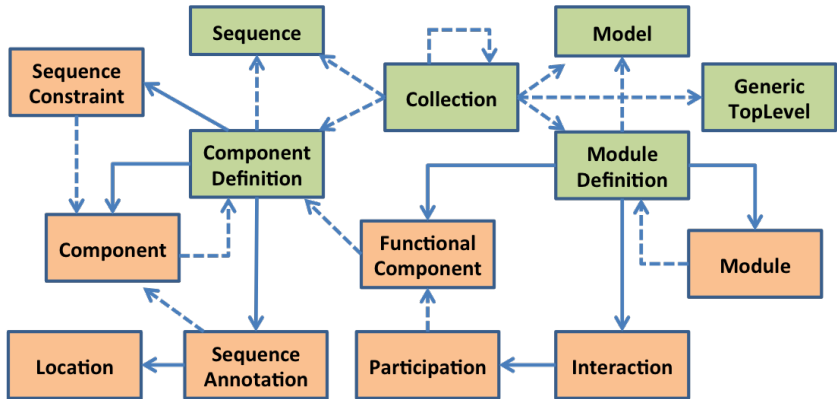
SBOL 2 Data Model



SBOL 2 Data Model



SBOL 2 Data Model



SBOL Libraries

- Crucial to the success of a standard is software infrastructure to support developers' integration of the standard within their tools.
- There are several library implementations of the SBOL data structure, which provide an *application programmers interface* (API) for tool developers to interact with SBOL data objects.
 - libSBOLj - native Java library
 - sboljs - Javascript library
 - libSBOL - C/C++ library
 - pySBOL - Python library

SBOL Libraries

- Crucial to the success of a standard is software infrastructure to support developers' integration of the standard within their tools.
- There are several library implementations of the SBOL data structure, which provide an *application programmers interface* (API) for tool developers to interact with SBOL data objects.
 - libSBOLj - native Java library
 - sboljs - Javascript library
 - libSBOL - C/C++ library
 - pySBOL - Python library

- Library available from Maven central:
 - Group Id - org.sbolstandard
 - Artifact Id - libSBOLj
 - Version - 2.2.1
- Distribution includes detailed documentation for the class definitions and the methods provided by the API.
- Supports validation/conversion to/from FASTA, GenBank, and SBOL1.
- Includes support to search, fetch, and submit designs to SynBioHub.
- Utilized by an online validator/converter available from the SBOL website, and it also provides a webservice to be used by non-java applications.

SBOL Documents

- libSBOLj organizes all SBOL data within an **SBOLDocument**.
- Includes a list of each type of *top level* object:
 - **Collections**,
 - **ModuleDefinitions**,
 - **ComponentDefinitions**,
 - **Sequences**,
 - **Models**, and
 - **GenericTopLevel** objects.
- These lists are organized to allow for easy search by their *unique reference identifiers* (URIs) and validation that they are distinct.
- libSBOLj includes methods for creating, updating, accessing, and removing these data objects, as well as, their child objects.

Creating an SBOLDocument

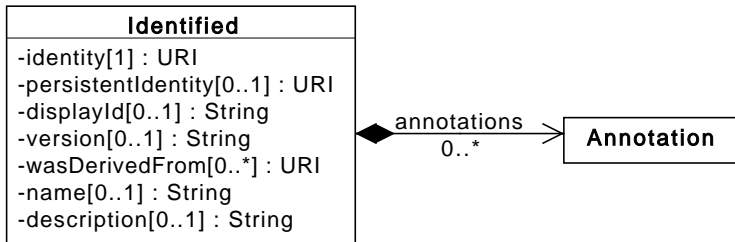
```
String uriPrefix = "http://cellocad.org/";  
SBOLDocument document = new SBOLDocument();  
document.setDefaultURIPrefix(uriPrefix);  
document.setComplete(true);  
document.setCreateDefaults(true);
```

- Default URI prefix - prefix to use when none provided to create method.
- Complete - ensure that all URI references point to valid SBOL objects.
- Create defaults - implicitly create **ComponentInstances** as needed.

Understanding UML Diagrams

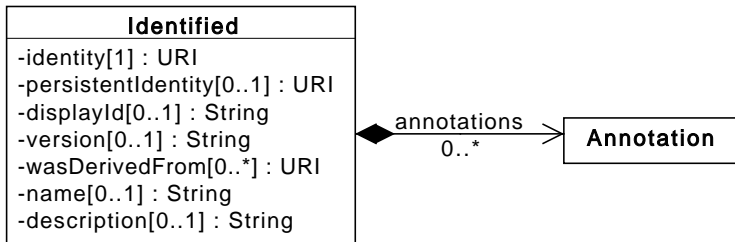
- Classes represented as boxes labeled with their member variables with types and cardinalities.
- Arrows with cardinality indicate associations between classes.
- A hollow diamond at the origin of an arrow represents shared aggregation (i.e., object is referenced and not owned).
- A solid diamond at the origin of an arrow represents composite aggregation (i.e., child object is owned by its parent object).
- Hollow arrows are used to represent inheritance.

Identified (Base Class for All SBOL Objects)



- *identity* - globally unique URI to identify this object (required).
- *persistentIdentity* - identity shared by multiple versions of the same object (optional).
- *displayId* - human-readable id composed of alphanumeric and underscore characters (optional).
- *version* - uses semantic versioning to identify multiple versions of the same object (optional).

Identified (Base Class for All SBOL Objects)



- *wasDerivedFrom* - identities of objects that this is derived from (optional).
- *name* - human-readable String of arbitrary characters (optional).
- *description* - thorough text description of the object (optional).
- *annotations* - additional data about this object (more later).

Compliant Top-Level URIs

$\langle \text{URI prefix} \rangle / \langle \text{displayId} \rangle / \langle \text{version} \rangle$

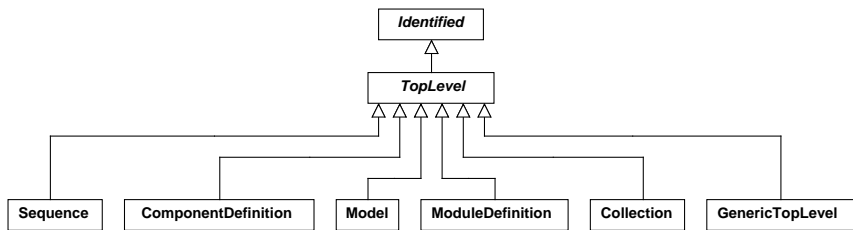
- 1 The *identity* MUST begin with a *URI prefix* that maps to a domain over which the user has control.
- 2 The *persistentIdentity* and *displayId* properties are REQUIRED.
- 3 The *persistentIdentity* MUST end with a delimiter ('/', '#', or ':') followed by the *displayId* of the object.
- 4 If an object is not given a *version*, then its *identity* and *persistentIdentity* properties MUST contain the same *URI*.
- 5 If an object has a *version*, then its *identity* property MUST contain a *URI* of the form $\langle \text{persistentIdentity} \rangle / \langle \text{version} \rangle$.

Compliant Child URIs

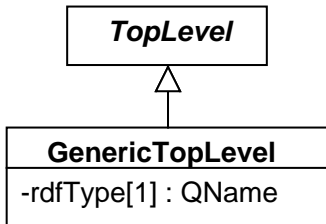
$\langle \textit{parent persistentIdentity} \rangle / \langle \textit{displayId} \rangle / \langle \textit{parent version} \rangle$

- 1 The *persistentIdentity* MUST begin with the *persistentIdentity* of its parent object and be immediately followed by a delimiter ('/', '#', or ':') and the *displayId* of the object.
- 2 The *version* MUST contain the same *String* as the *version* property of the object's parent object.

