## Contents

1 Introduction 3

2 Installation 5
   2.1 Using Pip 5
   2.2 Using Python 5
   2.3 Using Installer for Windows 6
   2.4 For Linux Users 6

3 Testing pySBOL 7

4 Getting Started with SBOL 9
   4.1 Creating an SBOL Document 9
   4.2 Creating SBOL Data Objects 11
   4.3 Using Ontology Terms for Attribute Values 12
   4.4 Adding and Getting Objects from a Document 13
   4.5 Getting, Setting, and Editing Attributes 13
   4.6 Creating, Adding and Getting Child Objects 14
   4.7 Creating and Editing Reference Properties 15
   4.8 Iterating and Indexing List Properties 15
   4.9 Searching a Document 16
   4.10 Copying Documents and Objects 16
   4.11 Converting To and From Other Sequence Formats 17
   4.12 Creating Biological Designs 17

5 Biological Parts Repositories 19
   5.1 Re-using Genetic Parts From Online Repositories 19
   5.2 Searching Part Repos 20
   5.3 Submitting Designs to a Repo 21

6 Computer-aided Design for Synthetic Biology 23
   6.1 Design Abstraction 23
   6.2 Hierarchical DNA Assembly 23
   6.3 Editing a Primary Structure 24
   6.4 Sequence Assembly 24
   6.5 Genome Integration 25
   6.6 Full Example Code 25
7  Design-Build-Test-Learn Workflows 27
8  API 31
9  Indices and tables 217
Python Module Index 219
Index 221
pySBOL is a SWIG-Python wrapper around libSBOL, a module for reading, writing, and constructing genetic designs according to the standardized specifications of the Synthetic Biology Open Language (SBOL).
pySBOL provides Python interfaces and their implementation for Synthetic Biology Open Language (SBOL). The current version of pySBOL implements SBOL Core Specification 2.1.0. The library provides an API to work with SBOL objects, the functionality to read GenBank, FASTA, and SBOL version 1 and 2 documents as XML/RDF files, to write GenBank, FASTA, and SBOL version 1 and 2 documents, and to validate the correctness of SBOL 2 documents. This is a Python binding for C/C++ based libSBOL. Currently, pySBOL supports Python version 2.7 and 3.6 only. pySBOL is made freely available under the Apache 2.0 license.

To install, go to Installation Page.

- The current snapshot of pySBOL is available on GitHub.
- Any problems or feature requests for pySBOL should be reported on the GitHub issue tracker.
- An overview of pySBOL can be found here.
- For further information about the pySBOL library, its implementation, or its usage, please feel free to contact the libSBOL team.

pySBOL is brought to you by Bryan Bartley, Kiri Choi, and SBOL Developers.

Current support for the development of pySBOL is generously provided by the NSF through the Synthetic Biology Open Language Resource collaborative award.
Currently, we support Python 2.7 and Python 3.6 for both 32 bit and 64 bit architecture. Python by default comes with package manager. Follow the steps below to install pySBOL. If you have Windows, and would like to try our Windows binary installers, check Using Installer for Windows section.

2.1 Using Pip

pySBOL is available for Windows and Mac OSX via PyPI, which is the simplest method to obtain pySBOL. To install pySBOL using pip, run following line on console:

```
pip install pysbol
```

If you encounter permission errors on Mac OSX, you may install pysbol to your user site-packages directory as follows:

```
pip install pysbol --user
```

Or alternatively, you may install as a super-user:

```
sudo -H pip install pysbol
```

To update pySBOL using pip, run:

```
pip install -U pysbol
```

2.2 Using Python

1 - Download the source code of latest release here and extract it. If you would like to try out our latest snapshot, use git and type following command in the console or terminal which will clone the source under pysbol folder.

```
git clone https://github.com/SynBioDex/pysbol.git
```

2 - Open your console or terminal. Go to package’s root directory and Run the installer script by using the following command line. This will install pySBOL2 to the Python release associated with the console or terminal you are using.
python setup.py install

If you are having problems, make sure your console/terminal is associated with the right Python environment you wish to use.

3 - Test the pySBOL by importing it in Python.

import sbol

If you have trouble importing the module with the setup script, check to see if there are multiple Python installations on your machine and also check the output of the setup script to see which version of Python is the install target. You can also test the module locally from inside the Mac_OSX/sbol or Win_32/sbol folders.

2.3 Using Installer for Windows

We provide binary installers for Windows users only. Simply download the installers and execute it to install it. Installer will look for your local Python distributions.

Be sure to use the installers with the same Python version and architecture with the one installed in your local machine!

2.4 For Linux Users

Currently, Linux users should build pySBOL from source through libSBOL. Go to libSBOL installation page and follow the instructions for Debian/Ubuntu.
Testing pySBOL

pySBOL comes with a testing function to check the integrity of the library. To run the tester, simply execute the following command.

```python
import sbol
sbol.testSBOL()
```

The output tells you whether certain test has been passed or not.

```plaintext
testAddComponentDefinition (sbol.unit_tests.TestComponentDefinitions) ... ok
testCDDisplayId (sbol.unit_tests.TestComponentDefinitions) ... ok
testRemoveComponentDefinition (sbol.unit_tests.TestComponentDefinitions) ... ok
testAddSequece (sbol.unit_tests.TestSequences) ... ok
testRemoveSequence (sbol.unit_tests.TestSequences) ... ok
testSeqDisplayId (sbol.unit_tests.TestSequences) ... ok
testSequenceElement (sbol.unit_tests.TestSequences) ... ok
testDiscard (sbol.unit_tests.TestMemory) ... ok
```
This beginner’s guide introduces the basic principles of pySBOL for new users. Most of the examples discussed in this guide are excerpted from the example script. The objective of this documentation is to familiarize users with the basic patterns of the API. For more comprehensive documentation about the API, refer to documentation about specific classes and methods.

The class structure and data model for the API is based on the Synthetic Biology Open Language. For more detail about the SBOL standard, visit sbolstandard.org or refer to the specification document. This document provides diagrams and description of all the standard classes and properties that comprise SBOL.

4.1 Creating an SBOL Document

In a previous era, engineers might sit at a drafting board and draft a design by hand. The engineer’s drafting sheet in pySBOL is called a Document. The Document serves as a container, initially empty, for SBOL data objects which represent elements of a biological design. Usually the first step is to construct a Document in which to put your objects. All file I/O operations are performed on the Document. The read and write methods are used for reading and writing files in SBOL format.

```python
>>> doc = Document()
>>> doc.read('crispr_example.xml')
>>> doc.write('crispr_example_out.xml')
```

Reading a Document will wipe any existing contents clean before import. However, you can import objects from multiple files into a single Document object using Document.append(). This can be advantageous when you want to integrate multiple objects from different files into a single design. This kind of data integration is an important and useful feature of SBOL.

A Document may contain different types of SBOL objects, including ComponentDefinitions, ModuleDefinitions, Sequences, and Models. These objects are collectively referred to as TopLevel objects because they can be referenced directly from a Document. The total count of objects contained in a Document is determined using the len function. To view an inventory of objects contained in the Document, simply print it.
Each SBOL object in a Document is uniquely identified by a special string of characters called a Uniform Resource Identifier (URI). A URI is used as a key to retrieve objects from the Document. To see the identities of objects in a Document, iterate over them using a Python iterator.

```python
>>> for obj in doc:
...     print(obj)
...
http://sbols.org/CRISPR_Example/mKate_seq/1.0.0
http://sbols.org/CRISPR_Example/gRNA_b_nc/1.0.0
http://sbols.org/CRISPR_Example/mKate_cds/1.0.0
...
```

These objects are sorted into object stores based on the type of object. For example to view ComponentDefinition objects specifically, iterate through the Document.componentDefinitions store:

```python
Similarly, you can iterate through Document.moduleDefinitions, Document.sequences, Document.models, or any top level object. The last type of object, Annotation Objects is a special case which will be discussed later.
```

These URIs are said to be sbol-compliant. An sbol-compliant URI consists of a scheme, a namespace, a local identifier (also called a displayId), and a version number. In this tutorial, we use URIs of the type http://sbols.org/CRISPR_Example/my_obj/1.0.0, where the scheme is indicated by http://, the namespace is http://sbols.org/CRISPR_Example, the local identifier is my_obj, and the version is 1.0.0. SBOL-compliant URIs enable shortcuts that make the pySBOL API easier to use and are enabled by default. However, users are not required to use sbol-compliant URIs if they don’t want to, and this option can be turned off.

Based on our inspection of objects contained in the Document above, we can see that these objects were all created in the namespace http://sbols.org/CRISPR_Example. Thus, in order to take advantage of SBOL-compliant URIs, we set an environment variable that configures this namespace as the default. In addition we set some other configuration options.

```python
>>> setHomespace('http://sbols.org/CRISPR_Example')
```

Setting the Homespace has several advantages. It simplifies object creation and retrieval from Documents. In addition, it serves as a way for a user to claim ownership of new objects. Generally users will want to specify a Homespace that
corresponds to their organization’s web domain.

### 4.2 Creating SBOL Data Objects

Biological designs can be described with SBOL data objects, including both structural and functional features. The principle classes for describing the structure and primary sequence of a design are `ComponentDefinitions`, `Components`, `Sequences`, and `SequenceAnnotations`. The principle classes for describing the function of a design are `ModuleDefinitions`, `Modules`, `Interactions`, and `Participations`. Other classes such as `Design`, `Build`, `Test`, `Analysis`, `Activity`, and `Plan` are used for managing workflows.

In the official SBOL specification document, classes and their properties are represented as box diagrams. Each box represents an SBOL class and its attributes. Following is an example of the diagram for the `ComponentDefinition` class which will be referred to in later sections. These class diagrams follow conventions of the Unified Modeling Language.

As introduced in the previous section, SBOL objects are identified by a uniform resource identifier (URI). When a new object is constructed, the user must assign a unique identity. The identity is ALWAYS the first argument supplied to the constructor of an SBOL object. Depending on which configuration options for pySBOL are specified, different algorithms are applied to form the complete URI of the object. The following examples illustrate these different configuration options.

The first set of configuration options demonstrates ‘open-world’ mode, which means that URIs are explicitly specified in full by the user, and the user is free to use whatever convention or conventions they want to form URIs. Open-world configuration can be useful sometimes when integrating data objects derived from multiple files or web resources, because it makes no assumptions about the format of URIs.

```python
>>> setHomespace('')
>>> Config.setOption('sbol_compliant_uris', False)
>>> Config.setOption('sbol_typed_uris', False)
>>> crispr_template = ModuleDefinition('http://sbols.org/CRISPR_Example/CRISPR_Template')
>>> print(crispr_template)
http://sbols.org/CRISPR_Example/CRISPR_Template
```

The second set of configuration options demonstrates use of a default namespace for constructing URIs. The advantage of this approach is simply that it reduces repetitive typing. Instead of typing the full namespace for a URI every time an object is created, the user simply specifies the local identifier. The local identifier is appended to the namespace. This is a handy shortcut especially when working interactively in the Python interpreter.

```python
>>> setHomespace('http://sbols.org/CRISPR_Example/')
>>> Config.setOption('sbol_compliant_uris', False)
```

(continues on next page)
The third set of configuration options demonstrates SBOL-compliant mode. In this example, a version number is appended to the end of the URI. Additionally, when operating in SBOL-compliant mode, the URIs of child objects are algorithmically constructed according to automated rules (not shown here).

The final example demonstrates typed URIs. When this option is enabled, the type of SBOL object is included in the URI. Typed URIs are useful because sometimes the user may want to re-use the same local identifier for multiple objects. Without typed URIs this may lead to collisions between non-unique URIs. This option is enabled by default, but the example file CRISPR_example.py does not use typed URIs, so for all the examples in this guide this option is assumed to be disabled.

Constructors for SBOL objects follow a fairly predictable pattern. The first argument is ALWAYS the identity of the object. Other arguments may follow, depending on in the SBOL class has required attributes. Attributes are required if the specification says they are. In a UML diagram, required fields are indicated as properties with a cardinality of 1 or more. For example, a ComponentDefinition (see the UML diagram above) has only one required field, types, which specifies one or more molecular types for a component. Required fields SHOULD be specified when calling a constructor. If they are not, they will be assigned default values. The following creates a protein component. If the BioPAX term for protein were not specified, then the constructor would create a ComponentDefinition of type BIOPAX_DNA by default.

4.3 Using Ontology Terms for Attribute Values

Notice the ComponentDefinition.types attribute is specified using a predefined constant. The ComponentDefinition.types property is one of many SBOL attributes that uses ontology terms as property values. The ComponentDefinition.types property uses the BioPax ontology <https://bioportal.bioontology.org/ontologies/BP/?p=classes&conceptid=root> to be specific. Ontologies are standardized, machine-readable vocabularies that categorize concepts within a domain of scientific study. The SBOL 2.0 standard unifies many different ontologies into a high-level, object-oriented model.
Ontology terms also take the form of Uniform Resource Identifiers. Many commonly used ontological terms are built-in to pySBOL as predefined constants. If an ontology term is not provided as a built-in constant, its URI can often be found by using an ontology browser tool online. Browse Sequence Ontology terms here <http://www.sequenceontology.org/browser/obob.cgi> and Systems Biology Ontology terms here. While the SBOL specification often recommends particular ontologies and terms to be used for certain attributes, in many cases these are not rigid requirements. The advantage of using a recommended term is that it ensures your data can be interpreted or visualized by other applications that support SBOL. However, in many cases an application developer may want to develop their own ontologies to support custom applications within their domain.

The following example illustrates how the URIs for ontology terms can be easily constructed, assuming they are not already part of pySBOL’s built-in ontology constants.

```python
>>> SO_ENGINEERED_FUSION_GENE = SO + '0000288'  # Sequence Ontology term
>>> SO_ENGINEERED_FUSION_GENE
'http://identifiers.org/so/SO:0000288'
>>> SBO_DNA_REPLICATION = SBO + '0000204'  # Systems Biology Ontology term
>>> SBO_DNA_REPLICATION
'http://identifiers.org/biomodels.sbo/SBO:0000204'
```

### 4.4 Adding and Getting Objects from a Document

In some cases a developer may want to use SBOL objects as intermediate data structures in a computational biology workflow. In this case the user is free to manipulate objects independently of a Document. However, if the user wishes to write out a file with all the information contained in their object, they must first add it to the Document. This is done using `add` methods. The names of these methods follow a simple pattern, simply “add” followed by the type of object.

```python
>>> doc.addModuleDefinition(crispr_template)
>>> doc.addComponentDefinition(cas9)
```

Objects can be retrieved from a Document by using `get` methods. These methods ALWAYS take the object’s full URI as an argument.

```python
>>> crispr_template = doc.getModuleDefinition('http://sbols.org/CRISPR_Example/CRISPR_Template/1.0.0')
>>> cas9 = doc.getComponentDefinition('http://sbols.org/CRISPR_Example/cas9_generic/1.0.0')
```

When working interactively in a Python environment, typing long form URIs can be tedious. Operating in SBOL-compliant mode allows the user an alternative means to retrieve objects from a Document using local identifiers.

```python
>>> Config.setOption('sbol_compliant_uris', True)
>>> Config.setOption('sbol_typed_uris', False)
>>> crispr_template = doc.moduleDefinitions['CRISPR_Template']
>>> cas9 = doc.componentDefinitions['cas9_generic']
```

### 4.5 Getting, Setting, and Editing Attributes

The attributes of an SBOL object can be accessed like other Python class objects, with a few special considerations. For example, to get the values of the `displayId` and `identity` properties of any object:

Note that `displayId` gives only the shorthand, local identifier for the object, while the `identity` property gives the full URI.
The attributes above return singleton values. Some attributes, like `ComponentDefinition.roles` and `ComponentDefinition.types` support multiple values. Generally these attributes have plural names. If an attribute supports multiple values, then it will return a list. If the attribute has not been assigned any values, it will return an empty list.

```python
>>> cas9.types
['http://www.biopax.org/release/biopax-level3.owl#Protein']
>>> cas9.roles
[]
```

Setting an attribute follows the ordinary convention for assigning attribute values:

```python
>>> crispr_template.description = 'This is an abstract, template module'
```

To set multiple values:

```python
>>> plasmid = ComponentDefinition('pBBR', BIOPAX_DNA, '1.0.0')
>>> plasmid.roles = [ SO_PLASMID, SO_CIRCULAR ]
```

Although properties such as `types` and `roles` behave like Python lists in some ways, beware that list operations like `append` and `extend` do not work directly on these kind of attributes, due to the nature of the C++ bindings. If you need to append values to an attribute, use the following idiom:

```python
>>> plasmid.roles = [ SO_PLASMID ]
>>> plasmid.roles = plasmid.roles + [ SO_CIRCULAR ]
```

To clear all values from an attribute, set to `None`:

```python
>>> plasmid.roles = None
```

## 4.6 Creating, Adding and Getting Child Objects

Some SBOL objects can be composed into hierarchical parent-child relationships. In the specification diagrams, these relationships are indicated by black diamond arrows. In the UML diagram above, the black diamond indicates that `ComponentDefinitions` are parents of `SequenceAnnotations`. Properties of this type can be modified using the `add` method and passing the child object as the argument.

```python
>>> point_mutation = SequenceAnnotation('PointMutation')
>>> target_promoter.sequenceAnnotations.add(point_mutation)
```

Alternatively, the `create` method captures the construction and addition of the `SequenceAnnotation` in a single function call. The `create` method ALWAYS takes one argument—the URI of the new object. All other values are initialized with default values. You can change these values after object creation, however.

```python
>>> target_promoter.sequenceAnnotations.create('PointMutation')
```

Conversely, to obtain a Python reference to the `SequenceAnnotation` from its identity:

```python
>>> point_mutation = target_promoter.sequenceAnnotations.get('PointMutation')
```

Or equivalently:

```python
>>> point_mutation = target_promoter.sequenceAnnotations['PointMutation']
```
4.7 Creating and Editing Reference Properties

Some SBOL objects point to other objects by way of URI references. For example, ComponentDefinitions point to their corresponding Sequences by way of a URI reference. These kind of properties correspond to white diamond arrows in UML diagrams, as shown in the figure above. Attributes of this type contain the URI of the related object.

```python
>>> eyfp_gene = ComponentDefinition('EYFPGene', BIOPAX_DNA)
>>> seq = Sequence('EYFPSequence', 'atgnntaa', SBOL_ENCODING_IUPAC)
>>> eyfp_gene.sequence = seq
>>> print (eyfp_gene.sequence)
'http://sbols.org/Sequence/EYFPSequence/1.0.0'
```

Note that assigning the `seq` object to the `eyfp_gene.sequence` actually results in assignment of the object’s URI. An equivalent assignment is as follows:

```python
>>> eyfp_gene.sequence = seq.identity
>>> print (eyfp_gene.sequence)
'http://sbols.org/Sequence/EYFPSequence/1.0.0'
```

4.8 Iterating and Indexing List Properties

Some properties can contain multiple values or objects. Additional values can be specified with the add method. In addition you may iterate over lists of objects or values.

```python
# Iterate through objects (black diamond properties in UML)
for p in cas9_complex_formation.participations:
    print(p)
    print(p.roles)

# Iterate through references (white diamond properties in UML)
for role in reaction_participant.roles:
    print(role)
```

Numerical indexing of lists works as well:

```
for i_p in range(0, len(cas9_complex_formation.participations)):
    print(cas9_complex_formation.participations[i_p])
```

4.9 Searching a Document

To see if an object with a given URI is already contained in a Document or other parent object, use the `find` method. Note that `find` function returns the target object cast to its base type which is `SBOLObject`, the generic base class for all SBOL objects. The actual SBOL type of this object, however is `ComponentDefinition`. If necessary the base class can be downcast using the `cast` method.

```
>>> obj = doc.find('http://sbols.org/CRISPR_Example/mKate_gene/1.0.0')
>>> obj
SBOLObject
>>> parseClassName(obj.type)
'ComponentDefinition'
>>> cd = obj.cast(ComponentDefinition)
```
>>> cd
ComponentDefinition

The **find** method is probably more useful as a boolean conditional when the user wants to automatically construct URIs for objects and needs to check if the URI is unique or not. If the object is found, **find** returns an object reference (True), and if the object is not found, it returns None (False). The following code snippet demonstrates a function that automatically generates ComponentDefinitions.

```python
def createNextComponentDefinition(doc, local_id):
    i_cdef = 0
    cdef_uri = getHomespace() + '/%s_%d/1.0.0' %(local_id, i_cdef)
    while doc.find(cdef_uri):
        i_cdef += 1
        cdef_uri = getHomespace() + '/%s_%d/1.0.0' %(local_id, i_cdef)
    doc.componentDefinitions.create(' %s_%d' %(local_id, i_cdef))
```

### 4.10 Copying Documents and Objects

Copying a Document can result in a few different ends, depending on the user’s goal. The first option is to create a simple clone of the original Document. This is shown below in which the user is assumed to have already created a Document with a single ComponentDefinition. After copying, the object in the Document clone has the same identity as the object in the original Document.

```python
>>> for o in doc:
...    print o
...
http://examples.org/ComponentDefinition/cd/1
>>> doc2 = doc.copy()
>>> for o in doc2:
...    print o
...
http://examples.org/ComponentDefinition/cd/1
```

More commonly a user wants to import objects from the target Document into their Homespace. In this case, the user can specify a target namespace for import. Objects in the original Document that belong to the target namespace are copied into the user’s Homespace. Contrast the example above with the following.

```python
>>> setHomespace('http://sys-bio.org')
>>> doc2 = doc.copy('http://examples.org')
>>> for o in doc:
...    print o
...
http://examples.org/ComponentDefinition/cd/1
>>> for o in doc2:
...    print o
...
http://sys-bio.org/ComponentDefinition/cd/1
```

In the examples above, the **copy** method returns a new Document. However, it is possible to integrate the result of multiple **copy** operations into an existing Document.

```python
>>> for o in doc1:
...    print o
```
4.11 Converting To and From Other Sequence Formats

It is possible to convert SBOL to and from other common sequence formats. Conversion is performed by calling the online converter tool, so an internet connection is required. Currently the supported formats are SBOL2, SBOL1, FASTA, GenBank, and GFF3. The following example illustrates how to import and export to these different formats. Note that conversion can be lossy.

```python
>>> doc.exportToFormat('GenBank', 'crispr_example_out.gb')
```

```python
>>> doc.importFromFormat('GenBank', 'crispr_example_out.gb')
```

4.12 Creating Biological Designs

This concludes the basic methods for manipulating SBOL data structures. Now that you’re familiar with these basic methods, you are ready to learn about libSBOL’s high-level design interface for synthetic biology. See SBOL Examples.
CHAPTER 5

Biological Parts Repositories

5.1 Re-using Genetic Parts From Online Repositories

In today’s modern technological society, a variety of interesting technologies can be assembled from “off-the-shelf” components, including cars, computers, and airplanes. Synthetic biology is inspired by a similar idea. Synthetic biologists aim to program new biological functions into organisms by assembling genetic code from off-the-shelf DNA sequences. PySBOL puts an inventory of biological parts at your fingertips.