TopLevel



GenericTopLevel



GenericTopLevel Objects and Annotations

- Software tools that need to store data that is not encoded within SBOL can do so using **GenericTopLevel** objects and custom **Annotations**.
- When the library reader encounters a tag for a **TopLevel** object that it does not recognize, this data is stored within a **GenericTopLevel** object.
- Within **TopLevel** objects, when a tag is not recognized the data is stored within a custom **Annotation** object.
- Tools using our library that do not recognize this data will round-trip it unmodified when writing an SBOL file.
- Tools that would like to make use of this data can interpret and manipulate the raw data, which is stored in a tree-like data structure.

Creating SBOL Data Objects

- SBOLDocument class includes create methods for each TopLevel object.
 - displayId only - URI prefix taken from default, no version
 - displayId, version - URI prefix taken from default
 - URIprefix, displayId, version
- All required fields are also parameters to these create methods.

create<TopLevel>(URIprefix, displayId, version, <required fields>)

```
String provNS = "http://www.w3.org/ns/prov#";
GenericTopLevel genericTopLevel = document.createGenericTopLevel("cello2sbol", version,
    new QName(provNS, "Activity", "prov"));
```

Setting and Editing Optional Fields

- Methods to set/unset each optional field, as well as check if isSet.

```
genericTopLevel.setName("Cello_UCF_to_SBOL_conversion");  
genericTopLevel.setDescription("Conversion_of_the_Cello_UCF_parts_and_metadata_to_SBOL2");
```

- Methods to create annotations (String, URI, int, double, and nested).

```
String dcNS = "http://purl.org/dc/elements/1.1/";  
genericTopLevel.createAnnotation(new QName(dcNS, "creator", "dc"), "Prashant_Vaidyanathan");  
genericTopLevel.createAnnotation(new QName(dcNS, "creator", "dc"), "Chris_J._Myers");  
TimeZone tz = TimeZone.getTimeZone("UTC");  
DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'");  
df.setTimeZone(tz);  
createdDate = df.format(new Date());  
genericTopLevel.createAnnotation(new QName(provNS, "endedAtTime", "prov"), createdDate);
```

GenericTopLevel (Example)

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sbol="http://sbols.org/v2#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:prov="http://www.w3.org/ns/prov#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <prov:Activity rdf:about="http://cellocad.org/cello2sbol/1">
    <sbol:persistentIdentity rdf:resource="http://cellocad.org/cello2sbol/">
    <sbol:displayId> cello2sbol</sbol:displayId>
    <sbol:version> 1</sbol:version>
    <dcterms:title> Cello UCF to SBOL conversion</dcterms:title>
    <dcterms:description> Conversion of the Cello UCF parts and metadata to SBOL2
    </dcterms:description>
    <dc:creator> Prashant Vaidyanathan</dc:creator>
    <dc:creator> Chris J. Myers</dc:creator>
    <prov:endedAtTime>2017-08-09T00:25Z</prov:endedAtTime>
  </prov:Activity>
</rdf:RDF>
```

- JSON file format including the following sections:
 - Parts - basic DNA parts (promoters, rbs, cds, terminators, etc.)
 - Gate parts - compositions of DNA parts to form complete functions
 - Gates - additional information about these gate parts
 - Other information - response functions, gate toxicity, etc.

Cello UCF File Example

```
{
  "collection": "parts",
  "type": "cds",
  "name": "AmtR",
  "dnasequence": "ATGGCAGGCCGAGTTGGTTCGTCGCCGTCGTAGTGCACCGCGTCGTGCAGGTA AAAATCCGCGTGAAGAAATTC..."
}, ...
{
  "collection": "gates",
  "regulator": "AmtR",
  "group_name": "AmtR",
  "gate_name": "A1_AmtR",
  "gate_type": "NOR",
  "system": "TetR",
  "color_hexcode": "3BA9E0"
}, ...
{
  "collection": "gate_parts",
  "gate_name": "A1_AmtR",
  "expression_cassettes": [
    {
      "maps_to_variable": "x",
      "cassette_parts": [
        "BydvJ",
        "A1",
        "AmtR",
        "L3S2P55"
      ]
    }
  ]
},
  "promoter": "pAmtR"
}, ...
```

Parsing Cello JSON File

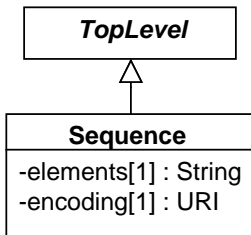
```
HashMap<String, JSONObject> partsMap = new HashMap<String, JSONObject> ();
HashSet<JSONObject> gate_partsArr = new HashSet<JSONObject> ();
HashMap<String, JSONObject> gatesMap = new HashMap<String, JSONObject> ();

JSONParser parser = new JSONParser ();
JSONArray a = (JSONArray) parser.parse(new FileReader(JSONfilename));

for (Object o : a)
{
    JSONObject ucf = (JSONObject) o;
    String collection = (String) ucf.get("collection");

    if (collection.equals("parts")) {
        partsMap.put((String)ucf.get("name"), ucf);
    }
    else if (collection.equals("gate_parts")) {
        gate_partsArr.add(ucf);
    }
    else if (collection.equals("gates")) {
        gatesMap.put((String)ucf.get("gate_name"), ucf);
    }
}
```

Sequence



- *elements* - String of characters representing constituents of a biological or chemical molecule.
- *encoding* - URI indicating how elements are to be interpreted.

Encoding

IUPAC DNA, RNA

IUPAC Protein

SMILES

URI

<http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html>

<http://www.chem.qmul.ac.uk/iupac/AminoAcid/>

<http://www.opensmiles.org/opensmiles.html>

CD Type

DNA, RNA

Protein

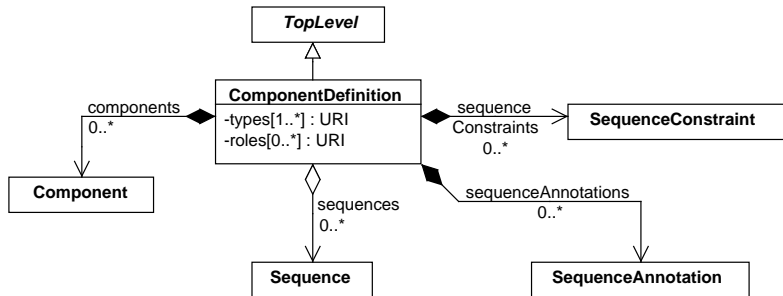
SmallMolecule

Creating Sequences for Parts

```
for (JSONObject part : partsMap.values()) {  
  
    String name = (String)part.get("name");  
    String dnasequence = (String)part.get("dnasequence");  
    Sequence sequence = document.createSequence(name + "_sequence", version, dnasequence,  
        Sequence.IUPAC_DNA);  
    sequence.setName(name+"_sequence");  
    sequence.addWasDerivedFrom(derivedFrom);  
    sequence.createAnnotation(new QName(provNS, "wasGeneratedBy", "prov"), activityURI);  
    sequence.createAnnotation(new QName(dcTermsNS, "created", "dcTerms"), createdAt);  
}
```

```
<sbol:Sequence rdf:about="http://cellocad.org/AmtR_sequence/1">  
  <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_sequence"/>  
  <sbol:displayId> AmtR_sequence </sbol:displayId>  
  <sbol:version> 1 </sbol:version>  
  <prov:wasDerivedFrom rdf:resource="https://github.com/CIDARLAB/cello/blob/master/resources/UCF/E">  
  <dcterms:title> AmtR_sequence </dcterms:title>  
  <prov:wasGeneratedBy rdf:resource="http://cellocad.org/cello2sbol/1"/>  
  <dcterms:created> 2017-08-09T13:54Z </dcterms:created>  
  <sbol:elements> ATGGCAGGCGCAGTTGGTCGTCGCGTCGTAGTGCACCGCGTCGTGCAGGTAAAAATCCGCGTGAAGAAATTCTGGATGC  
  <sbol:encoding rdf:resource="http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html"/>  
</sbol:Sequence>
```

ComponentDefinition



ComponentDefinition (Types/Roles)

CD Type

URI for BioPAX Term

DNA	http://www.biopax.org/release/biopax-level3.owl#DnaRegion
RNA	http://www.biopax.org/release/biopax-level3.owl#RnaRegion
Protein	http://www.biopax.org/release/biopax-level3.owl#Protein
Small Molecule	http://www.biopax.org/release/biopax-level3.owl#SmallMolecule
Complex	http://www.biopax.org/release/biopax-level3.owl#Complex

CD Role

URI for SequenceOntology Term

CD Type

Promoter	http://identifiers.org/so/SO:0000167	DNA
RBS	http://identifiers.org/so/SO:0000139	DNA
CDS	http://identifiers.org/so/SO:0000316	DNA
Terminator	http://identifiers.org/so/SO:0000141	DNA
Gene	http://identifiers.org/so/SO:0000704	DNA
Operator	http://identifiers.org/so/SO:0000057	DNA
Engineered Gene	http://identifiers.org/so/SO:0000280	DNA
mRNA	http://identifiers.org/so/SO:0000234	RNA

Converting Part Type to SBOL ComponentDefinition Role

```
public static URI getRole(String type) {
String so = "http://identifiers.org/so/";
if (type.equals("ribozyme")) {
    return URI.create(so + "SO:0000374");
}
else if (type.equals("scar")) {
    return URI.create(so + "SO:0001953");
}
else if (type.equals("cds")) {
    return URI.create(so + "SO:0000316");
}
else if (type.equals("promoter")) {
    return URI.create(so + "SO:0000167");
}
else if (type.equals("rbs")) {
    return URI.create(so + "SO:0000139");
}
else if (type.equals("terminator")) {
    return URI.create(so + "SO:0000141");
} else {
    System.err.println("Part_Type_not_found");
    return null;
}
}
```

Converting Parts to SBOL ComponentDefinitions

```
ComponentDefinition componentDefinition =
    document.createComponentDefinition(name, version, ComponentDefinition.DNA);
componentDefinition.setName(name);
componentDefinition.addWasDerivedFrom(derivedFrom);
componentDefinition.createAnnotation(new QName provNS, "wasGeneratedBy", "prov"), activityURI);
componentDefinition.createAnnotation(new QName(dcTermsNS, "created", "dcTerms"), createdAt);
String partType = (String)part.get("type");
componentDefinition.addRole(getRole(partType));
componentDefinition.addSequence(sequence);

if (partType.equals("cds")) {

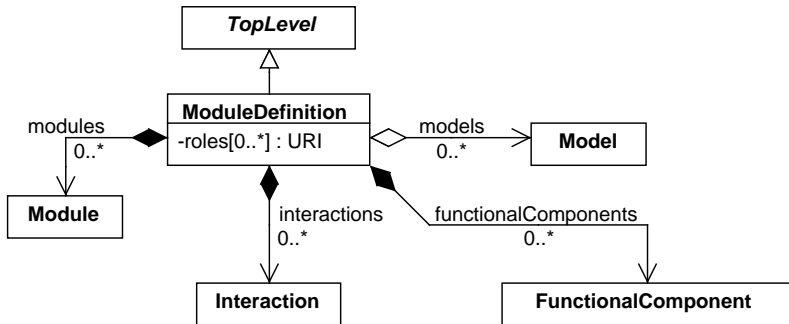
    ComponentDefinition proteinComponentDefinition =
        document.createComponentDefinition(name+"_protein", version, ComponentDefinition.PROTEIN);
    proteinComponentDefinition.setName(name+"_protein");
    proteinComponentDefinition.createAnnotation(new QName provNS, "wasGeneratedBy", "prov"),
        activityURI);
    proteinComponentDefinition.createAnnotation(new QName(dcTermsNS, "created", "dcTerms"),
        createdAt);
}
```

Converting Parts to SBOL ComponentDefinitions

```
<sbol:ComponentDefinition rdf:about="http://cellocad.org/AmtR/1">
  <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR"/>
  <sbol:displayId> AmtR </sbol:displayId>
  <sbol:version> 1 </sbol:version>
  <prov:wasDerivedFrom rdf:resource="https://github.com/CIDARLAB/cello/blob/master/resources/UCF/E">
  <dcterms:title> AmtR </dcterms:title>
  <prov:wasGeneratedBy rdf:resource="http://cellocad.org/cello2sbol/1"/>
  <dcterms:created> 2017-08-09T13:54Z </dcterms:created>
  <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://cellocad.org/AmtR_sequence/1"/>
</sbol:ComponentDefinition>

<sbol:ComponentDefinition rdf:about="http://cellocad.org/AmtR_protein/1">
  <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein"/>
  <sbol:displayId> AmtR_protein </sbol:displayId>
  <sbol:version> 1 </sbol:version>
  <dcterms:title> AmtR_protein </dcterms:title>
  <prov:wasGeneratedBy rdf:resource="http://cellocad.org/cello2sbol/1"/>
  <dcterms:created> 2017-08-09T13:54Z </dcterms:created>
  <sbol:type rdf:resource="http://www.biopax.org/release/biopax-level3.owl#Protein"/>
</sbol:ComponentDefinition>
```

ModuleDefinition

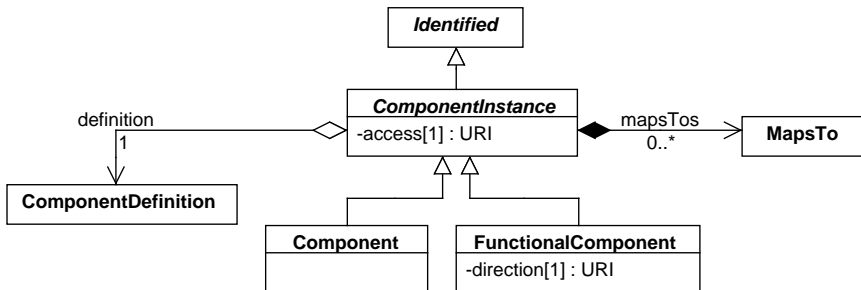


Creating ModuleDefinitions for Genetic Production

```
ModuleDefinition moduleDefinition =
    document.createModuleDefinition(name+"_protein_production", version);
moduleDefinition.setName(name+"_protein_production");
moduleDefinition.createAnnotation(new QName(provNS, "wasGeneratedBy", "prov"), activityURI);
moduleDefinition.createAnnotation(new QName(dcTermsNS, "created", "dcTerms"), createdDate);
```

```
<sbol:ModuleDefinition rdf:about="http://cellocad.org/AmtR_protein_production/1">
  <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein_production"/>
  <sbol:displayId> AmtR_protein_production </sbol:displayId>
  <sbol:version> 1 </sbol:version>
  <dcterms:title> AmtR_protein_production </dcterms:title>
  <prov:wasGeneratedBy rdf:resource="http://cellocad.org/cello2sbol/1"/>
  <dcterms:created>2017-08-09T13:54Z</dcterms:created>
</sbol:ModuleDefinition>
```


FunctionalComponent (Child of ModuleDefinition)



Access URI

<http://sbols.org/v2#public>
<http://sbols.org/v2#private>

Direction URI

<http://sbols.org/v2#in>
<http://sbols.org/v2#out>
<http://sbols.org/v2#inout>
<http://sbols.org/v2#none>

Description

MAY be referred to by remote **MapsTo** objects.
MUST NOT be referred to by remote **MapsTo** objects.

Description

Indicates that it is an input.
Indicates that it is an output.
Indicates that it is both an input and output
Indicates that it is neither an input nor output.

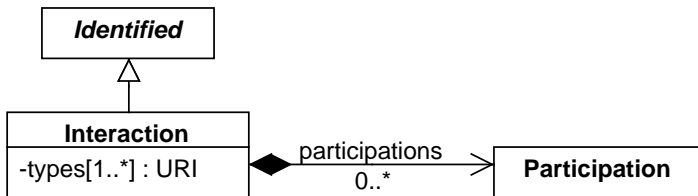
Creating FunctionalComponents for Genetic Production