For example, the iGEM Registry of Standard Biological Parts is an online resource that many synthetic biologists are familiar with. The Registry is an online database that catalogs a vast inventory of genetic parts, mostly contributed by students in the iGEM competition. These parts are now available in SBOL format in the SynBioHub knowledgebase, hosted by Newcastle University. The code example below demonstrates how a programmer can access these data.

The following code example shows how to pull data about biological components from the SynBioHub repository. In order to pull a part, simply locate the web address of that part by browsing the SynBioHub repository online. Alternatively, pySBOL also supports programmatic querying of SynBioHub (see below).

The interface with the SynBioHub repository is represented by a PartShop object. The following code retrieves parts corresponding to promoter, coding sequence (CDS), ribosome binding site (RBS), and transcriptional terminator.

```python
>>> igem = PartShop('https://synbiohub.org')
>>> igem.pull('https://synbiohub.org/public/igem/BBa_R0010/1', doc)
```

Typing full URIs can be tedious. Therefore the PartShop interface provides a shortcut for retrieving objects. It will automatically construct a URI from the PartShop namespace and the part’s displayId. Constrain the above with the following.

```python
>>> igem = PartShop('https://synbiohub.org/public/igem')
>>> igem.pull('BBa_B0032', doc)
>>> igem.pull('BBa_E0040', doc)
>>> igem.pull('BBa_B0012', doc)
```
The pull operation will retrieve ComponentDefinitions and their associated Sequence objects. Note that the objects are copied into the user’s Homespace:

```python
>>> for obj in doc:
    ...   print obj
    ...
http://examples.org/Sequence/BBa_R0010_sequence/1
http://examples.org/Sequence/BBa_B0012_sequence/1
http://examples.org/ComponentDefinition/BBa_E0040/1
http://examples.org/ComponentDefinition/BBa_B0012/1
http://examples.org/Sequence/BBa_E0040_sequence/1
http://examples.org/Activity/igem2sbol/1
http://examples.org/ComponentDefinition/BBa_R0010/1
http://examples.org/ComponentDefinition/BBa_B0032/1
http://examples.org/Sequence/BBa_B0032_sequence/1
```

### 5.2 Searching Part Repos

PySBOL supports three kinds of searches: a general search, an exact search, and an advanced search.

The following query conducts a general search which scans through identity, name, description, and displayId properties for a match to the search text, including partial, case-insensitive matches to substrings of the property value. Search results are returned as a SearchResponse object.

```python
records = igem.search('plasmid')
```

By default, the general search looks only for ComponentDefinitions, and only returns 25 records at a time in order to prevent server overload. The search above is equivalent to the one below, which explicitly specifies which kind of SBOL object to search for, an offset of 0 (explained below), and a limit of 25 records.

```python
records = igem.search('plasmid', SBOL_COMPONENT_DEFINITION, 0, 25)
```

Of course, these parameters can be changed to search for different type of SBOL objects or to return more records. For example, some searches may match a large number of objects, more than the specified limit allows. In this case, it is possible to specify an offset and to retrieve additional records in successive requests. The total number of objects in the repository matching the search criteria can be found using the searchCount method, which has the same call signature as the search method. It is a good idea to put a small delay between successive requests to prevent server overload. The following example demonstrates how to do this. The 100 millisecond delay is implemented using cross-platform C++11 headers chrono and thread. As of the writing of this documentation, this call retrieves 391 records.

```python
import time
records = SearchResponse()
search_term = 'plasmid'
limit = 25
total_hits = igem.searchCount(search_term)
for offset in range(0, total_hits, limit):
    records.extend( igem.search(search_term, SBOL_COMPONENT_DEFINITION, offset, limit) )
    time.sleep(0.1)
```

A SearchResponse object is returned by a query and contains multiple records. Each record contains basic data, including identity, displayId, name, and description fields. It is very important to realize however that the search does not retrieve the complete ComponentDefinition! In order to retrieve the full object, the user must call pull while specifying the target object’s identity.
Records in a `SearchResponse` can be accessed using iterators or numeric indices. The interface for each record behaves exactly like any other SBOL object:

```python
for record in records:
    print( record.identity.get() )
```

The preceding examples concern **general searches**, which scan through an object’s metadata for partial matches to the search term. In contrast, the **exact search** explicitly specifies which property of an object to search, and the value of that property must exactly match the search term. The following **exact search** will search for `ComponentDefinitions` with a role of promoter:

```python
records = igem.search(SO_PROMOTER, SBOL_COMPONENT_DEFINITION, SBOL_ROLES, 0, 25);
```

Finally, the **advanced search** allows the user to configure a search with multiple criteria by constructing a `SearchQuery` object. The following query looks for promoters that have an additional annotation indicating that the promoter is regulated (as opposed to constitutive):

```python
q = SearchQuery();
q['objectType'].set(SBOL_COMPONENT_DEFINITION);
q['limit'].set(25);
q['offset'].set(0);
q['role'].set(SO_PROMOTER);
q['role'].add('http://wiki.synbiohub.org/wiki/Terms/igem#partType/Regulatory');
total_hits = igem.searchCount(q);
records = igem.search(q);
```

## 5.3 Submitting Designs to a Repo

Users can submit their SBOL data directly to a PartShop using the pySBOL API. This is important, so that synthetic biologists may reuse the data and build off each other’s work. Submitting to a repository is also important for reproducing published scientific work. The synthetic biology journal ACS Synthetic Biology now encourages authors to submit SBOL data about their genetically engineered DNA to a repository like SynBioHub. In order to submit to a PartShop remotely, the user must first visit the appropriate website and register. Once the user has established an account, they can then log in remotely using pySBOL.

```python
>>> igem.login('johndoe@example.org', password)
```

Upon submission of a Document to SynBioHub, the Document will be converted to a Collection. Therefore, the Document requires that the `displayId`, `name`, and `description` properties are set prior to submission.

```python
>>> doc.displayId = 'my_collection'
>>> doc.name = 'my collection'
>>> doc.description = 'a description of your collection'
>>> igem.submit(doc)
```

Once uploaded, a new URI for the Collection is generated. This URI follows the pattern `<PART SHOP URI>/USER NAME>/<DOCUMENT DISPLAYID>_collection`. Other TopLevel objects in the Document are also mapped to new URIs. These URIs follow the pattern `<PART SHOP URI>/USER NAME>/<SBOL TYPE>_DISPLAYID`.

After submission, it is possible to attach other types of data files to SBOL objects. This requires the new URI of the target object and a path to the local file on the user’s machine.

```python
>>> igem.attachFile('<PART SHOP URI>/USER NAME>/<SBOL TYPE>_DISPLAYID', '<PATH TO LOCAL FILE>')
```
Likewise, it is possible to download a file attachment.

```python
>>> igem.downloadAttachment('PART SHOP URI/<USER NAME>/<SBOL TYPE>_DISPLAYID', 'PATH TO WRITE')
```
6.1 Design Abstraction

An advantage of the SBOL data format over GenBank is the ability to represent DNA as abstract components without specifying an exact sequence. An abstract design can be used as a template, with sequence information filled in later. In SBOL, a ComponentDefinition represents a biological component whose general function is known while its sequence is currently either unknown or unspecified. The intended function of the component is specified using a descriptive term from the Sequence Ontology (SO), a standard vocabulary for describing genetic parts. As the following example shows, some common SO terms are built in to PySBOL as pre-defined constants (see constants.h). This code example defines the new component as a gene by setting its roles property to the SO term for gene. Other terms may be found by browsing the Sequence Ontology online.

```python
# Construct an abstract design for a gene
gene = ComponentDefinition('gene_example')
gene.roles = SO_GENE
```

Design abstraction is an important engineering principle for synthetic biology. Abstraction enables the engineer to think at a high-level about functional characteristics of a system while hiding low-level physical details. For example, in electronics, abstract schematics are used to describe the function of a circuit, while hiding the physical details of how a printed circuit board is laid out. Computer-aided design (CAD) programs allow the engineer to easily switch back and forth between abstract and physical representations of a circuit. In the same spirit, PySBOL enables a CAD approach for designing genetic constructs and other forms of synthetic biology.

6.2 Hierarchical DNA Assembly

PySBOL also includes methods for assembling biological components (also referred to as biological parts in the synthetic biology literature) into abstraction hierarchies. Abstraction hierarchies are important from an engineering perspective because they allow engineers to assemble complicated systems from more basic components. Abstraction
hierarchies are also important from a biological perspective, because DNA sequences and biological structures in
general exhibit hierarchical organization, from the genome, to operons, to genes, to lower level genetic operators.
The following code assembles an abstraction hierarchy that describes a gene cassette. Note that subcomponents must
belong to a Document in order to be assembled, so a Document is passed as a parameter.

The gene cassette below is composed of genetic subcomponents including a promoter, ribosome binding site (RBS),
coding sequence (CDS), and transcriptional terminator, expressed in SBOL Visual schematic glyphs. The next exam-
ple demonstrates how an abstract design for this gene is assembled from its subcomponents.

```python
gene.assemblePrimaryStructure([r0010, b0032, e0040, b0012], doc)
```

After creating an abstraction hierarchy, it is then possible to iterate through an object’s primary structure of compo-
nents:

```python
for component_definition in gene.getPrimaryStructure():
    print (component_definition.identity)
```

This returns a list of ComponentDefinitions arranged in their primary sequence. Occasionally it is also helpful to get
Components arranged in their primary sequence as well. Note that the example below produces the same output as the
example above, and may be helpful for understanding the relationship between Components and ComponentDefi-
nitions.

```python
for component in gene.components:
    print (component.definition)
```

### 6.3 Editing a Primary Structure

Given an abstract representation of a primary structure as above, it is possible to modify it by inserting and deleting
Components. The following example deletes the R0010 promoter and replaces it with the R0011 promoter

```python
primary_structure = gene.getPrimaryStructureComponents()
b0032_component = primary_structure[1]
gene.deleteUpstreamComponent(b0032_component)

r0011 = ComponentDefinition('r0011')
r0011.roles = SO_CDS
gene.insertUpstreamComponent(b0032_component, r0011)
```

### 6.4 Sequence Assembly

A complete design adds explicit sequence information to the components in a template design or abstraction hi-
derarchy. In order to complete a design, Sequence objects must first be created and associated with the promoter,
CDS, RBS, terminator subcomponents. In contrast to the ComponentDefinition.assemble() method, which assembles
a template design, the ComponentDefinition.compile method recursively generates the complete sequence of a hier-
archical design from the sequence of its subcomponents. Compiling a DNA sequence is analogous to a programmer
compiling their code. In order to compile a ComponentDefinition, you must first assemble a template design from
ComponentDefinitions, as described in the previous section.

```python
target_sequence = gene.compile()
```

The compile method returns the target sequence as a string. In addition, it creates a new Sequence object and assigns
the target sequence to its elements property.
6.5 Genome Integration

In some cases, it may be useful to represent integration of vectors / transposons into genomes. The `integrateAtBaseCoordinate` method supports integration operations and produces a parsimonious representation of primary structure that is useful for manipulating large constructs. The following example demonstrates integration of the gene construct from the examples above into a `wild_type_genome`, thus generating the `integrated_genome`.

```python
integrated_genome = ComponentDefinition('integrated_genome')
integrated_genome.sequence = Sequence('integrated_genome_sequence')
wild_type_genome = ComponentDefinition('wild_type_genome')
wild_type_genome.sequence = Sequence('wild_type_genome_sequence')
wild_type_genome.sequence.elements = 'gggggggggg'
integrated_genome.integrateAtBaseCoordinate(wild_type_genome, gene, 5)
integrated_genome.compile()  # Calculate sequence of the integrated genome
```

6.6 Full Example Code

Full example code is provided below, which will create a file called “gene_cassette.xml”

```python
from sbol import *

setHomespace('http://sys-bio.org')
doc = Document()

gene = ComponentDefinition('gene_example')
r0010 = ComponentDefinition('R0010')
b0032 = ComponentDefinition('B0032')
e0040 = ComponentDefinition('E0040')
b0012 = ComponentDefinition('B0012')

r0010.roles = SO_PROMOTER
b0032.roles = SO_CDS
e0040.roles = SO_RBS
b0012.roles = SO_TERMINATOR

doc.addComponentDefinition(gene)
doc.addComponentDefinition([ r0010, b0032, e0040, b0012 ])
gene.assemblePrimaryStructure([ r0010, b0032, e0040, b0012 ])

first = gene.getFirstComponent()
print(first.identity)
last = gene.getLastComponent()
print(last.identity)

r0010.sequence = Sequence('R0010', 'ggctgca')
b0032.sequence = Sequence('B0032', 'aattatataaa')
e0040.sequence = Sequence('E0040', 'atgtaa')
b0012.sequence = Sequence('B0012', 'attcga')

target_sequence = gene.compile()
print(gene.sequence.elements)

result = doc.write('gene_cassette.xml')
print(result)
```
CHAPTER 7

Design-Build-Test-Learn Workflows

PySBOL can be used to manage computational and experimental workflows for synthetic biology. The API is based on the design-build-test-learn (DBTL) method for engineering problem solving. These workflows involve the following types of objects:

- A Design is a conceptual representation of a biological system that a synthetic biologist intends to implement in the lab. A Design may describe both structural composition or the intended function of a biological system. In more traditional engineering terms, a Design is analogous to a draft or blueprint, but is a purely digital representation.

- A Build describes an actual, physical sample in the laboratory. A DNA construct is the most common example of a Build in synthetic biology, but the definition can be extended to represent any kind of physical sample, including cells and reagents. A Build may be linked to a laboratory information management system using SBOL.

- A Test is a wrapper for experimental data files that are produced as a result of an experimental measurement on a Build. As a matter of scientific integrity, unaltered experimental data should and must be preserved. A Test object provides a link to those data.

- An Analysis is a wrapper for experimental data that has been processed or transformed. Common data transformations include subtracting background signal (“blanking”), log transformations, and model-fitting.

Note that these pySBOL classes are not part of the core SBOL standard, but are abstractions provided by the pySBOL interface for the convenience of the user. However, they map closely to the SBOL data model.

In order to organize and track data as a workflow proceeds, a user can create objects and link them together using Activity objects. An Activity uses certain types of objects as inputs and generates new objects. For example, under the DBTL formalization a Design is used to generate a Build. This implies that a digital blueprint for a biological system has been realized as a real construct the laboratory. If an experimental measurement is performed subsequently, via an experiment Activity, a Test object is generated. A Test performs measurement on Builds. Finally an Analysis may use the raw experimental data represented by a Test object. Thus, objects are created in a logical order that conforms to the DBTL formalism. This pattern is represented in the UML class diagram below.

An Activity is executed by an Agent which may be a person, a piece of software, or laboratory robotics. The Agent executes a Plan which may be a laboratory protocol written in natural language or a set of automated instructions. The classes Activity, Agent, and Plan are all defined according the Provenance Ontology (PROV-O).
The pySBOL API provides the Design, Build, Test, and Analysis classes to simplify workflow representation. However, it is also possible to use the API to construct PROV-O workflows that conform to other patterns.

The usage pattern described above can be summarized as follows: An Activity may use one or more objects of type X to generate one or more objects of type Y that come next in the DBTL workflow. Additionally, DBTL workflows can be cyclic. An Analysis may generate a Design. In this pattern, the Analysis represents a specification or prediction that is assumed to be true about the Design since it has been previously experimentally verified. Such a pattern is called a DBTL cycle.

The DBTL cycle is a generalized, iterative framework for engineering problem-solving—something like a scientific method for engineers. In the context of synthetic biology, the DBTL cycle may include processes such as pulling data about biological parts from online databases, assembling new genetic programs from DNA sequences, synthesizing and assembling DNA, performing quality control, measurement and model-based characterization of a DNA part's encoded behavior, submitting characterized parts to inventories, and publishing data sheets. Ideally, each cycle generates new knowledge that feeds back into new cycles in the form of alternative approaches, reformulated problems, or forward specifications for future designs.

In addition to the logical workflow order described above, other simple workflow patterns are allowed as well. An Activity that generates an object of type X may use other objects also of type X as inputs. For example, a laboratory construct may go through sequential stages of processing before the target Build is complete. Examples are enzymatic treatment, DNA purification, and transformation. At each stage, a Build uses a prior Build. Therefore Builds may use prior Builds as well as Designs.

Branching and intersecting workflows are other common patterns of usage. For example, an intersecting workflow occurs when a construct is assembled out of multiple physical components, such as occurs when a Gibson assembly uses multiple DNA samples. Multiple Build inputs are used to generate the new Build. Another kind of intersecting workflow occurs when an experimental Test is performed on multiple Build samples. Conversely, branching patterns may also occur. For example, a branching workflow occurs when transformation of a single Gibson reaction mixture generates multiple clones, each of which may be subjected to their own unique history of subsequent testing and analysis.

```
from sbol import *
```
doc = Document()
setHomespace('https://sys-bio.org')

doc = Document()

workflow_step_1 = Activity('build_1')
workflow_step_2 = Activity('build_2')
workflow_step_3 = Activity('build_3')
workflow_step_4 = Activity('build_4')
workflow_step_5 = Activity('build_5')
workflow_step_6 = Activity('test_1')
workflow_step_7 = Activity('analysis_1')

workflow_step_1.plan = Plan('PCR_protocol_part1')
workflow_step_2.plan = Plan('PCR_protocol_part2')
workflow_step_3.plan = Plan('PCR_protocol_part3')
workflow_step_4.plan = Plan('gibson_assembly')
workflow_step_5.plan = Plan('transformation')
workflow_step_6.plan = Plan('promoter_characterization')
workflow_step_7.plan = Plan('parameter_optimization')

setHomespace('')
Config.setOption('sbol_compliant_uris', False)  # Temporarily disable auto-
  →construction of URIs

workflow_step_1.agent = Agent('mailto:jdoe@sbols.org')
workflow_step_2.agent = workflow_step_1.agent
workflow_step_3.agent = workflow_step_1.agent
workflow_step_4.agent = workflow_step_1.agent
workflow_step_5.agent = workflow_step_1.agent
workflow_step_6.agent = Agent('http://sys-bio.org/plate_reader_1')
workflow_step_7.agent = Agent('http://tellurium.analogmachine.org')

Config.setOption('sbol_compliant_uris', True)
setHomespace('https://sys-bio.org')

doc.addActivity([workflow_step_1, workflow_step_2, workflow_step_3, workflow_step_4,
  →workflow_step_5, workflow_step_6, workflow_step_7])

target = Design('target')
part1 = workflow_step_1.generateBuild('part1', target)
part2 = workflow_step_2.generateBuild('part2', target)
part3 = workflow_step_3.generateBuild('part3', target)
gibson_mix = workflow_step_4.generateBuild('gibson_mix', target, [part1, part2, part3])
clones = workflow_step_5.generateBuild(['clone1', 'clone2', 'clone3'], target, gibson_mix)
experiment1 = workflow_step_6.generateTest('experiment1', clones)
analysis1 = workflow_step_7.generateAnalysis('analysis1', experiment1)

response = doc.write('dbtl.xml')
print(response)
A generated Entity is linked through a wasGeneratedBy relationship to an Activity, which is used to describe how different Agents and other entities were used. An Activity is linked through a qualifiedAssociation to Associations, to describe the role of agents, and is linked through qualifiedUsage to Usages to describe the role of other entities used as part of the activity. Moreover, each Activity includes optional startedAtTime and endedAtTime properties. When using Activity to capture how an entity was derived, it is expected that any additional information needed will be attached as annotations. This may include software settings or textual notes. Activities can also be linked together using the wasInformedBy relationship to provide dependency without explicitly specifying start and end times.

- **startedAtTime**: `DateTimeProperty`
- **endedAtTime**: `DateTimeProperty` The endedAtTime property is OPTIONAL and contains a dateTime value, indicating when the activity ended.
- **wasInformedBy**: `ReferencedObject` The wasInformedBy property is OPTIONAL and contains a URI of another activity.
- **associations**: `[OwnedObject< Association >]` The qualifiedAssociation property is OPTIONAL and MAY contain a set of URIs that refers to Association.
- **usages**: `[OwnedObject< Usage >]` The qualifiedUsage property is OPTIONAL and MAY contain a set of URIs that refers to Usage objects.
- **agent**: `[OwnedObject< Agent >]` An Agent object may be specified here, and it will be synced with the Association::agent property.
- **plan**: `[OwnedObject< Plan >]` A Plan object may be specified here, and it will be synced with the Association::plan property.
- **attachments**: `ReferencedObject`
- **persistentIdentity**: `URIProperty` The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

```py
class Activity(*args):
    A generated Entity is linked through a wasGeneratedBy relationship to an Activity, which is used to describe how different Agents and other entities were used. An Activity is linked through a qualifiedAssociation to Associations, to describe the role of agents, and is linked through qualifiedUsage to Usages to describe the role of other entities used as part of the activity. Moreover, each Activity includes optional startedAtTime and endedAtTime properties. When using Activity to capture how an entity was derived, it is expected that any additional information needed will be attached as annotations. This may include software settings or textual notes. Activities can also be linked together using the wasInformedBy relationship to provide dependency without explicitly specifying start and end times.

    • **startedAtTime**: `DateTimeProperty`
    • **endedAtTime**: `DateTimeProperty` The endedAtTime property is OPTIONAL and contains a dateTime value, indicating when the activity ended.
    • **wasInformedBy**: `ReferencedObject` The wasInformedBy property is OPTIONAL and contains a URI of another activity.
    • **associations**: `[OwnedObject< Association >]` The qualifiedAssociation property is OPTIONAL and MAY contain a set of URIs that refers to Association.
    • **usages**: `[OwnedObject< Usage >]` The qualifiedUsage property is OPTIONAL and MAY contain a set of URIs that refers to Usage objects.
    • **agent**: `[OwnedObject< Agent >]` An Agent object may be specified here, and it will be synced with the Association::agent property.
    • **plan**: `[OwnedObject< Plan >]` A Plan object may be specified here, and it will be synced with the Association::plan property.
    • **attachments**: `ReferencedObject`
    • **persistentIdentity**: `URIProperty` The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.
```
• **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/provo.h

```cpp
  copy (*args)
  copy(ns="", version="") -> SBOLClass &

  generateAnalysis (uris, test_usages, analysis_usages=None)
  Generate one or more Analysis objects.

  • **uris** [] One or more identifiers for the new Analysis object(s). If the sbol_compliant_uris option configuration is enabled, then the user should specify simple identifiers for the objects. Otherwise the user must provide full URIs each consisting of a scheme, namespace, and identifier.

  • **test_usages** [] A singleton Test object, list of Test objects, or None. Test usages represent raw experimental data used to generate an Analysis.

  • **analysis_usages** [] A singleton Analysis object, list of Analysis objects, or None. Analysis usages represent other analyses that the user wants to integrate into a single data set or data sheet.
```
A singleton Analysis or list of Analyses depending on whether the user specifies a single URI or list of URIs.