```
moduleDefinition.createFunctionalComponent (name, AccessType.PUBLIC,  
    componentDefinition.getIdentity(), DirectionType.NONE);
```

```
moduleDefinition.createFunctionalComponent (name+"_protein", AccessType.PUBLIC,  
    proteinComponentDefinition.getIdentity(), DirectionType.NONE);
```

```
<sbol:ModuleDefinition rdf:about="http://cellocad.org/AmtR_protein_production/1">  
  ...  
  <sbol:functionalComponent>  
    <sbol:FunctionalComponent rdf:about="http://cellocad.org/AmtR_protein_production/AmtR/1">  
      <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR"/>  
      <sbol:displayId> AmtR </sbol:displayId>  
      <sbol:version> 1 </sbol:version>  
      <sbol:definition rdf:resource="http://cellocad.org/AmtR/1"/>  
      <sbol:access rdf:resource="http://sbols.org/v2#public"/>  
      <sbol:direction rdf:resource="http://sbols.org/v2#none"/>  
    </sbol:FunctionalComponent>  
  </sbol:functionalComponent>  
  <sbol:functionalComponent>  
    <sbol:FunctionalComponent rdf:about="http://cellocad.org/AmtR_protein_production/AmtR_protein/1">  
      <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR_prot<br>1"/>  
      <sbol:displayId> AmtR_protein </sbol:displayId>  
      <sbol:version> 1 </sbol:version>  
      <sbol:definition rdf:resource="http://cellocad.org/AmtR_protein/1"/>  
      <sbol:access rdf:resource="http://sbols.org/v2#public"/>  
      <sbol:direction rdf:resource="http://sbols.org/v2#none"/>  
    </sbol:FunctionalComponent>  
  </sbol:functionalComponent>  
</sbol:ModuleDefinition>
```

Interaction (Child of ModuleDefinition)



Interaction Type

Inhibition

Stimulation

Biochemical Reaction

Non-Covalent Binding

Degradation

Genetic Production

Control

URI for SystemsBiologyOntology Term

<http://identifiers.org/biomodels.sbo/SBO:0000169>

<http://identifiers.org/biomodels.sbo/SBO:0000170>

<http://identifiers.org/biomodels.sbo/SBO:0000176>

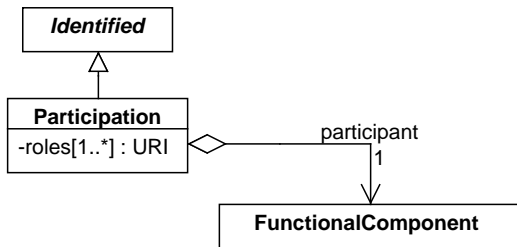
<http://identifiers.org/biomodels.sbo/SBO:0000177>

<http://identifiers.org/biomodels.sbo/SBO:0000179>

<http://identifiers.org/biomodels.sbo/SBO:0000589>

<http://identifiers.org/biomodels.sbo/SBO:0000168>

Participation (Child of Interaction)



Part. Role

Part. Role	URI for SBO Term
Inhibitor	http://identifiers.org/biomodels.sbo/SBO:0000020
Inhibited	http://identifiers.org/biomodels.sbo/SBO:0000642
Stimulator	http://identifiers.org/biomodels.sbo/SBO:0000459
Stimulated	http://identifiers.org/biomodels.sbo/SBO:0000643
Reactant	http://identifiers.org/biomodels.sbo/SBO:0000010
Product	http://identifiers.org/biomodels.sbo/SBO:0000011
Promoter	http://identifiers.org/biomodels.sbo/SBO:0000598
Modifier	http://identifiers.org/biomodels.sbo/SBO:0000019
Modified	http://identifiers.org/biomodels.sbo/SBO:0000644
Template	http://identifiers.org/biomodels.sbo/SBO:0000645

Interaction Types

Inhibition
Inhibition
Stimulation
Stimulation
Non-Covalent Binding, Degradation, Biochemical Reaction
Non-Covalent Binding, Genetic Production, Biochemical Reaction
Inhibition, Stimulation, Genetic Production
Biochemical Reaction, Control
Biochemical Reaction, Control
Genetic Production

Creating Interactions for Genetic Production

```
Interaction interaction = moduleDefinition.createInteraction(name+"_protein_interaction",
    SystemsBiologyOntology.GENETIC_PRODUCTION);
interaction.createParticipation(name, name, SystemsBiologyOntology.TEMPLATE);
interaction.createParticipation(name+"_protein", name+"_protein", SystemsBiologyOntology.PRODUCT);
```

```
<sbol:interaction>
  <sbol:Interaction rdf:about="http://cellocad.org/AmtR_protein_production/AmtR_protein_interact
    <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR_prot
    <sbol:displayId> AmtR_protein_interaction </sbol:displayId>
    <sbol:version> 1 </sbol:version>
    <sbol:type rdf:resource="http://identifiers.org/biomodels.sbo/SBO:0000589"/>
    <sbol:participation>
      <sbol:Participation rdf:about="http://cellocad.org/AmtR_protein_production/AmtR_protein_in
        <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR_
        <sbol:displayId> AmtR </sbol:displayId>
        <sbol:version> 1 </sbol:version>
        <sbol:role rdf:resource="http://identifiers.org/biomodels.sbo/SBO:0000645"/>
        <sbol:participant rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR/1"/>
      </sbol:Participation>
    </sbol:participation>
    <sbol:participation>
      <sbol:Participation rdf:about="http://cellocad.org/AmtR_protein_production/AmtR_protein_in
        <sbol:persistentIdentity rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR_
        <sbol:displayId> AmtR_protein </sbol:displayId>
        <sbol:version> 1 </sbol:version>
        <sbol:role rdf:resource="http://identifiers.org/biomodels.sbo/SBO:0000011"/>
        <sbol:participant rdf:resource="http://cellocad.org/AmtR_protein_production/AmtR_protein
      </sbol:Participation>
    </sbol:participation>
  </sbol:Interaction>
</sbol:interaction>
</sbol:ModuleDefinition>
```

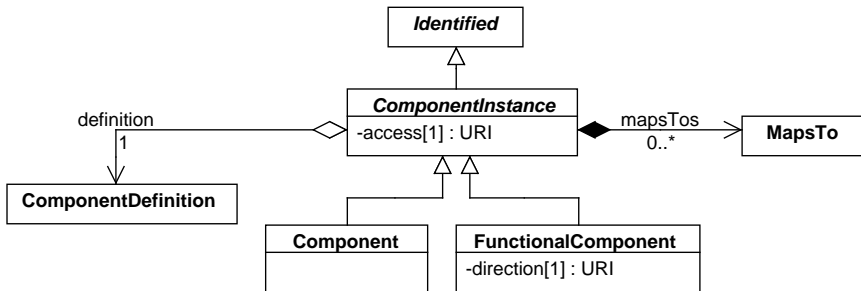
Converting Gate Parts to SBOL ComponentDefinitions