**generateBuild**(uris, design_usages, build_usages=None)
Generate one or more Build objects

- **uris** [] One or more identifiers for the new Build object(s). If the sbol_compliant_uris option configuration is enabled, then the user should specify simple identifiers for the objects. Otherwise the user must provide full URIs each consisting of a scheme, namespace, and identifier.

- **design_usages** [] A singleton Design object, list of Design objects, or None. Design usages represent the engineer’s intent or “blueprint” for the Build target.

- **build_usages** [] A singleton Build object, list of Build objects, or None. Build usages represent physical components, such as laboratory samples, that are assembled into the target Build.

A singleton Build or list of Builds depending on whether the user specifies a single URI or list of URIs.

**generateDesign**(uris, analysis_usages, design_usages=None)
Generate one or more Design objects

- **uris** [] One or more identifiers for the new Design object(s). If the sbol_compliant_uris option configuration is enabled, then the user should specify simple identifiers for the objects. Otherwise the user must provide full URIs each consisting of a scheme, namespace, and identifier.

- **analysis_usages** [] A singleton Analysis object, list of Analysis objects, or None. Analysis usages represent a prediction or forward-specification of the new Design’s intended structure or function.

- **design_usages** [] A singleton Design object, list of Design objects, or None. Design usages may represent previous Designs that are being tranformed or composed into the new Design.

A singleton Design or list of Designs depending on whether the user specifies a single URI or list of URIs.

**generateTest**(uris, build_usages, test_usages=None)
Generate one or more Test objects

- **uris** [] One or more identifiers for the new Test object(s). If the sbol_compliant_uris option configuration is enabled, then the user should specify simple identifiers for the objects. Otherwise the user must provide full URIs each consisting of a scheme, namespace, and identifier.

- **build_usages** [] A singleton Build object, list of Build objects, or None. Build usages represent samples or analytes used in an experimental measurement.

- **test_usages** [] A singleton Test object, list of Test objects, or None. Test usages represent other measurements or raw data that the user wants to integrate into a single data set.

A singleton Test or list of Tests depending on whether the user specifies a single URI or list of URIs.

**class ActivityProperty**(*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter : std::vector<std::string>::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

**add**(new_value)

```
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.
• **new_value**  [] A new string which will be added to a list of values.

```python
addValidationRule(*args)
addValidationRule(rule)
```

```python
clear()
clear()
```
Clear all property values.

```python
copy(target_property)
copy(target_property)
```
Copy property values to a target object’s property fields.

```python
find(query)
find(query) -> bool
```
Check if a value in this property matches the query.

```python
getLowerBound()
getLowerBound() -> char
```

```python
getOwner()
getOwner() -> SBOLObject &
```

```python
getTypeURI()
getTypeURI() -> rdf_type
```

```python
getUpperBound()
getUpperBound() -> char
```

```python
remove(index=0)
remove(index=0)
```
Remove a property value.

```python
set(*args)
set(new_value)
```
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value**  [] A new integer value for the property, which is converted to a raw string during serialization.

```python
validate(arg=None)
validate(arg=NULL)
```

```python
write()
write()
```

```python
class Agent(*args)
```
Examples of agents are person, organisation or software. These agents should be annotated with additional information, such as software version, needed to be able to run the same software again.

• **attachments**: ReferencedObject

• **persistentIdentity**: URIProperty  ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId**: TextProperty  ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its
String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version**  
  [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom**  
  [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy**  
  [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name**  
  [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description**  
  [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity**  
  [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/provo.h

```
copy(*args)  
copy(ns="", version="") -> SBOLClass &
```

class AgentProperty(*args)  
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The**  
  [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter : std::vector<std::string>::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```
add(new_value)  
add(new_value)  
```

Appends the new value to a list of values, for properties that allow it.

- **new_value**  
  [] A new string which will be added to a list of values.
addValidationRule(*args)
    addValidationRule(rule)

clear()
    clear()
    Clear all property values.

copy(target_property)
    copy(target_property)
    Copy property values to a target object’s property fields.

find(query)
    find(query) -> bool
    Check if a value in this property matches the query.

getLowerBound()
    getLowerBound() -> char

getOwner()
    getOwner() -> SBOLObject &

getTypeURI()
    getTypeURI() -> rdf_type

getUpperBound()
    getUpperBound() -> char

remove(index=0)
    remove(index=0)
    Remove a property value.

set(*args)
    set(new_value)
    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
    • new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
    validate(arg=NULL)

write()
    write()

class AliasedOwnedFunctionalComponent(property_owner, sbol_uri, alias_uri, lower_bound, upper_bound, validation_rules)

    • alias : rdf_type
    • python_iter : std::vector< std::string >::iterator

add(sbol_obj)
    add(new_value)
    Appends the new value to a list of values, for properties that allow it.
    • new_value [] A new string which will be added to a list of values.

create(uri)
    create(uri) -> SBOLClass &
**define** (definition_object)
   define(definition_object) -> SBOLClass &

**get** (*args)
   get(uri="") -> SBOLClass &

**set** (sbol_obj)
   set(new_value)
   Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
   
   • **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class** Analysis (*args)

- **rawData** [ReferencedObject ] A reference to a Test object which contains the raw data for an Analysis.
- **dataFiles** [ReferencedObject ] References to file Attachments which contain experimental data sets.
- **dataSheet** [ReferencedObject ] A reference to a datasheet file.
- **consensusSequence** [OwnedObject< Sequence > ] A sequence object that represents a consensus sequence from DNA sequencing data.
- **attachments**: ReferencedObject
- **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.
- **displayId** [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
- **version** [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.
- **wasDerivedFrom** [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.
- **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.
- **name** [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name
properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

```python
copy (*args)
copy(ns="", version="") -> SBOLClass &
```

```python
reportAmbiguity()

'reportAmbiguity() -> std::unordered_map< std::string, std::tuple< int, int, float > >'
```

```python
reportCoverage()

'reportCoverage() -> std::unordered_map< std::string, std::tuple< int, int, float > >'
```

```python
reportError()

'reportError() -> std::unordered_map< std::string, std::tuple< int, int, float > >'
```

```python
reportIdentity()

'reportIdentity() -> std::unordered_map< std::string, std::tuple< int, int, float > >'
```

```python
verifyTarget (consensus_sequence)
verifyTarget(consensus_sequence)

Compare a consensus Sequence to the target Sequence.
```

class AnalysisProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The []** SBOL specification currently supports string, URI, and integer literal values.

- **python_iter:** std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```python
add (new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

addValidationRule (*args)
addValidationRule(rule)
```

clear ()
clear()

Clear all property values.
copy (target_property)
    copy(target_property)

    Copy property values to a target object’s property fields.

find (query)
    find(query) -> bool

    Check if a value in this property matches the query.

getLowerBound ()
    getLowerBound() -> char

getOwner ()
    getOwner() -> SBOLObject &

getTypeURI ()
    getTypeURI() -> rdf_type

getUpperBound ()
    getUpperBound() -> char

remove (index=0)
    remove(index=0)

    Remove a property value.

set (*args)
    set(new_value)

    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
    
    • new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
    validate(arg=NULL)

write ()
    write()

class Association (*args)

    An Association is linked to an Agent through the agent relationship. The Association includes the hadRole property to qualify the role of the Agent in the Activity.
    
    • agent [ReferencedObject ] The agent property is REQUIRED and MUST contain a URI that refers to an Agent object.
    
    • roles [URIProperty ] The hadRole property is REQUIRED and MUST contain a URI that refers to a particular term describing the usage of the agent.
    
    • plan [ReferencedObject ] The hadPlan property is OPTIONAL and contains a URI that refers to a Plan.
    
    • persistentIdentity [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.
    
    • displayId [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
• **version**  [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom**  [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy**  [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name**  [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description**  [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity**  [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/provo.h

class AssociationProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

**add(new_value)**
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

**addValidationRule(*args)**
addValidationRule(rule)

clear()
clear()
Clear all property values.

**copy** *(target_property)*

    copy(target_property)

Copy property values to a target object’s property fields.

**find** *(query)*

    find(query) -> bool

Check if a value in this property matches the query.

**getLowerBound** *

    getLowerBound() -> char

**getOwner** *

    getOwner() -> SBOLObject &

**getTypeURI** *

    getTypeURI() -> rdf_type

**getUpperBound** *

    getUpperBound() -> char

**remove** *(index=0)*

    remove(index=0)

Remove a property value.

**set** *(new_value)*

    set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**validate** *(arg=None)*

    validate(arg=NULL)

**write** *

    write()

**class Attachment** *(args)*

The Attachment class is a general container for data files, especially experimental data files. Attachment is a TopLevel object, and any other TopLevel object can refer to a list of attachments.

• **source** [URIProperty] The source is a link to the external file and is REQUIRED.

• **format** : URIProperty

• **size** : IntProperty

• **hash** : TextProperty

• **attachments** : ReferencedObject

• **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its
String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/attachment.h

```cpp
copy ( nargs )
    copy(ns="", version="") -> SBOLClass &
```

**class AttachmentProperty ( nargs )**

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [ ] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add ( new_value )
    add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [ ] A new string which will be added to a list of values.
**addValidationRule**(*args*)
   addValidationRule(rule)

**clear**()
   clear()
   Clear all property values.

**copy**(*target_property*)
   copy(target_property)
   Copy property values to a target object’s property fields.

**find**(*query*)
   find(query) -> bool
   Check if a value in this property matches the query.

**getLowerBound**()
   getLowerBound() -> char

**getOwner**()
   getOwner() -> SBOLObject &

**getTypeURI**()
   getTypeURI() -> rdf_type

**getUpperBound**()
   getUpperBound() -> char

**remove**(*index=0*)
   remove(index=0)
   Remove a property value.

**set**(*args*)
   set(new_value)
   Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
   - **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**validate**(*arg=None*)
   validate(arg=NULL)

**write**()
   write()

**class Build**(*args*)
   A Build is a realization of a Design. For practical purposes, a Build can represent a biological clone, a plasmid, or other laboratory sample. For a given Design, there may be multiple Builds realized in the lab. A Build represents the second step in libSBOL’s formalized Design-Build-Test-Analyze workflow.
   - **design** [ReferencedObject] A reference to a Design object which represents the intended structure and function for this Build.
   - **structure** [OwnedObject< ComponentDefinition >] The experimentally verified structure of the construct as verified by DNA sequencing or other analysis.
   - **behavior** [OwnedObject< ModuleDefinition >] The observed behavior of the constructed system.
   - sysbio_type : URIProperty
   - _structure : ReferencedObject
pysbol2 Documentation, Release 2.3.0

- _behavior : ReferencedObject
- built : URIProperty
- attachments : ReferencedObject
- **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- displayId [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- version [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- wasDerivedFrom [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- wasGeneratedBy [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- name [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- description [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- identity [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/dbtl.h

```cpp
copy(*args)
copy(ns="", version="") -> SBOLClass &

class BuildProperty(*args)
Member properties of all SBOL objects are defined using a Property object.
```
The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter**: std::vector<std::string>::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

**add** (new_value)

add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value**: [] A new string which will be added to a list of values.

**addValidationRule** (*args)

addValidationRule(rule)

**clear** ()

clear()

Clear all property values.

**copy** (target_property)

copy(target_property)

Copy property values to a target object’s property fields.

**find** (query)

find(query) -> bool

Check if a value in this property matches the query.

**getLowerBound** ()

gtLowerBound() -> char

**getOwner** ()

gtOwner() -> SBOLObject &

**getTypeURI** ()

gtTypeURI() -> rdf_type

**getUpperBound** ()

gtUpperBound() -> char

**remove** (index=0)

remove(index=0)

Remove a property value.

**set** (*args)

set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value**: [] A new integer value for the property, which is converted to a raw string during serialization.

**validate** (arg=None)

validate(arg=NULL)

**write** ()

write()
class Collection(*args)
The Collection class is a class that groups together a set of TopLevel objects that have something in common.

Some examples of Collection objects: . Results of a query to find all ComponentDefinition objects in a repository that function as promoters . A set of ModuleDefinition objects representing a library of genetic logic gates. . A ModuleDefinition for a complex design, and all of the ModuleDefinition, ComponentDefinition, Sequence, and Model objects used to provide its full specification.

- **members** [URIProperty] The members property of a Collection is OPTIONAL and MAY contain a set of URI references to zero or more TopLevel objects.

- **attachments** : ReferencedObject

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/collection.h
copy (*args)
copy(ns='', version='') -> SBOLClass &

class CollectionProperty (*args)
Member properties of all SBOL objects are defined using a Property object.
The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class
converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.
- **python_iter** : `std::vector< std::string >::iterator`

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add (new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

addValidationRule (*args)
addValidationRule(rule)

clear ()
clear()

Clear all property values.

copy (target_property)
copy(target_property)

Copy property values to a target object’s property fields.

find (query)
find(query) -> bool

Check if a value in this property matches the query.

getLowerBound ()
getLowerBound() -> char

goOwner ()
getOwner() -> SBOLObject &

getTypeURI ()
getTypeURI() -> rdf_type

goUpperBound ()
goUpperBound() -> char

remove (index=0)
remove(index=0)

Remove a property value.

set (*args)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.
validate(arg=None)
validate(arg=NULL)

write()
write()

class CombinatorialDerivation(*args)

A ComponentDerivation specifies the composition of a combinatorial design or variant library for common use cases in synthetic biology, such as tuning the performance of a genetic circuit or biosynthetic pathway through combinatorial DNA assembly and screening.

- **strategy** [URIProperty] The strategy property is OPTIONAL and has a data type of URI.
  
  Table 1 provides a list of REQUIRED strategy URIs. If the strategy property is not empty, then it MUST contain a URI from Table 1. This property recommends how many ComponentDefinition objects a user SHOULD derive from the template ComponentDefinition. Strategy URI Description

  A user SHOULD derive all ComponentDefinition objects with a unique substructure as specified by the Component objects contained by the template ComponentDefinition and the VariableComponent objects contained by the CombinatorialDerivation.

  - http://sbols.org/v2#enumerate
    A user SHOULD derive all ComponentDefinition objects with a unique substructure as specified by the Component objects contained by the template ComponentDefinition and the VariableComponent objects contained by the CombinatorialDerivation. Strategy URI Description

  - http://sbols.org/v2#sample
    A user SHOULD derive a subset of all ComponentDefinition objects with a unique substructure as specified by the Component objects contained by the template ComponentDefinition and the VariableComponent objects contained by the CombinatorialDerivation. The manner in which this subset is chosen is for the user to decide.

- **masterTemplate** [ReferencedObject] The master property is REQUIRED and MUST contain a URI that refers to a ComponentDefinition. This ComponentDefinition is expected to serve as a template for the derivation of new ComponentDefinition objects. Consequently, its components property SHOULD contain one or more Component objects that describe its substructure (referred to hereafter as template Component objects), and its sequenceConstraints property MAY also contain one or more SequenceConstraint objects that constrain this substructure.

- **variableComponents** [OwnedObject< VariableComponent >] VariableComponent objects denote the choices available when deriving the library of variants specified by a CombinatorialDerivation.

- **attachments**: ReferencedObject

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.
wasDerivedFrom [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

wasGeneratedBy [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

name [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

description [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

identity [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/combinatorialderivation.h

```cpp
copy ( *args)
    copy(ns="", version="") -> SBOLClass &
```

class CombinatorialDerivationProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [ ] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add (new_value)
    add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [ ] A new string which will be added to a list of values.

```cpp
addValidationRule (*args)
    addValidationRule(rule)
```

clear ()
    clear()
```

Clear all property values.
**copy** *(target_property)*

    copy(target_property)

Copy property values to a target object’s property fields.

**find** *(query)*

    find(query) -> bool

Check if a value in this property matches the query.

**getLowerBound** *

    getLowerBound() -> char

**getOwner** *

    getOwner() -> SBOLObject &

**getTypeURI** *

    getTypeURI() -> rdf_type

**getUpperBound** *

    getUpperBound() -> char

**remove** *(index=0)*

    remove(index=0)

Remove a property value.

**set** *(args)*

    set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

  * new_value [] A new integer value for the property, which is converted to a raw string during serialization.

**validate** *(arg=None)*

    validate(arg=NULL)

**write** *

    write()
• **roleIntegration**: [URIProperty] A roleIntegration specifies the relationship between a Component instances own set of roles and the set of roles on the included sub-ComponentDefinition.

The roleIntegration property has a data type of URI. A Component instance with zero roles MAY OPTIONALLY specify a roleIntegration. A Component instance with one or more roles MUST specify a roleIntegration from the table below. If zero Component roles are given and no Component roleIntegration is given, then http://sbols.org/v2#mergeRoles is assumed. It is RECOMMENDED to specify a set of Component roles only if the integrated result set of roles would differ from the set of roles belonging to this Components included sub-ComponentDefinition. roleIntegration URI

Description

http://sbols.org/v2#overrideRoles

In the context of this Component, ignore any roles given for the included sub-ComponentDefinition. Instead use only the set of zero or more roles given for this Component.

http://sbols.org/v2#mergeRoles

Use the union of the two sets: both the set of zero or more roles given for this Component as well as the set of zero or more roles given for the included sub-ComponentDefinition.

• **definition**: [ReferencedObject] The definition property is a REQUIRED URI that refers to the ComponentDefinition of the ComponentInstance.

As described in the previous section, this ComponentDefinition effectively provides information about the types and roles of the ComponentInstance. The definition property MUST NOT refer to the same ComponentDefinition as the one that contains the ComponentInstance. Furthermore, ComponentInstance objects MUST NOT form a cyclical chain of references via their definition properties and the ComponentDefinition objects that contain them. For example, consider the ComponentInstance objects A and B and the ComponentDefinition objects X and Y. The reference chain X contains A, A isdefinedby Y, Y contains B, and B isdefinedby X isyclical.

• **access**: [URIProperty] The access property is a REQUIRED URI that indicates whether the ComponentInstance can be referred to remotely by a MapsTo.

The value of the access property MUST be one of the following URIs. Access URI

Description

http://sbols.org/v2#public

The ComponentInstance MAY be referred to by remote MapsTo objects

http://sbols.org/v2#private

The ComponentInstance MAY be referred to by remote MapsTo objects

• **mapsTos**: [OwnedObject<MapsTo>] The mapsTos property is OPTIONAL and MAY contain a set of MapsTo objects that refer to and link together ComponentInstance objects (both Component objects and FunctionalComponent objects) within a larger design.

• **persistentIdentity**: [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId**: [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/component.h

class ComponentDefinition (*args)

The ComponentDefinition class represents the structural entities of a biological design.

The primary usage of this class is to represent structural entities with designed sequences, such as DNA, RNA, and proteins, but it can also be used to represent any other entity that is part of a design, such as small molecules, proteins, and complexes

• **types** [URIProperty] The types property is a REQUIRED set of URIs that specifies the category of biochemical or physical entity (for example DNA, protein, or small molecule) that a ComponentDefinition object abstracts for the purpose of engineering design.

The types property of every ComponentDefinition MUST contain one or more URIs that MUST identify terms from appropriate ontologies, such as the BioPAX ontology or the ontology of Chemical Entities of Biological Interest. See the table below for examples. Type

 URI for BioPAX Term
LibSBOL symbol
DNA

http://www.biopax.org/release/biopax-level3.owl#DnaRegion
BIOPAX_DNA
Roles

The roles property is an OPTIONAL set of URIs that clarifies the potential function of the entity represented by a ComponentDefinition in a biochemical or physical context.

The roles property of a ComponentDefinition MAY contain one or more URIs that MUST identify terms from ontologies that are consistent with the types property of the ComponentDefinition. For example, the roles property of a DNA or RNA ComponentDefinition could contain URIs identifying terms from the Sequence Ontology (SO). See the table below for common examples Role URI for Sequence Ontology Term

<table>
<thead>
<tr>
<th>LibSBOL symbol</th>
<th>URI for Sequence Ontology Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td><a href="http://identifiers.org/so/SO:0000001">http://identifiers.org/so/SO:0000001</a></td>
</tr>
<tr>
<td>Promoter</td>
<td><a href="http://identifiers.org/so/SO:0000167">http://identifiers.org/so/SO:0000167</a></td>
</tr>
<tr>
<td>RBS</td>
<td><a href="http://identifiers.org/so/SO:0000139">http://identifiers.org/so/SO:0000139</a></td>
</tr>
<tr>
<td>CDS</td>
<td><a href="http://identifiers.org/so/SO:0000316">http://identifiers.org/so/SO:0000316</a></td>
</tr>
<tr>
<td>Terminator</td>
<td><a href="http://identifiers.org/so/SO:0000141">http://identifiers.org/so/SO:0000141</a></td>
</tr>
</tbody>
</table>

• roles [URIProperty] The roles property is an OPTIONAL set of URIs that clarifies the potential function of the entity represented by a ComponentDefinition in a biochemical or physical context.

The roles property of a ComponentDefinition MAY contain one or more URIs that MUST identify terms from ontologies that are consistent with the types property of the ComponentDefinition. For example, the roles property of a DNA or RNA ComponentDefinition could contain URIs identifying terms from the Sequence Ontology (SO). See the table below for common examples Role URI for Sequence Ontology Term

<table>
<thead>
<tr>
<th>LibSBOL symbol</th>
<th>URI for Sequence Ontology Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td><a href="http://identifiers.org/so/SO:0000001">http://identifiers.org/so/SO:0000001</a></td>
</tr>
<tr>
<td>Promoter</td>
<td><a href="http://identifiers.org/so/SO:0000167">http://identifiers.org/so/SO:0000167</a></td>
</tr>
<tr>
<td>RBS</td>
<td><a href="http://identifiers.org/so/SO:0000139">http://identifiers.org/so/SO:0000139</a></td>
</tr>
<tr>
<td>CDS</td>
<td><a href="http://identifiers.org/so/SO:0000316">http://identifiers.org/so/SO:0000316</a></td>
</tr>
<tr>
<td>Terminator</td>
<td><a href="http://identifiers.org/so/SO:0000141">http://identifiers.org/so/SO:0000141</a></td>
</tr>
</tbody>
</table>

Gene

http://identifiers.org/so/SO:0000704
Operator
http://identifiers.org/so/SO:0000057

Engineered Gene
http://identifiers.org/so/SO:0000280

mRNA
http://identifiers.org/so/SO:0000234

Effector
http://identifiers.org/chebi/CHEBI:35224

- **components** [OwnedObject< Component > ] The components property is OPTIONAL and MAY specify a set of Component objects that are contained by the ComponentDefinition. The components properties of ComponentDefinition objects can be used to construct a hierarchy of Component and ComponentDefinition objects. If a ComponentDefinition in such a hierarchy refers to one or more Sequence objects, and there exist ComponentDefinition objects lower in the hierarchy that refer to Sequence objects with the same encoding, then the elements properties of these Sequence objects SHOULD be consistent with each other, such that well-defined mappings exist from the lower level elements to the higher level elements. This mapping is also subject to any restrictions on the positions of the Component objects in the hierarchy that are imposed by the SequenceAnnotation or SequenceConstraint objects contained by the ComponentDefinition objects in the hierarchy. The set of relations between Component and ComponentDefinition objects is strictly acyclic.

- **sequences** [ReferencedObject ] The sequences property is OPTIONAL and MAY include a URI that refer to a Sequence object. The referenced object defines the primary structure of the ComponentDefinition.

  - **sequence** : OwnedObject< Sequence >

- **sequenceAnnotations** [OwnedObject< SequenceAnnotation > ] The sequenceAnnotations property is OPTIONAL and MAY contain a set of SequenceAnnotation objects. Each SequenceAnnotation specifies and describes a potentially discontiguous region on the Sequence objects referred to by the ComponentDefinition.

- **sequenceConstraints** [OwnedObject< SequenceConstraint > ] The sequenceConstraints property is OPTIONAL and MAY contain a set of SequenceConstraint objects. These objects describe any restrictions on the relative, sequence-based positions and/or orientations of the Component objects contained by the ComponentDefinition. For example, the ComponentDefinition of a gene might specify that the position of its promoter Component precedes that of its CDS Component. This is particularly useful when a ComponentDefinition lacks a Sequence and therefore cannot specify the precise, sequence-based positions of its Component objects using SequenceAnnotation objects.

- **attachments** : ReferencedObject

- **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This...
convention represents versions as sequences of numbers and qualifiers that are separated by the characters - and · and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/componentdefinition.h

**addDownstreamFlank** (target, elements)

addDownstreamFlank(target, elements)

This may be a useful method when building up SBOL representations of natural DNA sequences.

For example it is often necessary to specify components that are assumed to have no meaningful role in the design, but are nevertheless important to fill in regions of sequence. This method autoconstructs a ComponentDefinition and Sequence object to create an arbitrary flanking sequence around design Components. The new ComponentDefinition will have Sequence Ontology type of flanking_sequence.

• **target** [] The new flanking sequence will be placed downstream of the target

• **elements** [] The primary sequence elements will be assigned to the autoconstructed Sequence object. The encoding is inferred

**addUpstreamFlank** (target, elements)

addUpstreamFlank(target, elements)

This may be a useful method when building up SBOL representations of natural DNA sequences.

For example it is often necessary to specify components that are assumed to have no meaningful role in the design, but are nevertheless important to fill in regions of sequence. This method autoconstructs a ComponentDefinition and Sequence object to create an arbitrary flanking sequence around design Components. The new ComponentDefinition will have Sequence Ontology type of flanking_region or SO:0000239

• **target** [] The new flanking sequence will be placed upstream of the target
• **elements**  [] The primary sequence elements will be assigned to the autoconstructed Sequence object. The encoding is inferred

```python
assemble(*args)
assemble(list_of_uris, assembly_standard="")
```
Assembles ComponentDefinitions into an abstraction hierarchy.

The resulting data structure is a partial design, still lacking a primary structure or explicit sequence. To form a primary structure out of the ComponentDefinitions, call linearize after calling assemble. To fully realize the target sequence, use Sequence::assemble().

• **list_of_uris**  [] A list of URIs for the constituent ComponentDefinitions, or displayIds if using SBOL-compliant URIs

• **assembly_standard**  [] An optional argument such as IGEM_STANDARD_ASSEMBLY that affects how components are composed and the final target sequence

```python
assemblePrimaryStructure(*args)
assemblePrimaryStructure(primary_structure, doc, assembly_standard="")
```
Assembles ComponentDefinition into a linear primary structure.

The resulting data structure is a partial design, still lacking an explicit sequence. To fully realize the target sequence, use Sequence::assemble().

• **list_of_components**  [] A list of subcomponents that will compose this ComponentDefinition

• **doc**  [] The Document to which the assembled ComponentDefinitions will be added

• **assembly_standard**  [] An optional argument such as IGEM_STANDARD_ASSEMBLY that affects how components are composed and the final target sequence

```python
build()
build() -> ComponentDefinition &
```

```python
compile()
compile() -> std::string
```
Compiles an abstraction hierarchy of ComponentDefinitions into a nucleotide sequence. If no Sequence object is associated with this ComponentDefinition, one will be automatically instantiated.

```python
copy(*args)
copy(ns=", version="") -> SBOLClass &
```

```python
disassemble(range_start=1)
disassemble(range_start=1)
```
Instantiates a Component for every SequenceAnnotation. When converting from a flat GenBank file to a flat SBOL file, the result is a ComponentDefinition with SequenceAnnotations. This method will convert the flat SBOL file into hierarchical SBOL.

```python
getDownstreamComponent(current_component)
getchDownstreamComponent(current_component) -> Component &
```
Get the downstream Component.

The downstream component

```python
getFirstComponent()
getchFirstComponent() -> Component &
```
Gets the first Component in a linear sequence.

The first component in sequential order
getInSequentialOrder()
getInSequentialOrder() -> std::vector< Component * >
Orders this ComponentDefinition’s member Components into a linear arrangement based on Sequence Constraints.

Primary sequence structure

getLastComponent()
getLastComponent() -> Component &
Gets the last Component in a linear sequence.

The last component in sequential order

getPrimaryStructure()
getPrimaryStructure() -> std::vector< ComponentDefinition * >
Get the primary sequence of a design in terms of its sequentially ordered Components.

getUpstreamComponent(current_component)
getUpstreamComponent(current_component) -> Component &
Get the upstream Component.

The upstream component

hasDownstreamComponent(current_component)
hasDownstreamComponent(current_component) -> int
Checks if the specified Component has a Component downstream in linear arrangement on the DNA strand.
Checks that the appropriate SequenceConstraint exists.

• current_component[] A Component in this ComponentDefinition
1 if found, 0 if not

hasUpstreamComponent(current_component)
hasUpstreamComponent(current_component) -> int
Checks if the specified Component has a Component upstream in linear arrangement on the DNA strand.
Checks that the appropriate SequenceConstraint exists.

• current_component[] A Component in this ComponentDefinition
1 if found, 0 if not

insertDownstream(target, insert)
insertDownstream(target, insert)
Insert a Component downstream of another in a primary sequence, shifting any adjacent Components downstream as well.

• target[] The target Component will be upstream of the insert Component after this operation.
• insert[] The insert Component is inserted downstream of the target Component.

insertUpstream(target, insert)
insertUpstream(target, insert)
Insert a Component upstream of another in a primary sequence, shifting any adjacent Components upstream as well.

• target[] The target Component will be downstream of the insert Component after this operation.
• insert[] The insert Component is inserted upstream of the target Component.
isComplete (*args)
    isComplete() -> bool

    Recursively verifies that the parent Document contains a ComponentDefinition and Sequence for each and every ComponentDefinition in the abstraction hierarchy.

    If a ComponentDefinition is not complete, some objects are missing from the Document or externally linked. Diagnose with isComplete(std::string &msg)

    true if the abstraction hierarchy is complete, false otherwise.

isRegular (*args)
    isRegular() -> bool

    Recursively checks if this ComponentDefinition defines a SequenceAnnotation and Range for every Sequence.

    Regularity is more stringent than completeness. A design must be complete to be regular. If the Component is irregular, diagnose with isRegular(std::string &msg)

    true if the abstraction hierarchy is regular, false otherwise.

linearize (*args)
    linearize(list_of_uris)

class ComponentDefinitionProperty (*args)

    Member properties of all SBOL objects are defined using a Property object.

    The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

    • The [] SBOL specification currently supports string, URI, and integer literal values.

    • python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add (new_value)
    add(new_value)

    Appends the new value to a list of values, for properties that allow it.

    • new_value [] A new string which will be added to a list of values.

addValidationRule (*args)
    addValidationRule(rule)
clear()
clear()
Clear all property values.

copy (target_property)
copy(target_property)
Copy property values to a target object’s property fields.

find (query)
find(query) -> bool
Check if a value in this property matches the query.

getLowerBound ()
getLowerBound() -> char

getOwner ()
getOwner() -> SBOLObject &

getTypeURI ()
getypeURI() -> rdf_type

getUpperBound ()
getUpperBound() -> char

remove (index=0)
remove(index=0)
Remove a property value.

set (*args)
set(new_value)
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
  • new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
validate(arg=NULL)

write ()
write()

class ComponentInstance (*args, **kwargs)
  • definition [ReferencedObject ] The definition property is a REQUIRED URI that refers to the ComponentDefinition of the ComponentInstance.

  As described in the previous section, this ComponentDefinition effectively provides information about the types and roles of the ComponentInstance. The definition property MUST NOT refer to the same ComponentDefinition as the one that contains the ComponentInstance. Furthermore, ComponentInstance objects MUST NOT form a cyclical chain of references via their definition properties and the ComponentDefinition objects that contain them. For example, consider the ComponentInstance objects A and B and the ComponentDefinition objects X and Y. The reference chain X contains A, A is defined by Y, Y contains B, and B is defined by X is cyclical.

  • access [URIProperty ] The access property is a REQUIRED URI that indicates whether the ComponentInstance can be referred to remotely by a MapsTo.

    The value of the access property MUST be one of the following URIs. Access URI Description
The ComponentInstance MAY be referred to by remote MapsTo objects

http://sbols.org/v2#public

The ComponentInstance MAY be referred to by remote MapsTo objects

http://sbols.org/v2#private

• mapsTos [OwnedObject< MapsTo > ] The mapsTos property is OPTIONAL and MAY contain a set of MapsTo objects that refer to and link together ComponentInstance objects (both Component objects and FunctionalComponent objects) within a larger design.

• persistentIdentity [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• displayId [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• version [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• wasDerivedFrom [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• wasGeneratedBy [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• name [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• description [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• identity [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

class ComponentProperty (*args)

Member properties of all SBOL objects are defined using a Property object.
The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** SBOL specification currently supports string, URI, and integer literal values.

- `python_iter` : `std::vector< std::string >::iterator`

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- `new_value` [] A new string which will be added to a list of values.

```cpp
addValidationRule(*args)
```

```cpp
clear()
```

Clear all property values.

```cpp
copy(target_property)
```

Copy property values to a target object’s property fields.

```cpp
find(query)
```

Check if a value in this property matches the query.

```cpp
getLowerBound()
```

```cpp
getOwner()
```

```cpp
getTypeURI()
```

```cpp
getUpperBound()
```

```cpp
remove(index=0)
```

Remove a property value.

```cpp
set(*args)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- `new_value` [] A new integer value for the property, which is converted to a raw string during serialization.

```cpp
validate(arg=None)
```

```cpp
write()
```
class Config
A class which contains global configuration variables for the libSBOL environment. Intended to be used like a
static class, configuration variables are accessed through the Config::setOptions and Config::getOption methods.
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/config.h

static getOption (option)
getOption(option) -> std::string
Get current option value for online validation and conversion.
  • option [] The option key

static setOption (*args)
setOption(option, value)

Config_getOption (option)
getOption(option) -> std::string
Get current option value for online validation and conversion.
  • option [] The option key

Config_setOption (*args)
setOption(option, value)

class Cut (*args)
The Cut class specifies a location between two coordinates of a Sequence’s elements. class Cut : public Location.
  • at [IntProperty] This property specifies the location between this nucleotide coordinate (or other sequence
element) and the nucleotide coordinate immediately following it. When equal to zero, the specified
region is immediately before the first nucleotide or character in the elements.
  • orientation [URIProperty] The orientation indicates how a region of double-stranded DNA represented
by the parent SequenceAnnotation and its associated Component are oriented.
The orientation may be one of the following values. By default it is set to
SBOL_ORIENTATION_INLINE. Orientation URI
libSBOL Symbol
http://sbols.org/v2#inline
SBOL_ORIENTATION_INLINE
http://sbols.org/v2#reverseComplement
SBOL_ORIENTATION_REVERSE_COMPLEMENT
  • persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of
URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each
other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity
URI.
  • displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String.
This property is intended to be an intermediate between name and identity that is machine-readable,
but more human-readable than the full URI of an identity. If the displayId property is used, then its
String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed
of only alphanumeric or underscore characters and MUST NOT begin with a digit.
  • version [VersionProperty] If the version property is used, then it is RECOMMENDED that version num-
bering follow the conventions of semantic versioning, particularly as implemented by Maven. This
convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, e.g., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/location.h

**class DateTimeProperty** (*args)
Contains a DateTime string following XML Schema.

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/properties.h

**stampTime()**

```
stampTime() -> std::string
```

Set this property with the current time.

**class Design** (*args)**
This class represents a biological Design. A Design is a conceptual representation of a biological system that a synthetic biologist intends to build. A Design is the first object created in libSBOL's formalized Design-Build-Test-Analysis workflow.

- **structure** [OwnedObject< ComponentDefinition >] The target biological structure for synthesis or other molecular assembly.

- **function** [OwnedObject< ModuleDefinition >] The intended function and predicted behavior of the Design object.

- **characterization** [ReferencedObject] A reference to an Analysis or multiple Analysis objects that contain characterization data, previously verified experimental knowledge, or explanatory models that inform a Design.
• attachments : ReferencedObject

• persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• version [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• wasDerivedFrom [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• wasGeneratedBy [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• name [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• description [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• identity [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/dbtl.h

```cpp
copy (*args)
copy(ns="", version="") -> SBOLClass &
```

class DesignProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

• The [] SBOL specification currently supports string, URI, and integer literal values.
• `python_iter`: `std::vector<std::string>::iterator`

C++ includes: `/Users/bbartley/Dev/git/libSBOL/source/property.h

`add(new_value)`
`add(new_value)`

Appends the new value to a list of values, for properties that allow it.

• `new_value` [] A new string which will be added to a list of values.

`addValidationRule(*args)`
`addValidationRule(rule)`

`clear()`
`clear()`

Clear all property values.

`copy(target_property)`
`copy(target_property)`

Copy property values to a target object’s property fields.

`find(query)`
`find(query) -> bool`

Check if a value in this property matches the query.

`getLowerBound()`
`getLowerBound() -> char`

`getOwner()`
`getOwner() -> SBOLObject &`

`getTypeURI()`
`getTypeURI() -> rdf_type`

`getUpperBound()`
`getUpperBound() -> char`

`remove(index=0)`
`remove(index=0)`

Remove a property value.

`set(*args)`
`set(new_value)`

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• `new_value` [] A new integer value for the property, which is converted to a raw string during serialization.

`validate(arg=None)`
`validate(arg=NULL)`

`write()`
`write()`

`class Document(*args)`
Read and write SBOL using a Document class. The Document is a container for Components, Modules, and all other SBOLObjects.

• `designs`: `OwnedObject<Design>`
• **builds**: `OwnedObject< Build >`
• **tests**: `OwnedObject< Test >`
• **analyses**: `OwnedObject< Analysis >`
• **componentDefinitions**: `OwnedObject< ComponentDefinition >`
• **moduleDefinitions**: `OwnedObject< ModuleDefinition >`
• **models**: `OwnedObject< Model >`
• **sequences**: `OwnedObject< Sequence >`
• **collections**: `OwnedObject< Collection >`
• **activities**: `OwnedObject< Activity >`
• **plans**: `OwnedObject< Plan >`
• **agents**: `OwnedObject< Agent >`
• **attachments**: `OwnedObject< Attachment >`
• **combinatorialDerivations**: `OwnedObject< CombinatorialDerivation >`
• **implementations**: `OwnedObject< Implementation >`
• **sampleRosters**: `OwnedObject< SampleRoster >`
• **citations**: `URIProperty`
• **keywords**: `URIProperty`
• **python_iter**: `iterator`

• **persistentIdentity** [`URIProperty` ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId** [`TextProperty` ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [`VersionProperty` ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [`URIProperty` ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [`ReferencedObject` ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.
• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/document.h

```c++
addComponentDefinition (*args)
    addComponentDefinition(sbol_obj)

addModel (*args)
    addModel(sbol_obj)

addModuleDefinition (*args)
    addModuleDefinition(sbol_obj)

addNamespace (*args)
    addNamespace(ns, prefix)

    Add a new namespace to this Document.

    • **ns** [] The namespace, eg. http://sbols.org/v2#

    • **prefix** [] The namespace prefix, eg. sbol

addSequence (*args)
    addSequence(sbol_obj)

append (filename)
    append(filename)

    Read an RDF/XML file and attach the SBOL objects to this Document.

    New objects will be added to the existing contents of the Document

    • **filename** [] The full name of the file you want to read (including file extension)

copy (*args)
    copy(ns="", version="") -> SBOLClass &

countTriples()
    countTriples() -> int

def() -> iterator

def (uri)
    find(uri) -> SBOLObject *

    Search recursively for an SBOLObject in this Document that matches the uri.

    • **uri** [] The identity of the object to search for
```
A pointer to the SBOLObject, or NULL if an object with this identity doesn’t exist

**find_property** (uri)

find_property(uri) -> SBOLObject *

Search this object recursively to see if it contains a member property with the given RDF type.

- **uri** [] The RDF type of the property to search for.

A pointer to the object that contains a member property with the specified RDF type, NULL otherwise

**find_reference** (uri)

find_reference(uri) -> std::vector< SBOLObject * >

Search this object recursively to see if it contains a member property with the given RDF type and indicated property value.

- **uri** [] A URI, either an ontology term or an object reference, to search for

A vector containing all objects found that contain the URI in a property value

**generate** (world, sbol_serializer, sbol_buffer, sbol_buffer_len, ios, base_uri)

generate() -> SBOLClass &

Retrieve an object from the Document.

- **uri** [] The identity of the SBOL object you want to retrieve

- **SBOLClass** [] The type of SBOL object

**getAgent** (uri)

get(uri) -> SBOLClass &

Retrieve an object from the Document.

- **uri** [] The identity of the SBOL object you want to retrieve

- **SBOLClass** [] The type of SBOL object

**getAnalysis** (uri)

get(uri) -> SBOLClass &

Retrieve an object from the Document.

- **uri** [] The identity of the SBOL object you want to retrieve

- **SBOLClass** [] The type of SBOL object

**getAttachment** (uri)

get(uri) -> SBOLClass &

Retrieve an object from the Document.

- **uri** [] The identity of the SBOL object you want to retrieve

- **SBOLClass** [] The type of SBOL object
getBuild\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
- \( SBOLClass \) [] The type of SBOL object

getCollection\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
- \( SBOLClass \) [] The type of SBOL object

getCombinatorialDerivation\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
- \( SBOLClass \) [] The type of SBOL object

getComponentDefinition\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
- \( SBOLClass \) [] The type of SBOL object

getDesign\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
- \( SBOLClass \) [] The type of SBOL object

getExperiment\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
- \( SBOLClass \) [] The type of SBOL object

getExperimentalData\( (uri) \)
\[get(uri) \to SBOLClass &\]
Retrieve an object from the Document.

- \( uri \) [] The identity of the SBOL object you want to retrieve
getImplementation (uri)
  get(uri) -> SBOLClass &
  Retrieve an object from the Document.
  • uri [] The identity of the SBOL object you want to retrieve
  • SBOLClass [] The type of SBOL object

getModel (uri)
  get(uri) -> SBOLClass &
  Retrieve an object from the Document.
  • uri [] The identity of the SBOL object you want to retrieve
  • SBOLClass [] The type of SBOL object

getModuleDefinition (uri)
  get(uri) -> SBOLClass &
  Retrieve an object from the Document.
  • uri [] The identity of the SBOL object you want to retrieve
  • SBOLClass [] The type of SBOL object

getNamespaces ()
  getNamespaces() -> std::vector< std::string >
  A vector of namespaces Get namespaces contained in this Document

getPlan (uri)
  get(uri) -> SBOLClass &
  Retrieve an object from the Document.
  • uri [] The identity of the SBOL object you want to retrieve
  • SBOLClass [] The type of SBOL object

getSampleRoster (uri)
  get(uri) -> SBOLClass &
  Retrieve an object from the Document.
  • uri [] The identity of the SBOL object you want to retrieve
  • SBOLClass [] The type of SBOL object

getSequence (uri)
  get(uri) -> SBOLClass &
  Retrieve an object from the Document.
  • uri [] The identity of the SBOL object you want to retrieve
  • SBOLClass [] The type of SBOL object
getTest (uri)
get(uri) -> SBOLClass &
Retrieve an object from the Document.

• uri [] The identity of the SBOL object you want to retrieve

• SBOLClass [] The type of SBOL object

query_repository (command)
query_repository(command) -> std::string

read (filename)
read(filename)
Read an RDF/XML file and attach the SBOL objects to this Document.
Existing contents of the Document will be wiped.

• filename [] The full name of the file you want to read (including file extension)

readString (*args)
readString(sbol)
Convert text in SBOL into data objects.

• sbol [] A string formatted in SBOL

request_comparison (diff_file)
request_comparison(diff_file) -> std::string
Perform comparison on Documents using the online validation tool.
This is for cross-validation of SBOL documents with libSBOLj. Document comparison can also be performed using the built-in compare method.
The comparison results

request_validation (sbol)
request_validation(sbol) -> std::string

search_metadata (role, type, name, collection)
search_metadata(role, type, name, collection) -> std::string

summary ()
summary() -> std::string
Get a summary of objects in the Document, including SBOL core object and custom annotation objects.

validate ()
validate() -> std::string
Run validation on this Document via the online validation tool.
A string containing a message with the validation results
author: KC

write (filename)
write(filename) -> std::string
Serialize all objects in this Document to an RDF/XML file.

• filename [] The full name of the file you want to write (including file extension)
A string with the validation results, or empty string if validation is disabled
**writeString()**

**writeString() -> std::string**

Convert data objects in this Document into textual SBOL.

**class EnzymeCatalysisInteraction (**\*args**

- **enzyme** : *AliasedProperty*< *FunctionalComponent* |
- **substrates** : *AliasedProperty*< *FunctionalComponent* |
- **products** : *AliasedProperty*< *FunctionalComponent* |
- **cofactors** : *AliasedProperty*< *FunctionalComponent* |
- **sideproducts** : *AliasedProperty*< *FunctionalComponent* |

- **types** [ *URIProperty* ] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

  The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See [http://www.ebi.ac.uk/sbo/main/](http://www.ebi.ac.uk/sbo/main/)) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type

SBO_CONTROL

• **participations** [OwnedObject< Participation >] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer might want to specify that an Interaction will exist, even if its participants have not yet been determined.

• **functionalComponents** : OwnedObject< FunctionalComponent >

• **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

class Experiment(*args)
**copy** (*args)

```python
    copy(ns='', version='') -> SBOLClass &
```

**class** `ExperimentProperty` (*args*)

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter** : `std::vector< std::string >::iterator`

C++ includes: `/Users/bbartley/Dev/git/libSBOL/source/property.h`

**add** (*new_value*)

```python
    add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

**addValidationRule** (*args*)

```python
    addValidationRule(rule)
```

**clear** ()

```python
    clear()
```

Clear all property values.