```
for (JSONObject gate : gate_partsArr) {

    String gate_name = (String)gate.get("gate_name");
    ComponentDefinition componentDefinition =
        document.createComponentDefinition(gate_name, version, ComponentDefinition.DNA);
    componentDefinition.setName(gate_name);
    componentDefinition.addRole(SequenceOntology.ENGINEERED_REGION);
    componentDefinition.addWasDerivedFrom(derivedFrom);

    componentDefinition.createAnnotation(new QName(provNS, "wasGeneratedBy", "prov"), activityURI);
    componentDefinition.createAnnotation(new QName(dcTermsNS, "created", "dcTerms"), createdDate);
    componentDefinition.createAnnotation(new QName(celloNS, "family", "cello"),
        (String)gatesMap.get(gate_name).get("system"));
    componentDefinition.createAnnotation(new QName(celloNS, "gate_type", "cello"),
        (String)gatesMap.get(gate_name).get("gate_type"));
    componentDefinition.createAnnotation(new QName(celloNS, "group_name", "cello"),
        (String)gatesMap.get(gate_name).get("group_name"));
    componentDefinition.createAnnotation(new QName(celloNS, "color_hexcode", "cello"),
        (String)gatesMap.get(gate_name).get("color_hexcode"));
}
```

Component (Child of ComponentDefinition)



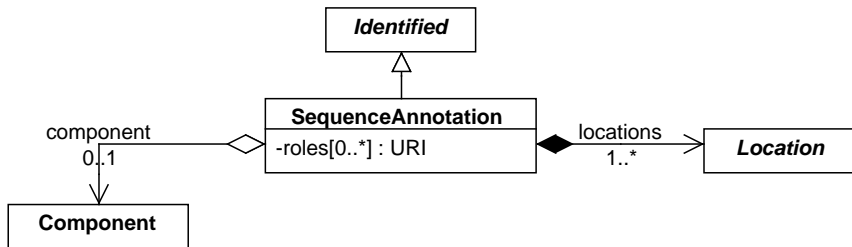
Access URI

<http://sbols.org/v2#public>
<http://sbols.org/v2#private>

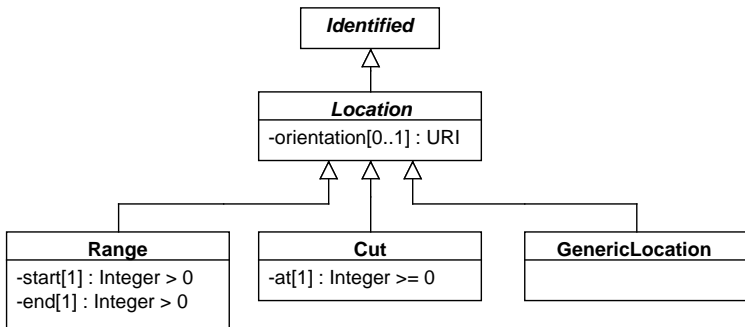
Description

MAY be referred to by remote **MapsTo** objects.
MUST NOT be referred to by remote **MapsTo** objects.

SequenceAnnotation (Child of ComponentDefinition)



Location (Child of SequenceAnnotation)



Orientation URI

<http://sbols.org/v2#inline>

<http://sbols.org/v2#reverseComplement>

Description

The region is inline with the sequence.

The region is on the reverse-complement translation.

Creating Composite Gates Using SequenceAnnotations

```
JSONArray expression_cassettes = (JSONArray) gate.get("expression_cassettes");
String seq = "";

for (Object obj : expression_cassettes) {

    int annotationCount = 0;
    int start = 1;

    JSONObject expression_cassette = (JSONObject) obj;
    JSONArray cassette_parts = (JSONArray) expression_cassette.get("cassette_parts");

    for (Object obj2 : cassette_parts) {

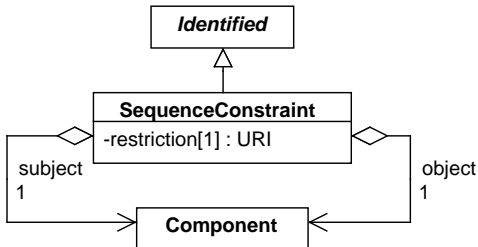
        String partId = (String) obj2;
        ComponentDefinition partComponentDefinition =
            document.getComponentDefinition(partId, version);
        String cass_seq = document.getSequence(partId+"_sequence", version).getElements();
        seq += cass_seq;

        componentDefinition.createComponent(partId, AccessType.PUBLIC, partId, version);

        SequenceAnnotation sa =
            componentDefinition.createSequenceAnnotation("annotation"+annotationCount,
                "range", start, start + cass_seq.length() - 1, OrientationType.INLINE);

        sa.setComponent(partId);
        start += cass_seq.length();
        annotationCount++;
    }
}
```

SequenceConstraint (Child of ComponentDefinition)



Restriction URI

<http://sbols.org/v2#precedes>

<http://sbols.org/v2#sameOrientationAs>

<http://sbols.org/v2#oppositeOrientationAs>

Description

subject **MUST** precede *object* **Component**.

subject & *object* **MUST** have same orientation.

subject & *object* **MUST** have opposite orientations.

Creating Composite Gates Using SequenceConstraints

```
JSONArray expression_cassettes = (JSONArray) gate.get("expression_cassettes");
for (Object obj : expression_cassettes) {

    int constraintCount = 0;
    Component previousComponent = null;
    Component currentComponent = null;

    JSONObject expression_cassette = (JSONObject) obj;
    JSONArray cassette_parts = (JSONArray) expression_cassette.get("cassette_parts");

    for (Object obj2 : cassette_parts) {

        String partId = (String) obj2;
        ComponentDefinition partComponentDefinition =
            document.getComponentDefinition(partId, version);
        currentComponent = componentDefinition.createComponent(partId, AccessType.PUBLIC,
            partId, version);

        if (previousComponent != null) {
            componentDefinition.createSequenceConstraint("constraint"+constraintCount,
                RestrictionType.PRECEDES,
                previousComponent.getIdentity(), currentComponent.getIdentity());
            constraintCount++;
        }

        previousComponent = currentComponent;
    }
}
```

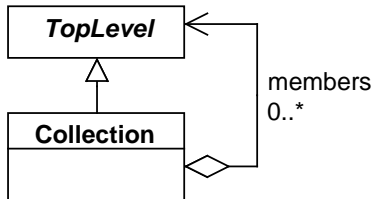
Creating ModuleDefinitions for Inhibition

```
if (partComponentDefinition.getRoles().contains(SequenceOntology.CDS)) {  
  
    String promoter = (String)gate.get("promoter");  
  
    if (document.getModuleDefinition(partId+"_"+promoter+"_repression", version)==null) {  
  
        ModuleDefinition moduleDefinition =  
            document.createModuleDefinition(partId+"_"+promoter+"_repression", version);  
        moduleDefinition.createAnnotation(new QName(provNS,"wasGeneratedBy","prov"), activityURI);  
        moduleDefinition.createAnnotation(new QName(dcTermsNS,"created","dcTerms"), createdDate);  
  
        Interaction interaction =  
            moduleDefinition.createInteraction(partId+"_"+promoter+"_repression",  
                SystemsBiologyOntology.INHIBITION);  
        interaction.createParticipation(partId+"_protein_participation", partId+"_protein",  
            SystemsBiologyOntology.INHIBITOR);  
        interaction.createParticipation(promoter+"_promoter_participation", promoter,  
            SystemsBiologyOntology.INHIBITED);  
    }  
}
```

Creating the Composite Sequence

```
Sequence sequence = document.createSequence(gate_name+"_sequence", version, seq,
    Sequence.IUPAC_DNA);
sequence.setName(gate_name+"_sequence");
sequence.addWasDerivedFrom(derivedFrom);
sequence.createAnnotation(new QName(provNS,"wasGeneratedBy","prov"), activityURI);
sequence.createAnnotation(new QName(dcTermsNS,"created","dcTerms"), createdDate);
componentDefinition.addSequence(sequence);
```

Collection

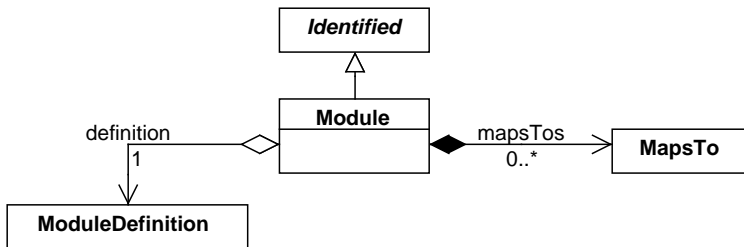


Creating CDS Part Collection

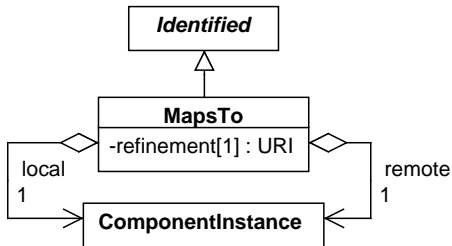
```
Collection cdsCollection = document.createCollection("cdsCollection", version);
for (ComponentDefinition cd : document.getComponentDefinitions()) {
    if (cd.containsRole(SequenceOntology.CDS)) {
        cdsCollection.addMember(cd.getIdentity());
    }
}
```