**copy** (target_property)

```python
    copy(target_property)
```

Copy property values to a target object’s property fields.

**find** (query)

```python
    find(query) -> bool
```

Check if a value in this property matches the query.

**getLowerBound** ()

```python
    getLowerBound() -> char
```

**getOwner** ()

```python
    getOwner() -> SBOLObject &
```

**getTypeURI** ()

```python
    getTypeURI() -> rdf_type
```

**getUpperBound** ()

```python
    getUpperBound() -> char
```

**remove** (index=0)

```python
    remove(index=0)
```

Remove a property value.

**set** (*args*)

```python
    set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.
validate \texttt{(arg=None)}
\hspace{2em} validate\texttt{(arg=NULL)}

\texttt{write()}
\hspace{2em} write()

\textbf{class} \texttt{ExperimentalData} (*\texttt{args})*

\texttt{copy (*\texttt{args})}
\hspace{2em} \texttt{copy(ns="", version="" \rightarrow SBOLClass &}

\textbf{class} \texttt{ExperimentalDataProperty} (*\texttt{args})*

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

\begin{itemize}
  \item \textit{The} [] SBOL specification currently supports string, URI, and integer literal values.
  \item python\_iter : std::vector< std::string >::\texttt{iterator}
\end{itemize}

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

\texttt{add (new\_value)}
\hspace{2em} \texttt{add(new\_value)}

Appends the new value to a list of values, for properties that allow it.

\begin{itemize}
  \item \textit{new\_value} [] A new string which will be added to a list of values.
\end{itemize}

\texttt{addValidationRule (*\texttt{args})}
\hspace{2em} \texttt{addValidationRule(rule)}

\texttt{clear ()}
\hspace{2em} \texttt{clear()}

Clear all property values.

\texttt{copy (target\_property)}
\hspace{2em} \texttt{copy(target\_property)}

Copy property values to a target object’s property fields.

\texttt{find (query)}
\hspace{2em} \texttt{find(query) \rightarrow bool}

Check if a value in this property matches the query.

\texttt{getLowerBound ()}
\hspace{2em} \texttt{getLowerBound() \rightarrow char}

\texttt{getOwner ()}
\hspace{2em} \texttt{getOwner() \rightarrow SBOLObject &}

\texttt{getTypeURI ()}
\hspace{2em} \texttt{getTypeURI() \rightarrow rdf\_type}

\texttt{getUpperBound ()}
\hspace{2em} \texttt{getUpperBound() \rightarrow char}

\texttt{remove (index=0)}
\hspace{2em} \texttt{remove(index=0)}

Remove a property value.
**set** (*args)

```
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**validate** (arg=None)

```
validate(arg=NULL)
```

**write**()

```
write()
```

**class FloatProperty (*args)**

FloatProperty objects are used to contain floats.

They can be used as member objects inside custom SBOL Extension classes.

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/properties.h

**get**()

```
get() -> double
```

Get the float value.

An integer

**getAll**()

```
getAll() -> std::vector< double >
```

**class FunctionalComponent (*args)**

The FunctionalComponent class is used to specify the functional usage of a ComponentDefinition inside a ModuleDefinition. The ModuleDefinition describes how the that describes how the FunctionalComponent interacts with others and summarizes their aggregate function.

- **direction** [URIProperty ] Each FunctionalComponent MUST specify via the direction property whether it serves as an input, output, both, or neither for its parent ModuleDefinition object.

The value for this property MUST be one of the URIs given in the table below. Direction URI Description

LibSBOL Symbol

- http://sbols.org/v2#in
  Indicating that the FunctionalComponent is an input.

- http://sbols.org/v2#out
  Indicates that the FunctionalComponent is an output.

- http://sbols.org/v2#inout
  Indicates that the FunctionalComponent is both an input and output.

- http://sbols.org/v2#none
  Indicates that the FunctionalComponent is neither an input or output.
SBOL_DIRECTION_NONE

- **definition** [ReferencedObject ] The definition property is a REQUIRED URI that refers to the ComponentDefinition of the ComponentInstance.
  
  As described in the previous section, this ComponentDefinition effectively provides information about the types and roles of the ComponentInstance. The definition property MUST NOT refer to the same ComponentDefinition as the one that contains the ComponentInstance. Furthermore, ComponentInstance objects MUST NOT form a cyclical chain of references via their definition properties and the ComponentDefinition objects that contain them. For example, consider the ComponentInstance objects A and B and the ComponentDefinition objects X and Y. The reference chain X contains A, A isdefinedby Y, Y contains B, and B isdefinedby X iscyclical.

- **access** [URIProperty ] The access property is a REQUIRED URI that indicates whether the ComponentInstance can be referred to remotely by a MapsTo.
  
  The value of the access property MUST be one of the following URIs. Access URI Description
  
<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://sbols.org/v2#public">http://sbols.org/v2#public</a></td>
<td>The ComponentInstance MAY be referred to by remote MapsTo objects</td>
</tr>
<tr>
<td><a href="http://sbols.org/v2#private">http://sbols.org/v2#private</a></td>
<td>The ComponentInstance MAY be referred to by remote MapsTo objects</td>
</tr>
</tbody>
</table>

- **mapStos** [OwnedObject< MapsTo > ] The mapsTos property is OPTIONAL and MAY contain a set of MapsTo objects that refer to and link together ComponentInstance objects (both Component objects and FunctionalComponent objects) within a larger design.

- **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.
• **name**  [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description**  [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity**  [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/component.h

```cpp
connect (interface_component)
connect(interface_component)
```

This method connects module inputs and outputs.

This convenience method auto-constructs a MapsTo object. See Biosystem Design for an example

• **interface_component** [] An input or output component from another ModuleDefinition that corresponds with this component.

```cpp
isMasked ()
isMasked() -> int
```

Used to tell if a FunctionalComponent is linked to an equivalent FunctionalComponent in another ModuleDefinition.

1 if the FunctionalComponent has been over-rided by another FunctionalComponent, 0 if it hasn’t.

```cpp
mask (masked_component)
mask(masked_component)
```

This method is used to state that FunctionalComponents in separate ModuleDefinitions are functionally equivalent.

Using this method will override the FunctionalComponent in the argument with the FunctionalComponent calling the method. This is useful for overriding a generic, template component with an explicitly defined component. This convenience method auto-constructs a MapsTo object. See Biosystem Design for an example

• **masked_component** [] The FunctionalComponent that is being masked (over-ridden)

```cpp
override (masked_component)
override(masked_component)
```

This method is used to state that FunctionalComponents in separate ModuleDefinitions are functionally equivalent.

Using this method will override the FunctionalComponent in the argument with the FunctionalComponent calling the method. This is useful for overriding a generic, template component with an explicitly defined component. This convenience method auto-constructs a MapsTo object. See Biosystem Design for an example

• **masked_component** [] The FunctionalComponent that is being masked (over-ridden)
class FunctionalComponentProperty (*args)
    Member properties of all SBOL objects are defined using a Property object.

    The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class
    converts SBOL data structures into RDF triples.

    • The [] SBOL specification currently supports string, URI, and integer literal values.

    • python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add (new_value)
    add(new_value)

    Appends the new value to a list of values, for properties that allow it.

    • new_value [] A new string which will be added to a list of values.

addValidationRule (*args)
    addValidationRule(rule)

clear ()
    clear()

    Clear all property values.

copy (target_property)
    copy(target_property)

    Copy property values to a target object’s property fields.

find (query)
    find(query) -> bool

    Check if a value in this property matches the query.

getLowerBound ()
    getLowerBound() -> char

getOwner ()
    getOwner() -> SBOLObject &

getTypeURI ()
    getTypeURI() -> rdf_type

getUpperBound ()
    getUpperBound() -> char

remove (index=0)
    remove(index=0)

    Remove a property value.

set (*args)
    set(new_value)

    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

    • new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
    validate(arg=NULL)
write()
write()

class GeneProductionInteraction (uri, gene, product)
  • gene : AliasedProperty< FunctionalComponent >
  • product : AliasedProperty< FunctionalComponent >
  • types [URIProperty ] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

    The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See http://www.ebi.ac.uk/sbo/main/) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type

<table>
<thead>
<tr>
<th>URI for SBO Term</th>
<th>LibSBOL symbol</th>
<th>Biochemical Reaction</th>
<th><a href="http://identifiers.org/biomodels.sbo/SBO:0000176">http://identifiers.org/biomodels.sbo/SBO:0000176</a></th>
<th>SBO_BIOCHEMICAL_REACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulation</td>
<td></td>
<td>Inhibition</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000169">http://identifiers.org/biomodels.sbo/SBO:0000169</a></td>
<td>SBO_INHIBITION</td>
</tr>
<tr>
<td>Genesic Production</td>
<td></td>
<td>Degradation</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000179">http://identifiers.org/biomodels.sbo/SBO:0000179</a></td>
<td>SBO_DEGRADATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000168">http://identifiers.org/biomodels.sbo/SBO:0000168</a></td>
<td>SBO_CONTROL</td>
</tr>
</tbody>
</table>

• participations [OwnedObject< Participation > ] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer
might want to specify that an Interaction will exist, even if its participants have not yet been determined.

- **functionalComponents** : OwnedObject< FunctionalComponent >

- **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

**class** GenericLocation (*args)

the GenericLocation class is included as a starting point for specifying regions on Sequence objects with encoding properties other than IUPAC and potentially nonlinear structure. This class can also be used to set the orientation of a SequenceAnnotation and any associated Component when their parent ComponentDefinition is a partial design that lacks a Sequence.

- **orientation** [URIProperty ] The orientation indicates how a region of double-stranded DNA represented by the parent SequenceAnnotation and its associated Component are oriented.
The orientation may be one of the following values. By default it is set to SBOL_ORIENTATION_INLINE. Orientation URI
libSBOL Symbol
http://sbols.org/v2#inline
SBOL_ORIENTATION_INLINE
http://sbols.org/v2#reverseComplement
SBOL_ORIENTATION_REVERSE_COMPLEMENT

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/location.h
class Identified(*args)

All SBOL-defined classes are directly or indirectly derived from the Identified abstract class.

An Identified object is identified using a Uniform Resource Identifier (URI), a unique string that identifies and refers to a specific object in an SBOL document or in an online resource such as a DNA repository.

• persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object must be referred to using either its identity URI or its persistentIdentity URI.

• displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value should be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• version [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• wasDerivedFrom [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• wasGeneratedBy [ReferencedObject] An Activity which generated this ComponentDefinition, e.g., a design process like codon-optimization or a construction process like Gibson Assembly.

• name [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools should instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• description [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• identity [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/identified.h

class Implementation(*args)

An Implementation represents a real, physical instance of a synthetic biological construct which may be associated with a laboratory sample. An Implementation may be linked back to its original design (either a ModuleDefinition or ComponentDefinition) using the wasDerivedFrom property inherited from the Identified superclass.
• **built** : URIProperty

• **attachments** : ReferencedObject

• **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId** [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/implementation.h

```cpp
void copy ( *args)
    copy(ns="", version=""") -> SBOLClass &

class ImplementationProperty ( *args)
    Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.
```
• **The** [] SBOL specification currently supports string, URI, and integer literal values.

• **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add(new_value)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

• **new_value []** A new string which will be added to a list of values.

```cpp
addValidationRule(*args)
addValidationRule(rule)
```

```cpp
clear()
clear()
```

Clear all property values.

```cpp
copy(target_property)
copy(target_property)
```

Copy property values to a target object’s property fields.

```cpp
find(query)
find(query) -> bool
```

Check if a value in this property matches the query.

```cpp
getLowerBound()
getLowerBound() -> char
```

```cpp
getOwner()
getOwner() -> SBOLObject &
```

```cpp
getTypeURI()
getTypeURI() -> rdf_type
```

```cpp
getUpperBound()
getUpperBound() -> char
```

```cpp
remove(index=0)
remove(index=0)
```

Remove a property value.

```cpp
set(*args)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value []** A new integer value for the property, which is converted to a raw string during serialization.

```cpp
validate(arg=None)
validate(arg=NULL)
```

```cpp
write()
write()
```

```cpp
class IntProperty(*args)
```

IntProperty objects are used to contain integers.
They can be used as member objects inside custom SBOL Extension classes.

- `python_iter : std::vector< std::string >::iterator`

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/properties.h

```cpp
get ()
get() -> int
Get the integer value.
An integer

getAll ()
getAll() -> std::vector< int >
```

**class Interaction (**args**)

The Interaction class provides more detailed description of how the Functional Components are intended to work together. For example, this class can be used to represent different forms of genetic regulation (e.g., transcriptional activation or repression), processes from the central dogma of biology (e.g., transcription and translation), and other basic molecular interactions (e.g., non-covalent binding or enzymatic phosphorylation).

- `types [URIProperty ] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

  The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See http://www.ebi.ac.uk/sbo/main/) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type

<table>
<thead>
<tr>
<th>LibSBOL symbol</th>
<th>URI for SBO Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biochemical Reaction</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000176">http://identifiers.org/biomodels.sbo/SBO:0000176</a></td>
</tr>
<tr>
<td>SBO_BIOCHEMICAL_REACTION</td>
<td></td>
</tr>
<tr>
<td>Inhibition</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000169">http://identifiers.org/biomodels.sbo/SBO:0000169</a></td>
</tr>
<tr>
<td>SBO_INHIBITION</td>
<td></td>
</tr>
<tr>
<td>Stimulation</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000170">http://identifiers.org/biomodels.sbo/SBO:0000170</a></td>
</tr>
<tr>
<td>SBO_STIMULATION</td>
<td></td>
</tr>
<tr>
<td>Genetic Production</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000589">http://identifiers.org/biomodels.sbo/SBO:0000589</a></td>
</tr>
<tr>
<td>SBO_GENETIC_PRODUCTION</td>
<td></td>
</tr>
<tr>
<td>Non-Covalent Binding</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000177">http://identifiers.org/biomodels.sbo/SBO:0000177</a></td>
</tr>
<tr>
<td>SBO_NONCOVALENT_BINDING</td>
<td></td>
</tr>
<tr>
<td>Degradation</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000179">http://identifiers.org/biomodels.sbo/SBO:0000179</a></td>
</tr>
</tbody>
</table>
```
SBO_DEGRADATION

Control

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

SBO_CONTROL

- **participations** [OwnedObject< Participation >] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer might want to specify that an Interaction will exist, even if its participants have not yet been determined.

- **functionalComponents** : OwnedObject< FunctionalComponent >

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, e.g., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within
this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/interaction.h

class InteractionProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

  - python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add(new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

addValidationRule (*args)
addValidationRule(rule)

clear ()
clear()

Clear all property values.

copy (target_property)
copy(target_property)

Copy property values to a target object’s property fields.

find (query)
find(query) -> bool

Check if a value in this property matches the query.

getLowerBound ()
getLowerBound() -> char

getOwner ()
getOwner() -> SBOLObject &

getTypeURI ()
g getTypeURI() -> rdf_type

getUpperBound ()
g getUpperBound() -> char

remove (index=0)
remove(index=0)

Remove a property value.

set (*args)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```python
validate(arg=None)
validate(arg=NULL)
```

```python
write()
write()
```

class Location (*args)

The Location class specifies the strand orientation of a Component and can be further extended by the Range, Cut, and GenericLocation classes.

• **orientation** [URIProperty] The orientation indicates how a region of double-stranded DNA represented by the parent SequenceAnnotation and its associated Component are oriented.

The orientation may be one of the following values. By default it is set to SBOL_ORIENTATION_INLINE. Orientation URI

<table>
<thead>
<tr>
<th>libSBOL Symbol</th>
<th>SBOL Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://sbols.org/v2#inline">http://sbols.org/v2#inline</a></td>
<td>SBOL_ORIENTATION_INLINE</td>
</tr>
<tr>
<td><a href="http://sbols.org/v2#reverseComplement">http://sbols.org/v2#reverseComplement</a></td>
<td>SBOL_ORIENTATION_REVERSE_COMPLEMENT</td>
</tr>
</tbody>
</table>

• **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name
properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/location.h

```cpp
class LocationProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The []** SBOL specification currently supports string, URI, and integer literal values.

- **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add(new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value []** A new string which will be added to a list of values.

addValidationRule(*args)
addValidationRule(rule)

clear()
clear()

Clear all property values.

copy(target_property)
copy(target_property)

Copy property values to a target object’s property fields.

find(query)
find(query) -> bool

Check if a value in this property matches the query.

getLowerBound()
getLowerBound() -> char

getOwner()
getOwner() -> SBOLObject &

getTypeURI()
getTypeURI() -> rdf_type

getUpperBound()
getUpperBound() -> char
remove(index=0)
remove(index=0)

Remove a property value.

set(*args)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

* new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
validate(arg=NULL)

write()
write()

class MapsTo(*args)

The purpose of the MapsTo class is to make identity relationships between different ComponentInstances in functional and structural hierarchies more clear. For example, a MapsTo object may be used to connect outputs and inputs between different low-level ModuleDefinitions contained in a higher level Module Definition. A MapsTo object may also be used to override a generic Component in a low-level ModuleDefinition with an explicit Component in a high-level ModuleDefinition, for example mapping a generic gene to an explicit component with a name and sequence.

* refinement [URIProperty ] Each MapsTo object MUST specify the relationship between its local and remote ComponentInstance objects using one of the REQUIRED refinement URIs provided in the table below.

Refinement URI
libSBOL Symbol
Description

http://sbols.org/v2#useRemote
SBOL_REFINEMENT_USE_REMOTE
All references to the local ComponentInstance MUST dereference to the remote ComponentInstance instead.

http://sbols.org/v2#useLocal
SBOL_REFINEMENT_USE_LOCAL
In the context of the ComponentDefinition or ModuleDefinition that contains the owner of the MapsTo, all references to the remote ComponentInstance MUST dereference to the local ComponentInstance instead.

http://sbols.org/v2#verifyIdentical
SBOL_REFINEMENT_VERIFY_IDENTICAL
The definition properties of the local and remoteComponentInstance objects MUST refer to the same ComponentDefinition.

http://sbols.org/v2#merge
SBOL_REFINEMENT_MERGE_DESCRIPTION
In the context of the ComponentDefinition or ModuleDefinition that contains the owner of the Mapsto, all references to the local ComponentInstance or the remote ComponentInstance MUST dereference to both objects.

- **local** [ReferencedObject] The identity of the lower level ComponentInstance.
- **remote** [ReferencedObject] The identity of the higher level ComponentInstance.
- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.
- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.
- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.
- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.
- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.
- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.
- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/mapsto.h

```cpp
class MapsToProperty (*args)
    Member properties of all SBOL objects are defined using a Property object.
```

Chapter 8. API
The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class
converts SBOL data structures into RDF triples.

- The SBOL specification currently supports string, URI, and integer literal values.
- python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/dev/git/libSBOL/source/property.h

```cpp
def add(new_value)
    add(new_value)

    Appends the new value to a list of values, for properties that allow it.

    • new_value [] A new string which will be added to a list of values.
```

```cpp
def addValidationRule(*args)
    addValidationRule(rule)
```

```cpp
def clear()
    clear()

    Clear all property values.
```

```cpp
def copy(target_property)
    copy(target_property)

    Copy property values to a target object’s property fields.
```

```cpp
def find(query)
    find(query) -> bool

    Check if a value in this property matches the query.
```

```cpp
def getLowerBound()
    getLowerBound() -> char
```

```cpp
def getOwner()
    getOwner() -> SBOLObject &
```

```cpp
def getTypeURI()
    getTypeURI() -> rdf_type
```

```cpp
def getUpperBound()
    getUpperBound() -> char
```

```cpp
def remove(index=0)
    remove(index=0)

    Remove a property value.
```

```cpp
def set(*args)
    set(new_value)

    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

    • new_value [] A new integer value for the property, which is converted to a raw string during serialization.
```

```cpp
def validate(arg=None)
    validate(arg=NULL)
```

```cpp
def write()
    write()
```
Member properties of all SBOL objects are defined using a Property object. The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The []** SBOL specification currently supports string, URI, and integer literal values.

- **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

**add (new_value)**

add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value []** A new string which will be added to a list of values.

**addValidationRule (args)**

addValidationRule(rule)

**clear ()**

clear()

Clear all property values.

**copy (target_property)**

copy(target_property)

Copy property values to a target object’s property fields.

**find (query)**

find(query) -> bool

Check if a value in this property matches the query.

**getLowerBound ()**

getLowerBound() -> char

**getOwner ()**

gowner() -> SBOLObject &

**getTypeURI ()**

getTypeURI() -> rdf_type

**getUpperBound ()**

getUpperBound() -> char

**remove (index=0)**

remove(index=0)

Remove a property value.

**set (args)**

set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value []** A new integer value for the property, which is converted to a raw string during serialization.
validate(arg=None)
validate(arg=NULL)

write()
write()

class Model(*args)
The purpose of the Model class is to serve as a placeholder for an external computational model and provide additional meta-data to enable better reasoning about the contents of this model.

In this way, there is minimal duplication of standardization efforts and users of SBOL can formalize the function of a ModuleDefinition in the language of their choice.

- **source** [URIProperty ] The source property is REQUIRED and MUST contain a URI reference to the source file for a model.

- **language** [URIProperty ] The language property is REQUIRED and MUST contain a URI that specifies the language in which the model is implemented.

  It is RECOMMENDED that this URI refer to a term from the EMBRACE Data and Methods (EDAM) ontology. The Table below provides a list of terms from this ontology and their URIs. If the language property of a Model is well-described by one these terms, then it MUST contain the URI for this term as its value.

<table>
<thead>
<tr>
<th>Model Language</th>
<th>URI for EDAM Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>libSBOL Symbol</td>
<td>SBML</td>
</tr>
<tr>
<td></td>
<td><a href="http://identifiers.org/edam/format_2585">http://identifiers.org/edam/format_2585</a></td>
</tr>
<tr>
<td></td>
<td>EDAM_SBML</td>
</tr>
<tr>
<td></td>
<td>CellML</td>
</tr>
<tr>
<td></td>
<td><a href="http://identifiers.org/edam/format_3240">http://identifiers.org/edam/format_3240</a></td>
</tr>
<tr>
<td></td>
<td>EDAM_CELLML</td>
</tr>
<tr>
<td></td>
<td>BioPAX</td>
</tr>
<tr>
<td></td>
<td><a href="http://identifiers.org/edam/format_3156">http://identifiers.org/edam/format_3156</a></td>
</tr>
<tr>
<td></td>
<td>EDAM_BIOPAX</td>
</tr>
</tbody>
</table>

- **framework** [URIProperty ] Model Language

  - URI for SBO Term
  - libSBOL Symbol
  - Continuous
    - http://identifiers.org/biomodels.sbo/SBO:0000062
    - SBO_CONTINUOUS
  - Discrete
    - http://identifiers.org/biomodels.sbo/SBO:0000063
    - SBO_DISCRETE

- **attachments** : ReferencedObject
• persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• version [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• wasDerivedFrom [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• wasGeneratedBy [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• name [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• description [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• identity [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification doucment.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/model.h

```
copy(*args)
copy(ns="", version="") -> SBOLClass &
```

class ModelProperty(*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

• The [] SBOL specification currently supports string, URI, and integer literal values.

• python_iter : std::vector<std::string>::iterator
add (new_value)
    add(new_value)
    Appends the new value to a list of values, for properties that allow it.
    * new_value [] A new string which will be added to a list of values.

addValidationRule (*args)
    addValidationRule(rule)

clear()
    clear()
    Clear all property values.

copy (target_property)
    copy(target_property)
    Copy property values to a target object's property fields.

find (query)
    find(query) -> bool
    Check if a value in this property matches the query.

getLowerBound ()
    getLowerBound() -> char

getOwner ()
    getOwner() -> SBOLObject &

getTypeURI ()
    getTypeURI() -> rdf_type

getUpperBound ()
    getUpperBound() -> char

remove (index=0)
    remove(index=0)
    Remove a property value.

set (*args)
    set(new_value)
    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
    * new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
    validate(arg=NULL)

write ()
    write()

class Module (*args)
The Module class represents a submodule of a ModuleDefinition within a hierarchical design.
    * definition [ReferencedObject] The definition property is a REQUIRED URI that refers to the ModuleDefinition for the Module.
The mapsTos property is an OPTIONAL set of MapsTo objects that refer to and link ComponentInstance objects together within the hierarchy of Module, ModuleDefinition, ComponentInstance, and ComponentDefinition objects.

The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

An Activity which generated this ComponentDefinition, e.g., a design process like codon-optimization or a construction process like Gibson Assembly.

The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/module.h

class ModuleDefinition (*args)

The ModuleDefinition class represents a grouping of structural and functional entities in a biological design. The primary usage of this class is to assert the molecular interactions and abstract function of its child entities.

The roles property is an OPTIONAL set of URIs that clarifies the intended function of a ModuleDefinition. These URIs might identify descriptive biological roles, such as metabolic pathway and signaling cascade, but they can also identify identify logical roles, such as inverter or
AND gate, or other abstract roles for describing the function of design. Interpretation of the meaning of such roles currently depends on the software tools that read and write them.

- **modules** \([\text{OwnedObject}<\text{Module}>]\) The modules property is OPTIONAL and MAY specify a set of Module objects contained by the ModuleDefinition. While the ModuleDefinition class is analogous to a specification sheet for a system of interacting biological elements, the Module class represents the occurrence of a particular subsystem within the system. Hence, this class allows a system design to include multiple instances of a subsystem, all defined by reference to the same ModuleDefinition. For example, consider the ModuleDefinition for a network of two-input repressor devices in which the particular repressors have not been chosen yet. This ModuleDefinition could contain multiple Module objects that refer to the same ModuleDefinition of an abstract two-input repressor device. Note that the set of relations between Module and ModuleDefinition objects is strictly acyclic.

- **interactions** \([\text{OwnedObject}<\text{Interaction}>]\) The interactions property is OPTIONAL and MAY specify a set of Interaction objects within the ModuleDefinition. The Interaction class provides an abstract, machine-readable representation of entity behavior within a ModuleDefinition. Each Interaction contains Participation objects that indicate the roles of the FunctionalComponent objects involved in the Interaction.

- **functionalComponents** \([\text{OwnedObject}<\text{FunctionalComponent}>]\) The functionalComponents property is OPTIONAL and MAY specify a set of FunctionalComponent objects contained by the ModuleDefinition. Just as a Module represents an instance of a subsystem in the overall system represented by a ModuleDefinition, a FunctionalComponent represents an instance of a structural entity (represented by a ComponentDefinition) in the system. This concept allows a ModuleDefinition to assert different interactions for separate copies of the same structural entity if needed. For example, a ModuleDefinition might contain multiple FunctionalComponent objects that refer to the same promoter ComponentDefinition, but assert different interactions for these promoter copies based on their separate positions in another ComponentDefinition that represents the structure of the entire system.

- **models** \([\text{ReferencedObject}]\) The models property is OPTIONAL and MAY specify a set of URI references to Model objects. Model objects are placeholders that link ModuleDefinition objects to computational models of any format. A ModuleDefinition object can link to more than one Model since each might encode system behavior in a different way or at a different level of detail.

- **attachments** : \[\text{ReferencedObject}\]

- **persistentIdentity** \[\text{URIProperty}\] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** \[\text{TextProperty}\] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** \[\text{VersionProperty}\] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** \[\text{URIProperty}\] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own
wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/moduledefinition.h

```cpp
assemble (*args)
assemble(list_of_modules)
```

Assemble a high-level ModuleDefinition from lower-level submodules.

```cpp
connect (output, input)
connect(output, input)
```

Connects the output of a sub-Module with the input of another sub-Module.

```cpp
list_of_modules [] A list of pointers to the submodule ModuleDefinitions
```

```cpp
setInput (input)
```

Configures a FunctionalComponent as an input for a Module.

```cpp
Useful for bottom-up assembly of Modules and sub-Modules
```
• **input** [] The FunctionalComponent that will be configured

```cpp
setOutput (*args)
setOutput(output)
```

Configures a FunctionalComponent as an output for a Module.

Useful for bottom-up assembly of Modules and sub-Modules.

• **output** [] The FunctionalComponent that will be configured

```cpp
class ModuleDefinitionProperty (*args)
```

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

• **The** [] SBOL specification currently supports string, URI, and integer literal values.

• **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add (new_value)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

```cpp
addValidationRule (*args)
addValidationRule(rule)
```

```cpp
clear ()
clear()
```

Clear all property values.

```cpp
copy (target_property)
copy(target_property)
```

Copy property values to a target object’s property fields.

```cpp
find (query)
find(query) -> bool
```

Check if a value in this property matches the query.

```cpp
getLowerBound ()
getLowerBound() -> char
```

```cpp
getOwner ()
getOwner() -> SBOLObject &
```

```cpp
getTypeURI ()
gTypeURI() -> rdf_type
```

```cpp
getUpperBound ()
gUpperBound() -> char
```

```cpp
remove (index=0)
remove(index=0)
```

Remove a property value.
set(*args)
    set(new_value)
    
    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
    
    • **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
    validate(arg=NULL)

write()
    write()

class ModuleProperty(*args)
    Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

• **The** [] SBOL specification currently supports string, URI, and integer literal values.

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add(new_value)
    add(new_value)
    
    Appends the new value to a list of values, for properties that allow it.
    
    • **new_value** [] A new string which will be added to a list of values.

addValidationRule(*args)
    addValidationRule(rule)

clear()
    clear()
    
    Clear all property values.

copy(target_property)
    copy(target_property)
    
    Copy property values to a target object's property fields.

find(query)
    find(query) -> bool
    
    Check if a value in this property matches the query.

getLowerBound()
    getLowerBound() -> char

getOwner()
    getOwner() -> SBOLObject &

getTypeURI()
    getTypeURI() -> rdf_type

getUpperBound()
    getUpperBound() -> char
remove(index=0)
remove(index=0)

Remove a property value.

set(*args)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
validate(arg=NULL)

write()
write()

class OwnedActivity(*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• SBOLClass [] The type of child SBOL object contained by this Property

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

add(sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• new_value [] A new string which will be added to a list of values.

clear()
clear()

Remove all children objects from the parent and destroy them.

create(uri)
create(uri) -> Test &

createCut(uri)
createCut(uri) -> Test &

createGenericLocation(uri)
createGenericLocation(uri) -> Test &

createRange(uri)
createRange(uri) -> Test &

define(definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values
specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

```python
find(uri)
find(uri) -> bool
```

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

```python
get(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getAll()
getAll() -> std::vector<SBOLClass *>
```

Get all the objects contained in the property.

A vector of pointers to the objects

```python
getCut(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getRange (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLEntries to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove (*args)
remove(index=0)

Remove an object from the list of objects and destroy it.

• index [] A numerical index for the object.

set (sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• new_value [] A new integer value for the property, which is converted to a raw string during serialization.

class OwnedAgent (*args)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• SBOLClass [] The type of child SBOL object contained by this Property

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

add (sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• new_value [] A new string which will be added to a list of values.

clear ()
clear()

Remove all children objects from the parent and destroy them.

create (uri)
create(uri) -> Test &
createCut (uri)
    create(uri) -> Test &

createGenericLocation (uri)
    create(uri) -> Test &

createRange (uri)
    create(uri) -> Test &

define (definition_object)
    define(definition_object) -> SBOLClass &

    Autoconstructs a child object and attaches it to the parent object. Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

    • SBOLClass [] The type of SBOL object that will be created

    • definition_object [] The returned object will reference the definition_object in its definition property.

    A reference to the child object check uniqueness of URI in Document

find (uri)
    find(uri) -> bool

    • uri [] The full uniform resource identifier of the object to search for in this property

    A boolean indicating whether found or not

going (*args)
    get(uri="") -> SBOLSubClass &

    Get the child object.

    • SBOLClass [] The type of the child object

    • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

    • uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

    A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getAll ()
    getAll() -> std::vector<SBOLClass *>

    Get all the objects contained in the property.

    A vector of pointers to the objects

goingCut (*args)
    get(uri="") -> SBOLSubClass &

    Get the child object.

    • SBOLClass [] The type of the child object

    • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getGenericLocation** (*args*)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getRange** (*args*)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**remove** (*args*)
remove(index=0)

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

**set** (sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedAnalysis** (*args*)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter : std::vector< std::string >::iterator**
add (sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• new_value [] A new string which will be added to a list of values.

clear ()
clear()

Remove all children objects from the parent and destroy them.

create (uri)
create(uri) -> Test &

createCut (uri)
create(uri) -> Test &

createGenericLocation (uri)
create(uri) -> Test &

createRange (uri)
create(uri) -> Test &

define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• SBOLClass [] The type of SBOL object that will be created.

• definition_object [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document.

find (uri)
find(uri) -> bool

• uri [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not.

get (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

• SBOLClass [] The type of the child object.

• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object. Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
getAll ()
getAll() -> std::vector< SBOLClass * >
Get all the objects contained in the property.
A vector of pointers to the objects

getCut (*args)
getcut(uri="") -> SBOLSubClass &
Get the child object.
• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

genericLocation (*args)
genericlocation(uri="") -> SBOLSubClass &
Get the child object.
• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getRange (*args)
getrange(uri="") -> SBOLSubClass &
Get the child object.
• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove (*args)
remove(index=0)
Remove an object from the list of objects and destroy it.
• index [] A numerical index for the object.

set (sbol_obj)
set(new_value)
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedAssociation(***args*)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter**: `std::vector< std::string >::iterator`

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

add(sbol_obj)

`add(new_value)`

Appends the new value to a list of values, for properties that allow it.

  - **new_value** [] A new string which will be added to a list of values.

clear()

clear()

Remove all children objects from the parent and destroy them.

create(uri)

`create(uri) -> Test &`

createCut(uri)

`createCut(uri) -> Test &`

createGenericLocation(uri)

`createGenericLocation(uri) -> Test &`

createRange(uri)

`createRange(uri) -> Test &`

define(definition_object)

`define(definition_object) -> SBOLClass &`

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

  - **SBOLClass** [] The type of SBOL object that will be created

  - **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

find(uri)

`find(uri) -> bool`

  - **uri** [] The full uniform resource identifier of the object to search for in this property
A boolean indicating whether found or not

**get** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll** ()

getAll() : std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

**getCut** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getGenericLocation** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getRange** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove (*args)
remove(index=0)

Remove an object from the list of objects and destroy it.

• index [] A numerical index for the object.

set (sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• new_value [] A new integer value for the property, which is converted to a raw string during serialization.

class OwnedAttachment (*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• SBOLClass [] The type of child SBOL object contained by this Property

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

add (sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• new_value [] A new string which will be added to a list of values.

clear ()
clear()

Remove all children objects from the parent and destroy them.

create (uri)
create(uri) -> Test &

createCut (uri)
create(uri) -> Test &

createGenericLocation (uri)
create(uri) -> Test &

createRange (uri)
create(uri) -> Test &

define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object's identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

```python
find(uri)
find(uri) -> bool
```

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

```python
get(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getAll() getAll() -> std::vector<SBOLClass *>
```

Get all the objects contained in the property.

A vector of pointers to the objects

```python
getCut(*args)
getcut(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation(*args)
genericlocation(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
**SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

**uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
getRange(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

```cpp
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```cpp
class OwnedBuild(*args)
```

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

```cpp
python_iter : std::vector< std::string >::iterator
```

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```cpp
add(sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```cpp
clear()
```

Remove all children objects from the parent and destroy them.
**create** (uri)
create(uri) -> Test &

**createCut** (uri)
create(uri) -> Test &

**createGenericLocation** (uri)
create(uri) -> Test &

**createRange** (uri)
create(uri) -> Test &

**define** (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object. Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

**find** (uri)
find(uri) -> bool

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get** (*args)
geget(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll**()
gegetAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

**getCut** (*args)
geget(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getRange(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedCollection(*args)**

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property
- `python_iter`: `std::vector<std::string>::iterator`

C++ includes: `/Users/bbartley/Dev/git/libSBOL/source/object.h`

```cpp
add(sbol_obj)
    add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- `new_value` [] A new string which will be added to a list of values.

```cpp
clear()
```

Remove all children objects from the parent and destroy them.

```cpp
create(uri)
    create(uri) -> Test &
```

```cpp
createCut(uri)
    create(uri) -> Test &
```

```cpp
createGenericLocation(uri)
    create(uri) -> Test &
```

```cpp
createRange(uri)
    create(uri) -> Test &
```

```cpp
define(definition_object)
    define(definition_object) -> SBOLClass &
```

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- `SBOLClass` [] The type of SBOL object that will be created

- `definition_object` [] The returned object will reference the definition_object in its definition property.

```cpp
A reference to the child object check uniqueness of URI in Document
```

```cpp
find(uri)
    find(uri) -> bool
```

- `uri` [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

```cpp
get(*args)
    get(uri="") -> SBOLSubClass &
```

Get the child object.

- `SBOLClass` [] The type of the child object

- `SBOLSubClass` [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- `uri` [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
getAll() -> std::vector< SBOLClass * >
```
Get all the objects contained in the property.
A vector of pointers to the objects

```cpp
getCut(*args)
get(uri="") -> SBOLSubClass &
```
Get the child object.
• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &
```
Get the child object.
• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
getRange(*args)
get(uri="") -> SBOLSubClass &
```
Get the child object.
• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
remove(*args)
remove(index=0)
```
Remove an object from the list of objects and destroy it.
• index [] A numerical index for the object.
```python
set (sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```class OwnedCombinatorialDerivation (*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property
- **python_iter**: std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```add (sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```clear ()
clear()

Remove all children objects from the parent and destroy them.

```create (uri)
create(uri) -> Test &

```createCut (uri)
create(uri) -> Test &

```createGenericLocation (uri)
create(uri) -> Test &

```createRange (uri)
create(uri) -> Test &

```define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document
**find** (uri)

\[
\text{find}(\text{uri}) \rightarrow \text{bool}
\]

- **uri** [] The full uniform resource identifier of the object to search for in this property
  
  A boolean indicating whether found or not

**get** (*args)

\[
\text{get}(\text{uri}="") \rightarrow \text{SBOLSubClass} &
\]

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll**()

\[
\text{getAll}() \rightarrow \text{std::vector<SBOLClass} *
\]

Get all the objects contained in the property.

A vector of pointers to the objects

**getCut** (*args)

\[
\text{get}(\text{uri}="") \rightarrow \text{SBOLSubClass} &
\]

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getGenericLocation** (*args)

\[
\text{get}(\text{uri}="") \rightarrow \text{SBOLSubClass} &
\]

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getRange** (*args)

\[
\text{get}(\text{uri}="") \rightarrow \text{SBOLSubClass} &
\]

Get the child object.
• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```
Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

### class OwnedComponent(*args)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

• **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```python
add(sbol_obj)
add(new_value)
```
Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

```python
clear()
clear()
```
Remove all children objects from the parent and destroy them.

```python
create(uri)
create(uri) -> Test &
createCut(uri)
create(uri) -> Test &
createGenericLocation(uri)
create(uri) -> Test &
createRange(uri)
create(uri) -> Test &
```
**define** (definition_object)
```
define(definition_object) -> SBOLClass &
```
Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

**find** (uri)
```
find(uri) -> bool
```
- **uri** [] The full uniform resource identifier of the object to search for in this property
A boolean indicating whether found or not

**get** (*args)
```
get(uri=““) -> SBOLSubClass &
```
Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object

**getAll** ()
```
getAll() -> std::vector< SBOLClass * >
```
Get all the objects contained in the property.
A vector of pointers to the objects

**getCut** (*args)
```
get(uri=““) -> SBOLSubClass &
```
Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object

If no URI is specified, the first object in this OwnedObject property is returned.
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

getRange(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

remove(*args)
remove(index=0)

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

set(sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serial-

ization.

class OwnedComponentDefinition(*args)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are
indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting
the parent object will delete the child objects, and adding the parent object to a Document will also add the child
object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

add(sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.
• **new_value**  [] A new string which will be added to a list of values.

**clear**
```
clear()
```
Remove all children objects from the parent and destroy them.

**create** (uri)
```
create(uri) -> Test &
```

**createCut** (uri)
```
create(uri) -> Test &
```

**createGenericLocation** (uri)
```
create(uri) -> Test &
```

**createRange** (uri)
```
create(uri) -> Test &
```

**define** (definition_object)
```
define(definition_object) -> SBOLClass &
```
Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• **SBOLClass**  [] The type of SBOL object that will be created

• **definition_object**  [] The returned object will reference the definition_object in its definition property.

**find** (uri)
```
find(uri) -> bool
```
• **uri**  [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get** (*args)
```
get(uri="") -> SBOLSubClass &
```
Get the child object.

• **SBOLClass**  [] The type of the child object

• **SBOLSubClass**  [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri**  [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll**()
```
getAll() -> std::vector< SBOLClass * >
```
Get all the objects contained in the property.

A vector of pointers to the objects
getCut(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass []** The type of the child object
- **SBOLSubClass []** A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri []** The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

genericLocation(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass []** The type of the child object
- **SBOLSubClass []** A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri []** The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

genericRange(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass []** The type of the child object
- **SBOLSubClass []** A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri []** The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove(*args)
remove(index=0)

Remove an object from the list of objects and destroy it.

- **index []** A numerical index for the object.

set(sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value []** A new integer value for the property, which is converted to a raw string during serialization.
class OwnedDesign(*args)
    A container property that contains child objects.

    Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

    • SBOLClass[] The type of child SBOL object contained by this Property
    • python_iter: std::vector< std::string >::iterator

    C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

    add(sbol_obj)
        add(new_value)
        Appends the new value to a list of values, for properties that allow it.

        • new_value[] A new string which will be added to a list of values.

clear()
    clear()
    Remove all children objects from the parent and destroy them.

    create(uri)
        create(uri) -> Test &

    createCut(uri)
        create(uri) -> Test &

    createGenericLocation(uri)
        create(uri) -> Test &

    createRange(uri)
        create(uri) -> Test &

    define(definition_object)
        define(definition_object) -> SBOLClass &
        Autoconstructs a child object and attaches it to the parent object.
        Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

        • SBOLClass[] The type of SBOL object that will be created

        • definition_object[] The returned object will reference the definition_object in its definition property.

    A reference to the child object check uniqueness of URI in Document

    find(uri)
        find(uri) -> bool

        • uri[] The full uniform resource identifier of the object to search for in this property

        A boolean indicating whether found or not
get (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

g GetAll ()
g GetAll () -> std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

g getCut (*args)
g getCut(*args) -> SBOLSubClass &

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

g getGenericLocation (*args)
g getGenericLocation (*args) -> SBOLSubClass &

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

g getRange (*args)
g getRange (*args) -> SBOLSubClass &

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object. Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**remove** (*args*)

remove(index=0)

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

**set** (sbol_obj)

set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class** OwnedExperiment (*args*)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

• **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add** (sbol_obj)

add(new_value)

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

**clear** ()

clear()

Remove all children objects from the parent and destroy them.

**create** (uri)

create(uri) -> Test &

**createCut** (uri)

create(uri) -> Test &

**createGenericLocation** (uri)

create(uri) -> Test &

**createRange** (uri)

create(uri) -> Test &

**define** (definition_object)

define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values
specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

```python
find (uri)
find(uri) -> bool
```

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

```python
get (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getAll ()
getAll() -> std::vector< SBOLClass * >
```

Get all the objects contained in the property.

A vector of pointers to the objects

```python
getCut (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

### getRange (**args**)
get(uri="") -> SBOLSubClass &

Get the child object.

• **SBOLClass** [] The type of the child object
• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

### remove (**args**)
remove(index=0)

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

### set (sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

### class OwnedExperimentalData (**args**)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

• **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

### add (sbol_obj)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

### clear ()
clear()

Remove all children objects from the parent and destroy them.