```
<sbol:Collection rdf:about="http://cellocad.org/cdsCollection/1">
  <sbol:persistentIdentity rdf:resource="http://cellocad.org/cdsCollection"/>
  <sbol:displayId cdsCollection </sbol:displayId>
  <sbol:version> 1 </sbol:version>
  <sbol:member rdf:resource="http://cellocad.org/SrpR/1"/>
  <sbol:member rdf:resource="http://cellocad.org/AmeR/1"/>
  <sbol:member rdf:resource="http://cellocad.org/HlyIIR/1"/>
  <sbol:member rdf:resource="http://cellocad.org/AmtR/1"/>
  <sbol:member rdf:resource="http://cellocad.org/PhlF/1"/>
  <sbol:member rdf:resource="http://cellocad.org/BM3R1/1"/>
  <sbol:member rdf:resource="http://cellocad.org/BetI/1"/>
</sbol:Collection>
```


Module (Child of ModuleDefinition)



MapsTo (Child of ComponentInstance)



Refinement URI

<http://sbols.org/v2#useRemote>

<http://sbols.org/v2#useLocal>

<http://sbols.org/v2#verifyIdentical>

<http://sbols.org/v2#merge>

Description

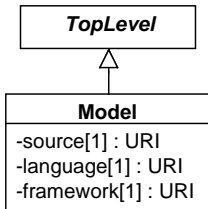
All references MUST dereference to the *remote* CI.

All references MUST dereference to the *local* CI.

The *definition* of the *local* and *remote* MUST be same **CD**.

All references MUST dereference to both objects.

Model



- *source* - URI reference to the source file for the model.
- *language* - URI that species language in which the model is implemented.

Model Language	URI for EDAM Term
-----------------------	--------------------------

SBML	http://identifiers.org/edam/format_2585
------	---

CellML	http://identifiers.org/edam/format_3240
--------	---

BioPAX	http://identifiers.org/edam/format_3156
--------	---

- *framework* - URI that species modeling framework used.

Framework	URI for SBO Term
------------------	-------------------------

Continuous	http://identifiers.org/biomodels.sbo/SBO:0000062
------------	---

Discrete	http://identifiers.org/biomodels.sbo/SBO:0000063
----------	---

Serialization Methods

- SBOLWriter class:

```
SBOLWriter.setKeepGoing(true);
```

```
SBOLWriter.write(document, String OR File OR OutputStream);  
SBOLWriter.write(document, String OR File OR OutputStream, fileType);  
fileType = SBOLDocument.RDF, RDFV1, GENBANK, FASTAformat  
document.write(String OR File OR OutputStream, fileType);
```

```
SBOLWriter.clearErrors();  
SBOLWriter.getNumErrors();  
SBOLWriter.getErrors();
```

- SBOLReader class:

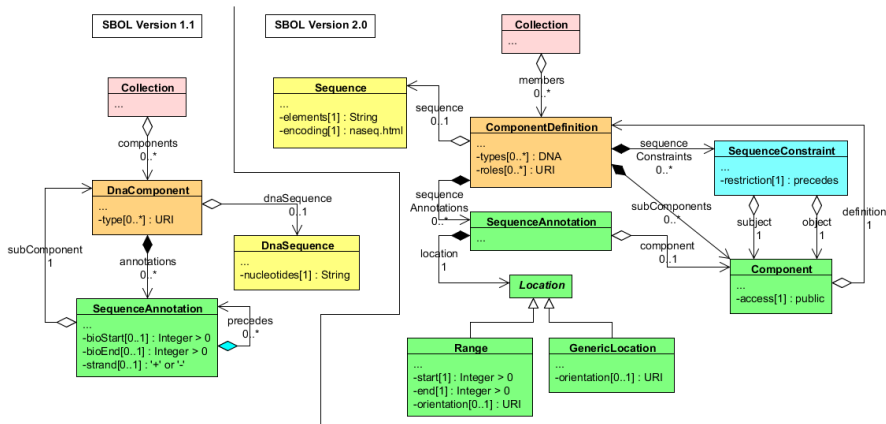
```
SBOLReader.setURIPrefix("http://cellocad.org/");  
SBOLReader.setVersion("1");  
SBOLReader.setKeepGoing(true);
```

```
SBOLReader.getSBOLVersion(String OR File OR InputStream);  
SBOLDocument document = SBOLReader.read(String OR File OR InputStream);  
document.read(String OR File OR InputStream);
```

```
SBOLReader.clearErrors();  
SBOLReader.getNumErrors();  
SBOLReader.getErrors();
```

Conversion

- Supports conversion to/from GenBank and FASTA formats.
- Reads/Writes SBOL 1.1 data files.



Validation

- Many validation rules checked when documents are read.

```
SBOLReader.setKeepGoing(true);
SBOLDocument document = SBOLReader.read(file);
if (SBOLReader.getNumErrors() > 0) {
    for(String error : SBOLReader.getErrors()) {
        System.out.println(error);
    }
}
```

- Remaining validation rules checked by validateSBOL method.

```
boolean complete = true; // Check that all referenced objects are included
boolean compliant = true; // Check that all URIs are compliant
boolean bestPractice = true; // Check best practice validation rules
SBOLValidate.validateSBOL(document, complete, compliant, bestPractice);
if (SBOLValidate.getNumErrors() > 0) {
    for(String error : SBOLValidate.getErrors()) {
        System.out.println(error);
    }
}
```

- Method to compare to SBOLDocuments.

```
SBOLValidate.compareDocuments(file, document, file2, document2);
```

SynBioHub Repository Support

- Uploading SBOL to SynBioHub:

```
SynBioHubFrontend sbh = new SynBioHubFrontend("https://synbiohub.org");
sbh.login("test@dummy.org", "test");
sbh.submit("CelloParts", "1", "Cello_Parts", "These_are_the_Cello_parts", "27034378",
    "", "1", document);
```

- Other SynBioHub functions:

```
public int getCount(String objectType)
public SBOLDocument getSBOL(URI topLevelUri)
public ArrayList<IdentifiedMetadata> getRootCollectionMetadata()
public ArrayList<IdentifiedMetadata> getSubCollectionMetadata(URI parentCollectionUri)
public ArrayList<IdentifiedMetadata> search(SearchQuery query)
public ArrayList<IdentifiedMetadata> getMatchingComponentDefinitionMetadata(String name,
    Set<URI> roles, Set<URI> types, Set<URI> collections, Integer offset, Integer limit)
```

More Information

- `libSBOLj` is open source under the Apache 2.0 License.

- **More information:**

<http://sbolstandard.org/software/libSBOL/java/>.

- Current snapshot on GitHub.
- Latest release (Version 2.1.0) and GitHub and Maven.
- Issue tracker for reporting bugs and feature requests.
- JavaDocs for all public methods.
- A brief getting started tutorial.
- A detailed code example and sample project for a CRISPR circuit.
- Several example code files.

sboljs - Javascript SBOL library

- Developed by James McLaughlin at Newcastle University and myself.
- Prerequisites: node.js and npm.
- Install: `npm install sboljs`

Creating an SBOL Document

```
var SBOLDocument = require('sboljs');  
...  
var sbol = new SBOLDocument();
```

Creating a GenericTopLevel

```
var actVersion = 1;
var actDisplayId = 'cello2sbol';
var actPersistentIdentity = urlprefix + actDisplayId;
var activityURI = actPersistentIdentity + '/' + actVersion;

const activity = sbol.genericTopLevel(activityURI, provNS + 'Activity');
activity.version = actVersion;
activity.displayId = actDisplayId;
activity.name = 'Cello UCF to SBOL conversion';
activity.description = 'Conversion of the Cello UCF parts and metadata to SBOL2';
activity.addStringAnnotation(dcNS + 'creator', 'Prashant Vaidyanathan');
activity.addStringAnnotation(dcNS + 'creator', 'Chris J. Myers');
// TODO: Need to change to PROV endedAtTime
activity.addStringAnnotation('http://purl.org/dc/terms/created', today.toISOString() + '');
activity.persistentIdentity = actPersistentIdentity;
activity.uri = activityURI;
```

Parsing Cello UCF JSON File

```
var ucf = JSON.parse(fs.readFileSync(ucfFilepath) + '');

var gate_partsArr = [];
var partsMap = {};
var partsSBOL = {}
var gatesMap = {};
...