### create (uri)
create(uri) -> Test &
**createCut** (uri)
create(uri) -> Test &

**createGenericLocation** (uri)
create(uri) -> Test &

**createRange** (uri)
create(uri) -> Test &

**define** (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

**find** (uri)
find(uri) -> bool

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get** (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll** ()
getAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

**getCut** (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```plaintext
getGenericLocation (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```plaintext
getRange (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```plaintext
remove (*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```plaintext
set (sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```plaintext
class OwnedFunctionalComponent (*args)
class OwnedFunctionalComponent (*args)
```

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

• **python_iter : std::vector< std::string >::iterator**
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add** *(sbol_obj)*

*add(new_value)*

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

**clear** *

clear()

Remove all children objects from the parent and destroy them.

**create** *(uri)*

create(uri) -> Test &

**createCut** *(uri)*

create(uri) -> Test &

**createGenericLocation** *(uri)*

create(uri) -> Test &

**createRange** *(uri)*

create(uri) -> Test &

**define** *(definition_object)*

define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

**find** *(uri)*

find(uri) -> bool

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get** *("args")*

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
getAll() -> std::vector< SBOLClass * >
Get all the objects contained in the property.
A vector of pointers to the objects

gCut(*args)
get(uri="") -> SBOLSubClass &
Get the child object.

- SBOLClass [] The type of the child object
- SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object
Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

gGenericLocation(*args)
get(uri="") -> SBOLSubClass &
Get the child object.

- SBOLClass [] The type of the child object
- SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object
Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

gRange(*args)
get(uri="") -> SBOLSubClass &
Get the child object.

- SBOLClass [] The type of the child object
- SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object
Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove(*args)
remove(index=0)
Remove an object from the list of objects and destroy it.

- index [] A numerical index for the object.

set(sbol_obj)
set(new_value)
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
• new_value [] A new integer value for the property, which is converted to a raw string during serialization.

class OwnedImplementation(*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• SBOLClass [] The type of child SBOL object contained by this Property

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

add(sbol_obj)
add(new_value)
Appends the new value to a list of values, for properties that allow it.

• new_value [] A new string which will be added to a list of values.

clear()
clear()
Remove all children objects from the parent and destroy them.

create(uri)
create(uri) -> Test &

createCut (uri)
createCut (uri) -> Test &

createGenericLocation (uri)
createGenericLocation (uri) -> Test &

createRange (uri)
createRange (uri) -> Test &

define(definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• SBOLClass [] The type of SBOL object that will be created

• definition_object [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

find(uri)
find(uri) -> bool

• uri [] The full uniform resource identifier of the object to search for in this property
A boolean indicating whether found or not

```python
get (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getAll ()
getAll() -> std::vector< SBOLClass * >
```

Get all the objects contained in the property.

A vector of pointers to the objects

```python
getCut (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getRange (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove (*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

```python
set (sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

class OwnedInteraction (*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```python
add (sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```python
clear ()
clear()
```

Remove all children objects from the parent and destroy them.

```python
create (uri)
create(uri) -> Test &
```

```python
createCut (uri)
create(uri) -> Test &
```

```python
createGenericLocation (uri)
create(uri) -> Test &
```

```python
createRange (uri)
create(uri) -> Test &
```

```python
define (definition_object)
define(definition_object) -> SBOLClass &
```

Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

```cpp
def(find, uri) -> bool
    find(uri) -> bool
    • uri [] The full uniform resource identifier of the object to search for in this property
    A boolean indicating whether found or not
```

```cpp
def(get, *args) -> SBOLSubClass &
    get(uri="") -> SBOLSubClass &
    Get the child object.

    • SBOLClass [] The type of the child object
    • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

    • uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
    A reference to the child object
    Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
```

```cpp
def(getAll, () -> std::vector< SBOLClass * >
    getAll() -> std::vector< SBOLClass * >
    Get all the objects contained in the property.
    A vector of pointers to the objects
```

```cpp
def(getCut, *args) -> SBOLSubClass &
    get(uri="") -> SBOLSubClass &
    Get the child object.

    • SBOLClass [] The type of the child object
    • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

    • uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
    A reference to the child object
    Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
```

```cpp
def(getGenericLocation, *args) -> SBOLSubClass &
    get(uri="") -> SBOLSubClass &
    Get the child object.

    • SBOLClass [] The type of the child object
```
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getRange (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove (*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

```python
set (sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

---

**class OwnedLocation (*args)**

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

```python
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h
```

```python
add (sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```python
clear ()
clear()
```

Remove all children objects from the parent and destroy them.
create (uri)
create(uri) -> Test &

createCut (uri)
create(uri) -> Test &

createGenericLocation (uri)
create(uri) -> Test &

createRange (uri)
create(uri) -> Test &

define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• SBOLClass [] The type of SBOL object that will be created

• definition_object [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

find (uri)
find(uri) -> bool

• uri [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

get (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

• SBOLClass [] The type of the child object

• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getAll ()
getAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.
A vector of pointers to the objects

getcut (*args)
getcut(uri="") -> SBOLSubClass &

Get the child object.

• SBOLClass [] The type of the child object
• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object. Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
get Generic Location(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object. Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
get Range(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object. Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class** **OwnedMapsTo**(*args)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property
• **python_iter**: `std::vector<std::string>::iterator`

C++ includes: `/Users/bbartley/Dev/git/libSBOL/source/object.h`  

```cpp
add(sbol_obj)  
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.  

• **new_value** [] A new string which will be added to a list of values.

```cpp
clear()
```

Remove all children objects from the parent and destroy them.  

```cpp
create(uri)  
create(uri) -> Test &
```

```cpp
createCut(uri)  
createCut(uri) -> Test &
```

```cpp
createGenericLocation(uri)  
createGenericLocation(uri) -> Test &
```

```cpp
createRange(uri)  
createRange(uri) -> Test &
```

```cpp
define(definition_object)  
define(definition_object) -> SBOLClass &
```

Autoconstructs a child object and attaches it to the parent object.  

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.  

• **SBOLClass** [] The type of SBOL object that will be created  

• **definition_object** [] The returned object will reference the definition_object in its definition property.  

```cpp
find(uri)  
find(uri) -> bool
```

A reference to the child object check uniqueness of URI in Document  

```cpp
get(*args)  
get(uri="") -> SBOLSubClass &
```

Get the child object.  

• **SBOLClass** [] The type of the child object  

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.  

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll** ()
```cpp
getAll() -> std::vector<SBOLClass *> 
```
Get all the objects contained in the property.
A vector of pointers to the objects

**getCut** (*args)
```cpp
get(uri="") -> SBOLSubClass & 
```
Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getGenericLocation** (*args)
```cpp
get(uri="") -> SBOLSubClass & 
```
Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getRange** (*args)
```cpp
get(uri="") -> SBOLSubClass & 
```
Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**remove** (*args)
```cpp
remove(index=0) 
```
Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.
set (sbol_obj)
    set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedMeasurement (**args**)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter**: std::vector< std::string >::iterator

**C++ includes**: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add (sbol_obj)**
    add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

**clear ()**
    clear()

Remove all children objects from the parent and destroy them.

**create (uri)**
    create(uri) -> Test &

**createCut (uri)**
    create(uri) -> Test &

**createGenericLocation (uri)**
    create(uri) -> Test &

**createRange (uri)**
    create(uri) -> Test &

**define (definition_object)**
    define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document
**find**(*uri*)
> find(*uri*) -> bool
> *
> **uri**  [] The full uniform resource identifier of the object to search for in this property
> A boolean indicating whether found or not

**get** (*args*)
> get(*uri=*""*) -> SBOLSubClass &
> Get the child object.
> *
> **SBOLClass**  [] The type of the child object
> **SBOLSubClass**  [] A derived class of SBOLClass. Use this type specialization when adding multiple
> types of SBOLObjects to a container.
> *
> **uri**  [] The specific URI for a child object if this OwnedObject property contains multiple objects,
> A reference to the child object Returns a child object from the OwnedObject property. If no URI is
> specified, the first object in this OwnedObject property is returned.

**getAll**()
> getAll() -> std::vector< SBOLClass * >
> Get all the objects contained in the property.
> A vector of pointers to the objects

**getCut** (*args*)
> get(*uri=*""*) -> SBOLSubClass &
> Get the child object.
> *
> **SBOLClass**  [] The type of the child object
> **SBOLSubClass**  [] A derived class of SBOLClass. Use this type specialization when adding multiple
> types of SBOLObjects to a container.
> *
> **uri**  [] The specific URI for a child object if this OwnedObject property contains multiple objects,
> A reference to the child object Returns a child object from the OwnedObject property. If no URI is
> specified, the first object in this OwnedObject property is returned.

**getGenericLocation** (*args*)
> get(*uri=*""*) -> SBOLSubClass &
> Get the child object.
> *
> **SBOLClass**  [] The type of the child object
> **SBOLSubClass**  [] A derived class of SBOLClass. Use this type specialization when adding multiple
> types of SBOLObjects to a container.
> *
> **uri**  [] The specific URI for a child object if this OwnedObject property contains multiple objects,
> A reference to the child object Returns a child object from the OwnedObject property. If no URI is
> specified, the first object in this OwnedObject property is returned.

**getRange** (*args*)
> get(*uri=*""*) -> SBOLSubClass &
> Get the child object.
• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```python
class OwnedModel(*args)
class OwnedModel(*args)
```

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

• **python_iter** : std::vector< std::string >::iterator

```python
add(sbol_obj)
add(sbol_obj)
```

Add the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

```python
clear()
clear()
```

Remove all children objects from the parent and destroy them.

```python
create(uri)
create(uri) -> Test &
createCut(uri)
createCut(uri) -> Test &
createGenericLocation(uri)
createGenericLocation(uri) -> Test &
createRange(uri)
createRange(uri) -> Test &
```
**define** *(definition_object)*

define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

**find** *(uri)*

find(uri) -> bool

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get** *("args")

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll** *

getAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

**getCut** *("args")

gcut(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
**getGenericLocation** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getSize** (*args*)

get(uri="") -> Integer &

Get the size of the child object.

- **size** [] The size of the child object

A reference to the child object Returns the size from the OwnedObject property. If no URI is specified, the size of the first object in this OwnedObject property is returned.

**remove** (*args*)

remove(index=0)

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

**set** (sbol_obj)

set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

---

**class OwnedModule** (*args*)

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter : std::vector std::string : : iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add** (sbol_obj)

add(new_value)

Appends the new value to a list of values, for properties that allow it.
• **new_value**  [] A new string which will be added to a list of values.

clear()
clear() Remove all children objects from the parent and destroy them.

create (uri)
create(uri) -> Test &

createCut (uri)
create(uri) -> Test &

createGenericLocation (uri)
create(uri) -> Test &

createRange (uri)
create(uri) -> Test &

define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• **SBOLClass**  [] The type of SBOL object that will be created

• **definition_object**  [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

find (uri)
find(uri) -> bool

• **uri**  [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

get (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

• **SBOLClass**  [] The type of the child object

• **SBOLSubClass**  [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri**  [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getAll()
getAll() -> std::vector<SBOLClass *>

Get all the objects contained in the property.

A vector of pointers to the objects

149
getCut (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getGenericLocation (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getRange (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove (*args)
remove(index=0)

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

set (sbol_obj)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.
class OwnedModuleDefinition(*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property
- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add**(sbol_obj)

    add(new_value)

    Appends the new value to a list of values, for properties that allow it.

    - **new_value** [] A new string which will be added to a list of values.

**clear**()

clear()

Remove all children objects from the parent and destroy them.

**create**(uri)

create(uri) -> Test &

**createCut**(uri)

createCut(uri) -> Test &

**createGenericLocation**(uri)

createGenericLocation(uri) -> Test &

**createRange**(uri)

createRange(uri) -> Test &

**define**(definition_object)

define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

**find**(uri)

    find(uri) -> bool

    - **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not
get (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

getAll ()
getAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

getCode (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

getGenericLocation (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

getRange (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```
Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```python
class OwnedParticipation(*args)
class OwnedParticipation
```
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

```python
python_iter : std::vector< std::string >::iterator
```
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```python
add(sbol_obj)
add(new_value)
```
Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```python
clear()
clear()
```
Remove all children objects from the parent and destroy them.

```python
create(uri)
create(uri) -> Test &
createCut(uri)
createCut(uri) -> Test &
createGenericLocation(uri)
createGenericLocation(uri) -> Test &
createRange(uri)
createRange(uri) -> Test &
define(definition_object)
define(definition_object) -> SBOLClass &
define(definition_object)
define(definition_object) -> SBOLClass &
define(definition_object)
define(definition_object) -> SBOLClass &
```
Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values...
specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

```
find (uri)
find(uri) -> bool

• uri [] The full uniform resource identifier of the object to search for in this property
A boolean indicating whether found or not
```

```
get (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```
getAll ()
getAll() -> std::vector< SBOLClass * >
```

Get all the objects contained in the property.

A vector of pointers to the objects

```
getCut (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```
getGenericLocation (*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects. A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getRange (*args)
get(uri=")" -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects. A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove (*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```python
class OwnedPlan (*args)
```

A container property that contains child objects. Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

```python
add(sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```python
clear()
clear()
```

Remove all children objects from the parent and destroy them.

```python
create (uri)
create(uri) -> Test &
```
createCut (uri)
create(uri) -> Test &

createGenericLocation (uri)
create(uri) -> Test &

createRange (uri)
create(uri) -> Test &

define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object. Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- SBOLClass [] The type of SBOL object that will be created
- definition_object [] The returned object will reference the definition_object in its definition property.

find (uri)
find(uri) -> bool

- uri [] The full uniform resource identifier of the object to search for in this property
A boolean indicating whether found or not

get (*args)
get(uri="") -> SBOLSubClass &
Get the child object.

- SBOLClass [] The type of the child object
- SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getAll ()
getAll() -> std::vector<SBOLClass *>
Get all the objects contained in the property.
A vector of pointers to the objects

getCode (*args)
getCode(uri="") -> SBOLSubClass &
Get the child object.

- SBOLClass [] The type of the child object
- SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getRange(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

- **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedSampleRoster(*args)**

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property

- **python_iter**: `std::vector<std::string>::iterator`
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add** *(sbol_obj)*
```
add(new_value)
```
Add to the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

**clear** *
```
clear()
```
Remove all children objects from the parent and destroy them.

**create** *(uri)*
```
create(uri) -> Test &
```

**createCut** *(uri)*
```
create(uri) -> Test &
```

**createGenericLocation** *(uri)*
```
create(uri) -> Test &
```

**createRange** *(uri)*
```
create(uri) -> Test &
```

**define** *(definition_object)*
```
define(definition_object) -> SBOLClass &
```
Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

**find** *(uri)*
```
find(uri) -> bool
```
A boolean indicating whether found or not.

**get** *("args")*
```
get(uri="") -> SBOLSubClass &
```
Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
getAll() -> std::vector< SBOLClass * >
Get all the objects contained in the property.
A vector of pointers to the objects

gCut(*args)
get(uri="") -> SBOLSubClass &
Get the child object.

• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

genericLocation(*args)
get(uri="") -> SBOLSubClass &
Get the child object.

• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

gRange(*args)
get(uri="") -> SBOLSubClass &
Get the child object.

• SBOLClass [] The type of the child object
• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is
specified, the first object in this OwnedObject property is returned.

remove(*args)
remove(index=0)
Remove an object from the list of objects and destroy it.

• index [] A numerical index for the object.

set(sbol_obj)
set(new_value)
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

class OwnedSequence(*args)
    A container property that contains child objects.

    Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

    • **SBOLClass** [] The type of child SBOL object contained by this Property

    • **python_iter**: std::vector<std::string>::iterator

    C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

    add(sbol_obj)
        add(new_value)

        Appends the new value to a list of values, for properties that allow it.

            • **new_value** [] A new string which will be added to a list of values.

    clear()
        clear()

        Remove all children objects from the parent and destroy them.

    create(uri)
        create(uri) -> Test &

    createCut(uri)
        create(uri) -> Test &

    createGenericLocation(uri)
        create(uri) -> Test &

    createRange(uri)
        create(uri) -> Test &

    define(definition_object)
        define(definition_object) -> SBOLClass &

        Autoconstructs a child object and attaches it to the parent object.

        Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

            • **SBOLClass** [] The type of SBOL object that will be created

            • **definition_object** [] The returned object will reference the definition_object in its definition property.

        A reference to the child object check uniqueness of URI in Document

    find(uri)
        find(uri) -> bool

        • **uri** [] The full uniform resource identifier of the object to search for in this property
A boolean indicating whether found or not

**get** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll** ()

getAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.
A vector of pointers to the objects

**getCut** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getGenericLocation** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getRange** (*args*)

get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.
• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

* **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

* **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedSequenceAnnotation(*args)**

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

* **SBOLClass** [] The type of child SBOL object contained by this Property

* **python_iter** :: std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```python
add(sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

* **new_value** [] A new string which will be added to a list of values.

```python
clear()
clear()
```

Remove all children objects from the parent and destroy them.

```python
create(uri)
create(uri) -> Test &
```

```python
createCut(uri)
createCut(uri) -> Test &
```

```python
createGenericLocation(uri)
createGenericLocation(uri) -> Test &
```

```python
createRange(uri)
createRange(uri) -> Test &
```

```python
define(definition_object)
define(definition_object) -> SBOLClass &
```

Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created

- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

```python
find(uri)
find(uri) -> bool

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not
```

```python
get(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
```

```python
getAll()
getAll() -> std::vector<SBOLClass *>

Get all the objects contained in the property.

A vector of pointers to the objects
```

```python
getCut(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object

- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
```

```python
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
getRange(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```cpp
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

### class OwnedSequenceConstraint(*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

• **python_iter : std::vector<std::string>::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```cpp
add(sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

```cpp
clear()
clear()
```

Remove all children objects from the parent and destroy them.
create (uri)
create(uri) -> Test &

createCut (uri)
create(uri) -> Test &

createGenericLocation (uri)
create(uri) -> Test &

createRange (uri)
create(uri) -> Test &

define (definition_object)
define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.
Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• SBOLClass [] The type of SBOL object that will be created

• definition_object [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

find (uri)
find(uri) -> bool

• uri [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

get (*args)
get(uri="") -> SBOLSubClass &

Get the child object.

• SBOLClass [] The type of the child object

• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getAll ()
getAll() -> std::vector< SBOLClass * >

Get all the objects contained in the property.

A vector of pointers to the objects

getcut (*args)
getcut(uri="") -> SBOLSubClass &

Get the child object.

• SBOLClass [] The type of the child object
• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getGenericLocation(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
getRange(*args)
get(uri="") -> SBOLSubClass &
```

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```python
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```python
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class OwnedTest(*args)**

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property
• `python_iter`: `std::vector< std::string >::iterator`

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add**(`sbol_obj`)

`add(new_value)`

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

**clear**()

`clear()`

Remove all children objects from the parent and destroy them.

**create**(`uri`)

`create(uri) -> Test &`

**createCut**(`uri`)

`createCut(uri) -> Test &`

**createGenericLocation**(`uri`)

`createGenericLocation(uri) -> Test &`

**createRange**(`uri`)

`createRange(uri) -> Test &`

**define**(`definition_object`)

`define(definition_object) -> SBOLClass &`

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

• **SBOLClass** [] The type of SBOL object that will be created

• **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

**find**(`uri`)

`find(uri) -> bool`

• **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get**(*args*)

`get(uri="") -> SBOLSClass &`

Get the child object.

• **SBOLClass** [] The type of the child object

• **SBOLSClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,
A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

getAll ()
getAll() -> std::vector<SBOLClass *>
Get all the objects contained in the property.
A vector of pointers to the objects

getcut (*args)
getch(uri="") -> SBOLSubClass &
Get the child object.

• SBOLClass [] The type of the child object

• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

genericLocation (*args)
getch(uri="") -> SBOLSubClass &
Get the child object.

• SBOLClass [] The type of the child object

• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

genericRange (*args)
getch(uri="") -> SBOLSubClass &
Get the child object.

• SBOLClass [] The type of the child object

• SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

remove (*args)
remove(index=0)
Remove an object from the list of objects and destroy it.

• index [] A numerical index for the object.
set(sbol_obj)
  set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

**class** OwnedUsage (*args)
A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

- **SBOLClass** [] The type of child SBOL object contained by this Property
- **python_iter**: std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

**add**(sbol_obj)
  add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

**clear**()
  clear()

Remove all children objects from the parent and destroy them.

**create**(uri)
  create(uri) -> Test &

**createCut**(uri)
  create(uri) -> Test &

**createGenericLocation**(uri)
  create(uri) -> Test &

**createRange**(uri)
  create(uri) -> Test &

**define**(definition_object)
  define(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document
find(uri)
find(uri) -> bool
• uri [] The full uniform resource identifier of the object to search for in this property
  A boolean indicating whether found or not

get(*args)
get(uri="") -> SBOLSubClass &
Get the child object.
  • SBOLClass [] The type of the child object
  • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.
  • uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
  A reference to the child object Returns a child object from the OwnedObject property. If no URI is
  specified, the first object in this OwnedObject property is returned.

getAll()
getAll() -> std::vector< SBOLClass * >
Get all the objects contained in the property.
A vector of pointers to the objects

getCut(*args)
get(uri="") -> SBOLSubClass &
Get the child object.
  • SBOLClass [] The type of the child object
  • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.
  • uri [] The specific URI for a child object if thisOwnedObject property contains multiple objects,
  A reference to the child object Returns a child object from the OwnedObject property. If no URI is
  specified, the first object in this OwnedObject property is returned.

getGenericLocation(*args)
get(uri="") -> SBOLSubClass &
Get the child object.
  • SBOLClass [] The type of the child object
  • SBOLSubClass [] A derived class of SBOLClass. Use this type specialization when adding multiple
types of SBOLObjects to a container.
  • uri [] The specific URI for a child object if this OwnedObject property contains multiple objects,
  A reference to the child object Returns a child object from the OwnedObject property. If no URI is
  specified, the first object in this OwnedObject property is returned.

getRange(*args)
get(uri="") -> SBOLSubClass &
Get the child object.
• **SBOLClass** [] The type of the child object

• **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

• **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

```cpp
remove(*args)
remove(index=0)
```

Remove an object from the list of objects and destroy it.

• **index** [] A numerical index for the object.

```cpp
set(sbol_obj)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```cpp
class OwnedVariableComponent(*args)
```

A container property that contains child objects.

Creates a composition out of two or more classes. In the SBOL specification, compositional relationships are indicated in class diagrams by arrows with black diamonds. A compositional relationship means that deleting the parent object will delete the child objects, and adding the parent object to a Document will also add the child object. Owned objects are stored in arbitrary order.

• **SBOLClass** [] The type of child SBOL object contained by this Property

```cpp
python_iter : std::vector< std::string >::iterator
```

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```cpp
add(sbol_obj)
add(new_value)
```

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

```cpp
clear()
clear()
```

Remove all children objects from the parent and destroy them.