//Go through the UCF file to store collections in local data structures (arrays and maps)
ucf.forEach(function (collection) {

    switch (collection.collection) {
        case 'parts':
            //partsArr.push(collection);
            partsMap[collection.name] = collection;
            break;
        case 'gate_parts':
            gate_partsArr.push(collection);
            break;
        case 'gates':
            gatesMap[collection.gate_name] = collection;
            break;
        ...
    }
}, this);
```

Converting Parts to SBOL (1)

```
Object.keys(partsMap).forEach(function (partkey) {
  var part = partsMap[partkey];
  var partName = part.name;

  if (!(partName in partsSBOL)) {
    const componentDefinition = sbol.componentDefinition()
    componentDefinition.version = version;
    componentDefinition.displayId = part.name;
    componentDefinition.name = part.name;
    componentDefinition.persistentIdentity = urlprefix + componentDefinition.displayId;
    componentDefinition.uri = componentDefinition.persistentIdentity + '/' +
      componentDefinition.version;
    componentDefinition.wasDerivedFrom = derivedFrom;
    componentDefinition.addUriAnnotation(provNS + 'wasGeneratedBy', activityURI);
    componentDefinition.addRole(getPartType(part.type));
    componentDefinition.addType(SBOLDocument.terms.dnaRegion);

    const sequence = sbol.sequence()
    sequence.displayId = part.name + '_sequence';
    sequence.name = part.name + '_sequence';
    sequence.version = version;
    sequence.elements = part.dnasequence;
    sequence.persistentIdentity = urlprefix + sequence.displayId;
    sequence.uri = sequence.persistentIdentity + '/' + sequence.version;
    sequence.wasDerivedFrom = derivedFrom;
    sequence.encoding = SBOLDocument.terms.dnaSequence;
    sequence.addUriAnnotation(provNS + 'wasGeneratedBy', activityURI);
    componentDefinition.addSequence(sequence)
  }
});
```

Converting Parts to SBOL (2)

```
// Create a Protein
if (part.type === 'cds') {
  const proteinComponentDefinition = sbol.componentDefinition();
  proteinComponentDefinition.version = version;
  proteinComponentDefinition.displayId = componentDefinition.displayId + '_protein';
  proteinComponentDefinition.name = componentDefinition.displayId + '_protein';
  proteinComponentDefinition.persistentIdentity = urlprefix +
    proteinComponentDefinition.displayId;
  proteinComponentDefinition.uri = proteinComponentDefinition.persistentIdentity + '/' +
    proteinComponentDefinition.version;
  proteinComponentDefinition.addType(SBOLDocument.terms.protein);
  proteinComponentDefinition.addUriAnnotation('prov:wasGeneratedBy', activityURI);
  partsSBOL[partName + '_protein'] = proteinComponentDefinition.uri;

  const moduleDefinition = sbol.moduleDefinition();
  moduleDefinition.name = proteinComponentDefinition.displayId + '_production';
  moduleDefinition.version = version;
  moduleDefinition.displayId = proteinComponentDefinition.displayId + '_production';
  moduleDefinition.persistentIdentity = urlprefix + moduleDefinition.displayId;
  moduleDefinition.uri = moduleDefinition.persistentIdentity + '/' + moduleDefinition.version;
  moduleDefinition.addUriAnnotation('prov:wasGeneratedBy', activityURI);
}
```

Converting Parts to SBOL (4)

```
//Functional Component for Coding sequence
const functionalComponentCDS = sbol.functionalComponent();
functionalComponentCDS.version = version;
functionalComponentCDS.displayId = componentDefinition.displayId + '_functionalComponent';
functionalComponentCDS.name = componentDefinition.name + '_functionalComponent';
functionalComponentCDS.persistentIdentity = moduleDefinition.persistentIdentity + '/' +
    functionalComponentCDS.displayId;
functionalComponentCDS.uri = functionalComponentCDS.persistentIdentity + '/' +
    functionalComponentCDS.version;
functionalComponentCDS.definition = componentDefinition;

//Functional Component for Protein
const functionalComponentProt = sbol.functionalComponent();
functionalComponentProt.version = version;
functionalComponentProt.displayId = proteinComponentDefinition.displayId + '_functionalComponent';
functionalComponentProt.name = proteinComponentDefinition.name + '_functionalComponent';
functionalComponentProt.persistentIdentity = moduleDefinition.persistentIdentity + '/' +
    functionalComponentProt.displayId;
functionalComponentProt.uri = functionalComponentProt.persistentIdentity + '/' +
    functionalComponentProt.version;
functionalComponentProt.definition = proteinComponentDefinition;
```

Converting Parts to SBOL (5)

```
//CDS to Protein interaction
const interaction = sbol.interaction();
interaction.displayId = proteinComponentDefinition.displayId + '_interaction';
interaction.name = interaction.displayId;
interaction.version = version;
interaction.persistentIdentity = moduleDefinition.persistentIdentity + '/' +
    interaction.displayId;
interaction.uri = interaction.persistentIdentity + '/' + interaction.version;
interaction.addType(productionSBO);

const participationCDS = sbol.participation();
participationCDS.version = version;
participationCDS.name = componentDefinition.displayId + '_participation';
participationCDS.displayId = componentDefinition.displayId + '_participation';
participationCDS.persistentIdentity = interaction.persistentIdentity + '/' +
    participationCDS.displayId;
participationCDS.uri = participationCDS.persistentIdentity + '/' + participationCDS.version;
participationCDS.addRole(templateSBO);
participationCDS.participant = functionalComponentCDS;

const participationProt = sbol.participation();
participationProt.version = version;
participationProt.name = proteinComponentDefinition.displayId + '_participation';
participationProt.displayId = proteinComponentDefinition.displayId + '_participation';
participationProt.persistentIdentity = interaction.persistentIdentity + '/' +
    participationProt.displayId;
participationProt.uri = participationProt.persistentIdentity + '/' + participationProt.version;
participationProt.addRole(productSBO);
participationProt.participant = functionalComponentProt;

interaction.addParticipation(participationCDS);
interaction.addParticipation(participationProt);
```


Converting Parts to SBOL (6)

```
    moduleDefinition.addFunctionalComponent (functionalComponentCDS);
    moduleDefinition.addFunctionalComponent (functionalComponentProt);
    moduleDefinition.addInteraction (interaction);
}
componentDefinition.addStringAnnotation ('http://purl.org/dc/terms/created',
    datecreated.toISOString() + '');
var uriVal = componentDefinition.uri;
partsSBOL[partName] = uriVal;
}
}, this);
```

Converting Gate Parts to SBOL (1)

```
gate_partsArr.forEach(function (gpart) {  
  var gpartName = gpart.gate_name;  
  const componentDefinition = sbol.componentDefinition();  
  componentDefinition.version = version;  
  componentDefinition.displayId = gpartName;  
  componentDefinition.name = gpartName;  
  componentDefinition.persistentIdentity = urlprefix + componentDefinition.displayId;  
  componentDefinition.uri = componentDefinition.persistentIdentity + '/' +  
    componentDefinition.version;  
  componentDefinition.wasDerivedFrom = derivedFrom;  
  componentDefinition.addUriAnnotation(provNS + 'wasGeneratedBy', activityURI);  
  componentDefinition.addUriAnnotation(tetR_family, gatesMap[gpartName].system);  
  componentDefinition.addUriAnnotation(group_name, gatesMap[gpartName].group_name);  
  componentDefinition.addUriAnnotation(gate_type, gatesMap[gpartName].gate_type );  
  componentDefinition.addUriAnnotation(color_hexcode, gatesMap[gpartName].color_hexcode);  
}
```

Converting Gate Parts to SBOL (2)

```
gpart.expression_cassettes.forEach(function (expression_cassettesArr) {  
  
  var seq = "";  
  var annotationCount = 0;  
  var start = 1;  
  
  expression_cassettesArr.cassette_parts.forEach(function (cassette) {  
  
    const component = sbol.component();  
    component.version = version;  
    component.displayId = cassette;  
    component.name = cassette;  
    component.persistentIdentity = componentDefinition.persistentIdentity + '/' +  
      component.displayId;  
    component.uri = component.persistentIdentity + '/' + component.version;  
    component.definition = sbol.lookupURI(partsSBOL[cassette]);  
    componentDefinition.addComponent(component);  
  
  });  
  
});
```

Converting Gate Parts to SBOL (3)

```
var cass_seq = partsMap[cassette].dnasequence;
seq += cass_seq;

const sa = sbol.sequenceAnnotation();
sa.displayId = 'annotation' + annotationCount;
annotationCount++;
sa.name = cassette;
sa.version = version;
sa.persistentIdentity = componentDefinition.persistentIdentity + '/' + sa.displayId;
sa.uri = sa.persistentIdentity + '/' + sa.version;
sa.component = component;
sa.description = partsMap[cassette].type;

const range = sbol.range();
range.displayId = 'range';
range.persistentIdentity = sa.persistentIdentity + '/' + range.displayId;
range.version = version;
range.uri = range.persistentIdentity + '/' + range.version;
range.start = start;
var end = start + cass_seq.length - 1;
range.end = end;
range.orientation = 'http://sbols.org/v2#inline';

sa.addLocation(range);
componentDefinition.addSequenceAnnotation(sa);
```

Converting Gate Parts to SBOL (4)

```
//Create Protein - Promoter repression Module definition
if (partsMap[cassette].type === 'cds') {
  if (!(cassette in moduleDefnMap)) {
    const moduleDefinition = sbol.moduleDefinition();
    moduleDefinition.version = version;
    moduleDefinition.name = cassette + '_' + gpart.promoter + '_repression';
    moduleDefinition.displayId = cassette + '_' + gpart.promoter + '_repression';
    moduleDefinition.persistentIdentity = urlprefix + moduleDefinition.displayId;
    moduleDefinition.uri = moduleDefinition.persistentIdentity + '/' +
      moduleDefinition.version;
    moduleDefinition.addUriAnnotation('provNS + 'wasGeneratedBy', activityURI);

    const functionalComponentCDS = sbol.functionalComponent();
    functionalComponentCDS.version = version;
    functionalComponentCDS.name = cassette + '_protein_functionalComponent';
    functionalComponentCDS.displayId = cassette + '_protein_functionalComponent';
    functionalComponentCDS.persistentIdentity = moduleDefinition.persistentIdentity + '/' +
      functionalComponentCDS.displayId;
    functionalComponentCDS.uri = functionalComponentCDS.persistentIdentity + '/' +
      functionalComponentCDS.version;
    functionalComponentCDS.definition = sbol.lookupURI(partsSBOL[cassette + '_protein']);

    const functionalComponentProm = sbol.functionalComponent();
    functionalComponentProm.version = version;
    functionalComponentProm.name = gpart.promoter + '_functionalComponent';
    functionalComponentProm.displayId = gpart.promoter + '_functionalComponent';
    functionalComponentProm.persistentIdentity = moduleDefinition.persistentIdentity + '/' +
      functionalComponentProm.displayId;
    functionalComponentProm.uri = functionalComponentProm.persistentIdentity + '/' +
      functionalComponentProm.version;
    functionalComponentProm.definition = sbol.lookupURI(partsSBOL[gpart.promoter]);
```

Converting Gate Parts to SBOL (5)

```
//Protein - Promoter Interaction
const interaction = sbol.interaction();
interaction.version = version;
interaction.name = cassette + '_' + gpart.promoter + '_interaction';
interaction.displayId = cassette + '_' + gpart.promoter + '_interaction';
interaction.persistentIdentity = moduleDefinition.persistentIdentity + '/' +
    interaction.displayId;
interaction.uri = interaction.persistentIdentity + '/' + interaction.version;
interaction.addType(inhibitionSO);

//Protein participation
const participantProt = sbol.participation();
participantProt.version = version;
participantProt.name = cassette + '_protein_participation';
participantProt.displayId = cassette + '_protein_participation';
participantProt.persistentIdentity = interaction.persistentIdentity + '/' +
    participantProt.displayId;
participantProt.uri = participantProt.persistentIdentity + '/' + participantProt.version;
participantProt.addRole(inhibitorSO);
participantProt.participant = functionalComponentCDS;

//Promoter participation
const participantProm = sbol.participation();
participantProm.version = version;
participantProm.name = gpart.promoter + '_participation';
participantProm.displayId = gpart.promoter + '_participation';
participantProm.persistentIdentity = interaction.persistentIdentity + '/' +
    participantProm.displayId;
participantProm.uri = participantProm.persistentIdentity + '/' + participantProm.version;
participantProm.addRole(inhibitedSO);
participantProm.participant = functionalComponentProm;
```

Converting Gate Parts to SBOL (6)

```
interaction.addParticipation(participantProt);
interaction.addParticipation(participantProm);

moduleDefinition.addFunctionalComponent(functionalComponentCDS);
moduleDefinition.addFunctionalComponent(functionalComponentProm);
moduleDefinition.addInteraction(interaction);

moduleDefnMap[cassette] = gpart.promoter;
}
}

start += cass_seq.length;

});
```

Converting Gate Parts to SBOL (7)

```
const sequence = sbol.sequence()
sequence.displayId = gpartName + '_sequence';
sequence.name = gpartName + '_sequence';
sequence.version = version;
sequence.elements = seq;
sequence.persistentIdentity = urlprefix + sequence.displayId;
sequence.uri = sequence.persistentIdentity + '/' + sequence.version;
sequence.wasDerivedFrom = derivedFrom;
sequence.encoding = SBOLDocument.terms.dnaSequence;
sequence.addUriAnnotation('prov:ns + 'wasGeneratedBy', activityURI);
componentDefinition.addSequence(sequence);

}, this);

componentDefinition.addType(SBOLDocument.terms.dnaRegion);
componentDefinition.addRole(gate_parts_so);
componentDefinition.addStringAnnotation('http://purl.org/dc/terms/created', datecreated.toISOString());
}, this);
```


Writing the SBOL File

```
var fs = require('fs');

fs.writeFile(resultSBOL, sbol.serializeXML(), function (err) {
  if (err) {
    return console.log(err);
  }

  console.log("Cello_SBOL_file_created!");
});
```

Conclusion (SBOL Compliant Software)

- Can either support all classes or only its structural subset.
- Can support import of SBOL, export of SBOL, or both (lossy/lossless).
- SBOL Test Suite is currently under development here:
<https://github.com/SynBioDex/SBOLTestSuite>
- Validate SBOL files generated using the SBOL Validator found here:
<http://www.async.ece.utah.edu/sbol-validator/>
- Report SBOL-compliant software to the SBOL editors
(editors@sbolstandard.org), so it can be listed on the SBOL website.