```cpp
create(uri)
create(uri) -> Test &
```

```cpp
createCut(uri)
create(uri) -> Test &
```

```cpp
createGenericLocation(uri)
create(uri) -> Test &
```

```cpp
createRange(uri)
create(uri) -> Test &
```
**define**(definition_object)

*define*(definition_object) -> SBOLClass &

Autoconstructs a child object and attaches it to the parent object.

Additionally, it sets the definition property of the child object, for example, in the case of creating Components, FunctionalComponents, and Modules. The new object will be constructed with default values specified in the constructor for this type of object. If SBOLCompliance is enabled, the child object’s identity will be constructed using the supplied displayId argument. Otherwise, the user should supply a full URI.

- **SBOLClass** [] The type of SBOL object that will be created
- **definition_object** [] The returned object will reference the definition_object in its definition property.

A reference to the child object check uniqueness of URI in Document

**find**(uri)

*find*(uri) -> bool

- **uri** [] The full uniform resource identifier of the object to search for in this property

A boolean indicating whether found or not

**get**("*args")

*get*(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getAll**()

*getAll*() -> *std::vector< SBOLClass * >*

Get all the objects contained in the property.

A vector of pointers to the objects

**getCut**("*args")

*getCut*(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass** [] The type of the child object
- **SBOLSubClass** [] A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri** [] The specific URI for a child object if this OwnedObject property contains multiple objects,

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.
**getGenericLocation** (*args*)

    get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass []** The type of the child object
- **SBOLSubClass []** A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri []** The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**getRange** (*args*)

    get(uri="") -> SBOLSubClass &

Get the child object.

- **SBOLClass []** The type of the child object
- **SBOLSubClass []** A derived class of SBOLClass. Use this type specialization when adding multiple types of SBOLObjects to a container.

- **uri []** The specific URI for a child object if this OwnedObject property contains multiple objects.

A reference to the child object Returns a child object from the OwnedObject property. If no URI is specified, the first object in this OwnedObject property is returned.

**remove** (*args*)

    remove(index=0)

Remove an object from the list of objects and destroy it.

- **index []** A numerical index for the object.

**set** (sbol_obj)

    set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value []** A new integer value for the property, which is converted to a raw string during serialization.

**class PartShop** (*args*)

A class which provides an API front-end for online bioparts repositories.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/partshop.h

**addSynBioHubAnnotations** (doc)

    addSynBioHubAnnotations(doc)

**attachFile** (topleveluri, filename)

    attachFile(topleveluri, filename)

Upload and attach a file to a TopLevel object in a PartShop.

- **top_level_uri []** The identity of the object to which the file will be attached
- **file_name []** A path to the file attachment
countCollection()

    count() -> int

    Return the count of objects contained in a PartShop.

    • SBOLClass [] The type of SBOL object, usually a ComponentDefinition

countComponentDefinition()

    count() -> int

    Return the count of objects contained in a PartShop.

    • SBOLClass [] The type of SBOL object, usually a ComponentDefinition

downloadAttachment(*args)

downloadAttachment(attachment_uri, path=".")

    Download a file attached to a TopLevel object in an online repository.

    • attachment_uri [] The full URI of the attached object

    • path [] The target path to which the file will be downloaded

getURL()

getURL() -> std::string

    Returns the network address of the PartShop.

    The URL of the online repository

login(*args)

    login(email, password="")

    In order to submit to a PartShop, you must login first.

    Register on SynBioHub to obtain account credentials.

    • email [] The email associated with the user’s SynBioHub account

    • password [] The user’s password

pull(*args)

    pull(uri, doc, recursive)

pullCollection(uri, doc, recursive=True)

pullCollection(uri, doc, recursive)

pullComponentDefinition(uri, doc, recursive=True)

pullComponentDefinition(uri, doc, recursive)

pullSequence(uri, doc, recursive=True)

pullSequence(uri, doc, recursive)

search(*args)

    search(q) -> SearchResponse &

    Perform an ADVANCED search using a SearchQuery object.

    • search_query [] A map of string key-value pairs. Keys are objectType, sbolTag, collection, dc-
terms:tag, namespace/tag, offset, limit.

    Search metadata A vector of maps with key-value pairs.

searchCount(*args)

    searchCount(q) -> int

    Returns the number of search records matching the given criteria for an ADVANCED search.
• **search_query** [] A map of string key-value pairs. See SearchQuery for required and optional criteria.

An integer count.

**searchRootCollections()**

```cpp
searchRootCollections() -> std::string
```

Returns all Collections that are not members of any other Collections.

• **doc** [] A Document to add the Collections to

**searchSubCollections(uri)**

```cpp
searchSubCollections(uri) -> std::string
```

Returns all Collections that are members of the Collection specified by its URI.

• **uri** [] The URI of a Collection of Collections

• **doc** [] A Document to add the subcollections to

**submit(*args)**

```cpp
submit(doc, collection="", overwrite=0) -> std::string
```

Submit an SBOL Document to SynBioHub.

• **doc** [] The Document to submit

• **collection** [] The URI of an SBOL Collection to which the Document contents will be uploaded

• **overwrite** [] An integer code: 0 (default) - do not overwrite, 1 - overwrite, 2 - merge

**class Participation(*args)**

Each Participation represents how a particular FunctionalComponent behaves in its parent Interaction.

• **roles** [URIProperty ] The roles property is an OPTIONAL set of URIs that describes the behavior of a Participation (and by extension its referenced FunctionalComponent) in the context of its parent Interaction.

The roles property MAY contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the participant role branch of the SBO. The table below provides a list of possible SBO terms for the roles property and their corresponding URIs. Role

<table>
<thead>
<tr>
<th>Systems Biology Ontology Term</th>
<th>LibSBOL Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inhibitor</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000020">http://identifiers.org/biomodels.sbo/SBO:0000020</a></td>
</tr>
<tr>
<td>Stimulator</td>
<td><a href="http://identifiers.org/biomodels.sbo/SBO:0000459">http://identifiers.org/biomodels.sbo/SBO:0000459</a></td>
</tr>
</tbody>
</table>
SBO_PRODUCT
Ligand
http://identifiers.org/biomodels.sbo/SBO:0000280

SBO_LIGAND
Non-covalent Complex
http://identifiers.org/biomodels.sbo/SBO:0000253

SBO_NONCOVALENT_COMPLEX
If a Participation is well described by one of the terms from this table then its roles property MUST contain the URI that identifies this term. Lastly, if the roles property of a Participation contains multiple URIs, then they MUST identify non-conflicting terms. For example, the SBO terms stimulator and inhibitor would conflict.

- **participant** [ReferencedObject] The participant property MUST specify precisely one FunctionalComponent object that plays the designated role in its parent Interaction object.

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.
• **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/participation.h

```cpp
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/participation.h

define (*args)
define(species, role="")

class ParticipationProperty (*args)
class ParticipationProperty (*args)

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

• **The** [] SBOL specification currently supports string, URI, and integer literal values.

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add (new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• new_value [] A new string which will be added to a list of values.

addValidationRule (*args)
addValidationRule(rule)

```cpp
clear ()
clear()

Clear all property values.

copy (target_property)
copy(target_property)

Copy property values to a target object’s property fields.

find (query)
find(query) -> bool

Check if a value in this property matches the query.

getLowerBound ()
getLowerBound() -> char

getOwner ()
getOwner() -> SBOLObject &

getTypeURI ()
getTypeURI() -> rdf_type

getUpperBound ()
getUpperBound() -> char

remove (index=0)
remove(index=0)
Remove a property value.

```python
set (*args)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- `new_value` [] A new integer value for the property, which is converted to a raw string during serialization.

```python
validate (arg=None)
validate(arg=NULL)
```

```python
write ()
write()
```

```python
class Plan (*args)
```

Examples of agents are person, organisation or software. These agents should be annotated with additional information, such as software version, needed to be able to run the same software again.

- `attachments : ReferencedObject`

- `persistentIdentity [URIProperty ]` The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- `displayId [TextProperty ]` The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULd be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- `version [VersionProperty ]` If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- `wasDerivedFrom [URIProperty ]` The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- `wasGeneratedBy [ReferencedObject ]` An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- `name [TextProperty ]` The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- `description [TextProperty ]` The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.
• **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/provo.h

```cpp
    copy (*args)
    copy(ns="", version="") -> SBOLClass &
```

**class PlanProperty (*args)**

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

• **The  []** SBOL specification currently supports string, URI, and integer literal values.

• **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
    add (new_value)
    add(new_value)
        Appends the new value to a list of values, for properties that allow it.

    * new_value [] A new string which will be added to a list of values.
```

```cpp
    addValidationRule (*args)
    addValidationRule(rule)
```

```cpp
    clear ()
    clear()
        Clear all property values.
```

```cpp
    copy (target_property)
    copy(target_property)
        Copy property values to a target object’s property fields.
```

```cpp
    find (query)
    find(query) -> bool
        Check if a value in this property matches the query.
```

```cpp
    getLowerBound ()
    getLowerBound() -> char
```

```cpp
    getOwner ()
    getOwner() -> SBOLObject &
```

```cpp
    getTypeURI ()
    getTypeURI() -> rdf_type
```

```cpp
    getUpperBound ()
    getUpperBound() -> char
```

```cpp
    remove (index=0)
    remove(index=0)
```

```cpp
    remove(index)
        Remove the value at the given index.
```

```cpp
    remove(index=0)
        Remove the first value from the list.
```

```cpp
    remove(index)
        Remove the value at the given index.
```

```cpp
    remove(index)
        Remove the value at the given index.
```

```cpp
    remove(index=0)
        Remove the first value from the list.
```
Remove a property value.

```python
set(*args)
set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```python
validate(arg=None)
validate(arg=NULL)
```

```python
write()
write()
```

```python
class Range(*args)
```

A Range object specifies a region via discrete, inclusive start and end positions that correspond to indices for characters in the elements String of a Sequence. Note that the index of the first location is 1, as is typical practice in biology, rather than 0, as is typical practice in computer science.

- **start** [IntProperty] Specifies the inclusive starting position of a sequence region. It must be 1 or greater.
- **end** [IntProperty] Specifies the inclusive end position of a sequence region. It must be equal to or greater than the start.
- **orientation** [URIProperty] The orientation indicates how a region of double-stranded DNA represented by the parent SequenceAnnotation and its associated Component are oriented.

The orientation may be one of the following values. By default it is set to SBOL_ORIENTATION_INLINE. Orientation URI

```
libSBOL Symbol
http://sbols.org/v2#inline
SBOL_ORIENTATION_INLINE
http://sbols.org/v2#reverseComplement
SBOL_ORIENTATION_REVERSE_COMPLEMENT
```

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version
of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/location.h

```cpp
adjoins (comparand)
adjoins(comparand) -> int
Indicate if these Ranges represent adjacent intervals.
Another Range object
1 if these Ranges adjoin or border each other, 0 if they are separated by an intervening Range
```

```cpp
contains (comparand)
contains(comparand) -> int
Calculates how many bases of the comparand are contained by this Range.
Another Range object
The number of bases which are contained (equivalent to the length of the comparand), or 0 if it is not contained.
```

```cpp
follows (comparand)
follows(comparand) -> int
Calculates how many bases separate these Ranges Another Range object.
The number of bases by which this Range follows the comparand, or 0 if it does not follow
```

```cpp
length ()
length() -> int
Returns the length of a Range.
```

```cpp
overlaps (comparand)
overlaps(comparand) -> int
Calculates how many bases separate this Range from the comparand.
Another Range object
```
The number of bases by which the Ranges overlap. If they overlap, this is always a positive number regardless of direction. If they do not overlap, returns zero.

```python
precedes (comparand)
precedes(comparand) -> int
```

Calculates how many bases separate these Ranges Another Range object.

The number of bases by which this Range precedes the comparand, or 0 if it does not precede.

```python
class ReferencedObject (*args)
```

A reference to another SBOL object. Contains a Uniform Resource Identifier (URI) that refers to an an associated object.

The object it points to may be another resource in this Document or an external reference, for example to an object in an external repository or database. In the SBOL specification, association by reference is indicated in class diagrams by arrows with open (white) diamonds.

- `python_iter`: `std::vector< std::string >::iterator`

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

```python
add (*args)
   add(obj)
```

Append a URI reference of an object to the property store.

- `The` [] referenced object

```python
addReference (uri)
   addReference(uri)
```

```python
create (uri)
   create(uri) -> std::string
```

Creates a new SBOL object corresponding to the RDF type specified in the Property definition.

Creates another SBOL object derived from TopLevel and adds it to the Document.

- `uri` [] A Uniform Resource Identifier (URI) for the new object, or a displayId if operating in SBOL-compliant mode (library default)

The full URI of the created object.

- `uri` [] In “open world” mode, this is a full URI and the same as the returned URI. If the default namespace for libSBOL has been configured, then this argument should simply be a local identifier. If SBOL-compliance is enabled, this argument should be the intended displayId of the new object.

A full URI is automatically generated and returned.

The full URI of the created object.

```python
set (*args)
   set(new_value)
```

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- `new_value` [] A new integer value for the property, which is converted to a raw string during serialization.

```python
setReference (uri)
   setReference(uri)
```

```python
class SBOLObject (*args)
```

An SBOLObject converts a C++ class data structure into an RDF triple store and contains methods for serializing and parsing RDF triples.
• identity [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/object.h

addPropertyValue(property_uri, val)
addPropertyValue(property_uri, val)

Append a value to a user-defined annotation property.

Either a literal or URI value

apply(callback_fn, user_data)
apply(callback_fn, user_data)

cast(python_class)
cast() -> SBOLClass &

compare(comparand)
compare(comparand) -> int

Compare two SBOL objects or Documents.

The behavior is currently undefined for objects with custom annotations or extension classes.

• comparand [] A pointer to the object being compared to this one.

1 if the objects are identical, 0 if they are different

find(uri)
find(uri) -> SBOLObject *

Search this object recursively to see if an object or any child object with URI already exists.

• uri [] The URI to search for.

A pointer to the object with this URI if it exists, NULL otherwise

find_property(uri)
find_property(uri) -> SBOLObject *

Search this object recursively to see if it contains a member property with the given RDF type.

• uri [] The RDF type of the property to search for.

A pointer to the object that contains a member property with the specified RDF type, NULL otherwise

find_property_value(*args)
find_property_value(uri, value, matches={}) -> std::vector< SBOLObject * >

Search this object recursively to see if it contains a member property with the given RDF type and indicated property value.

• uri [] The RDF type of the property to search for.

• value [] The property value to match

A vector containing all objects found that contain a member property with the specified RDF type

find_reference(uri)
find_reference(uri) -> std::vector< SBOLObject * >
Search this object recursively to see if it contains a member property with the given RDF type and indicated property value.

- **uri** [] A URI, either an ontology term or an object reference, to search for

A vector containing all objects found that contain the URI in a property value

**getAnnotation** (property_uri)

```cpp
getAnnotation(property_uri) -> std::string
```

Get the value of a custom annotation property by its URI.

Synonymous with getPropertyValue

- **property_uri** [] The URI for the property

The value of the property or SBOL_ERROR_NOT_FOUND

**getClassName** (type)

```cpp
getClassName(type) -> std::string
```

Parses a local class name from the RDF-type of this SBOL Object

**getProperties** ()

```cpp
getProperties() -> std::vector< std::string >
```

Gets URIs for all properties contained by this object.

This includes SBOL core properties as well as custom annotations. Use this to find custom extension data in an SBOL file.

A vector of URIs that identify the properties contained in this object

**getPropertyValue** (property_uri)

```cpp
getPropertyValue(property_uri) -> std::string
```

Get the value of a custom annotation property by its URI.

- **property_uri** [] The URI for the property

The value of the property or SBOL_ERROR_NOT_FOUND

**getPropertyValues** (property_uri)

```cpp
getPropertyValues(property_uri) -> std::vector< std::string >
```

Get all values of a custom annotation property by its URI.

- **property_uri** [] The URI for the property

A vector of property values or SBOL_ERROR_NOT_FOUND

**getTypeURI** ()

```cpp
getTypeURI() -> rdf_type
```

The uniform resource identifier that describes the RDF-type of this SBOL Object

**setAnnotation** (property_uri, val)

```cpp
setAnnotation(property_uri, val)
```

Set the value for a user-defined annotation property.

Synonymous with setPropertyValue If the value is a URI, it should be surrounded by angle brackets, else it will be interpreted as a literal value

**setPropertyValue** (property_uri, val)

```cpp
setPropertyValue(property_uri, val)
```

Set and overwrite the value for a user-defined annotation property.
Either a literal or URI value

```python
update_uri()
update_uri()
```

```python
class SampleRoster(*args)
```

A SampleRoster is a container used to group Builds that are included in an experiment together. A SampleRoster can be used to generate a Test in a Design- Build-Test-Learn workflow.

- **samples** [ReferencedObject] References to Builds which were tested in an experiment.
- **members** [URIProperty] The members property of a Collection is OPTIONAL and MAY contain a set of URI references to zero or more TopLevel objects.
- **attachments**: ReferencedObject
- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.
- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.
- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.
- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.
- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.
- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.
- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within
this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/dbtl.h

```c++
copy(*args)
copy(ns="", version="") -> SBOLClass &
```

**class SampleRosterProperty(*args)**

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The[]** SBOL specification currently supports string, URI, and integer literal values.

- **python_iter**: std::vector<std::string>::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```c++
add(new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value[]** A new string which will be added to a list of values.

addValidationRule(*args)
addValidationRule(rule)
```

clear()
clear()

Clear all property values.

```c++
copy(target_property)
copy(target_property)

Copy property values to a target object’s property fields.
```

```c++
find(query)
find(query) -> bool

Check if a value in this property matches the query.
```

```c++
getLowerBound()
getLowerBound() -> char
```

```c++
getOwner()
getOwner() -> SBOLObject &
```

```c++
getTypeURI()
getTypeURI() -> rdf_type
```

```c++
getUpperBound()
getUpperBound() -> char
```

```c++
remove(index=0)
remove(index=0)

Remove a property value.
```

```c++
set(*args)
set(new_value)
```
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

- **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
validate(arg=NULL)

write()
write()

class SearchQuery (*args)
A SearchQuery object is used to configure advanced searches for bioparts in a PartShop. Advanced searches are useful for matching values across multiple fields, or to specify multiple values in a single field.

- **objectType** [URIProperty] Set this property to indicate the type of SBOL object to search for. Set to SBOL_COMPONENT_DEFINITION by default.

- **limit** [IntProperty] Set this property to specify the total number of records to retrieve from a search request. By default 25 records are retrieved.

- **offset** [IntProperty] When the number of search hits exceeds the limit, the offset property can be used to retrieve more records.

- **attachments** : ReferencedObject

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.
• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/partshop.h

**class SearchResponse**

A SearchResponse is a container of search records returned by a search request.

- **records**: std::vector< sbol::Identified *>

- **python_iter**: std::vector< Identified * >::iterator

- **attachments**: ReferencedObject

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.
• **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/partshop.h

```cpp
end() -> iterator

extend(response)

Add more search records to an existing SearchResponse.

* response [] A SearchResponse object
```

class Sequence (*args)

The primary structure (eg, nucleotide or amino acid sequence) of a ComponentDefinition object.

• **elements** [TextProperty ] The elements property is a REQUIRED String of characters that represents the constituents of a biological or chemical molecule. For example, these characters could represent the nucleotide bases of a molecule of DNA, the amino acid residues of a protein, or the atoms and chemical bonds of a small molecule.

• **encoding** [URIProperty ] The encoding property is REQUIRED and has a data type of URI.

This property MUST indicate how the elements property of a Sequence MUST be formed and interpreted. For example, the elements property of a Sequence with an IUPAC DNA encoding property MUST contain characters that represent nucleotide bases, such as a, t, c, and g. The elements property of a Sequence with a Simplified Molecular-Input Line-Entry System (SMILES) encoding, on the other hand, MUST contain characters that represent atoms and chemical bonds, such as C, N, O, and =. It is RECOMMENDED that the encoding property contains a URI from the table below. The terms in the table are organized by the type of ComponentDefinition that typically refer to a Sequence with such an encoding. When the encoding of a Sequence is well described by one of the URIs in the table, it MUST contain that URI. ComponentDefinition type

Encoding

libSBOL Symbol

URI

DNA, RNA

IUPAC DNA, RNA

SBOL_ENCODING_IUPAC

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

Protein

IUPAC Protein

SBOL_ENCODING_IUPAC_PROTEIN

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

Small Molecule

SMILES
SBOL_ENCODING_SMILES

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

• attachments : ReferencedObject

• persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• version [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• wasDerivedFrom [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• wasGeneratedBy [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• name [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• description [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• identity [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified object's identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/sequence.h

assemble(*args)
assemble(composite_sequence="") -> std::string

Calculates the complete sequence of a high-level Component from the sequence of its subcomponents.

Prior to assembling the complete sequence, you must assemble a template design by calling ComponentDefinition::assemble for the ComponentDefinition that references this Sequence.
• **composite_sequence** [ ] Typically no value for the composite sequence should be specified by the user. This parameter is used to hold the composite sequence as it is passed to function calls at a higher-level of the recursion stack.

```python
compile()
    compile() -> std::string
```

Synonymous with Sequence::assemble. Calculates the complete sequence of a high-level Component from the sequence of its subcomponents. Prior to assembling the the complete sequence, you must assemble a template design by calling ComponentDefinition::assemble for the ComponentDefinition that references this Sequence.

```python
copy(*args)
    copy(ns="", version="") -> SBOLClass &
```

```python
length()
    length() -> int
```

The length of the primary sequence in the elements property

```python
synthesize(clone_id)
    synthesize(clone_id) -> ComponentDefinition &
```

• **clone_id** [ ] A URI for the build, or displayId if working in SBOLCompliant mode.

### class SequenceAnnotation (*args*)

The SequenceAnnotation class describes one or more regions of interest on the Sequence objects referred to by its parent ComponentDefinition. In addition, SequenceAnnotation objects can describe the substructure of their parent ComponentDefinition through association with the Component objects contained by this ComponentDefinition.

• **component** [ReferencedObject ] The component property is OPTIONAL and has a data type of URI. This URI MUST refer to a Component that is contained by the same parent ComponentDefinition that contains the SequenceAnnotation. In this way, the properties of the SequenceAnnotation, such as its description and locations, are associated with part of the substructure of its parent ComponentDefinition.

• **locations** [OwnedObject< Location > ] The locations property is a REQUIRED set of one or more Location objects that indicate which elements of a Sequence are described by the SequenceAnnotation.

Allowing multiple Location objects on a single SequenceAnnotation is intended to enable representation of discontinuous regions (for example, a Component encoded across a set of exons with interspersed introns). As such, the Location objects of a single SequenceAnnotation SHOULD NOT specify overlapping regions, since it is not clear what this would mean. There is no such concern with different SequenceAnnotation objects, however, which can freely overlap in Location (for example, specifying overlapping linkers for sequence assembly).

• **roles** [URIProperty ] Alternatively to describing substructure, a SequenceAnnotation can be utilized to identify a feature, such as a GenBank feature, of a specified Sequence.

In this use case, the SequenceAnnotation MUST NOT have a component property, but instead it would have a roles property. The roles property comprises an OPTIONAL set of zero or more URIs describing the specified sequence feature being annotated. If provided, these role URIs MUST identify terms from appropriate ontologies. Roles are not restricted to describing biological function; they may annotate Sequences function in any domain for which an ontology exists. It is RECOMMENDED that these role URIs identify terms that are compatible with the type properties of this SequenceAnnotations parent ComponentDefinition. For example, a role of a SequenceAnnotation which belongs to a ComponentDefinition of type DNA might refer to terms from the Sequence Ontology. See documentation for ComponentDefinition for a table of recommended ontology terms.
• **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/sequenceannotation.h

```cpp
contains(*args)
contains(comparand_list) -> std::vector< SequenceAnnotation * >

extract(start_reference=1)
extract(start_reference=1) -> ComponentDefinition &
Convert a SequenceAnnotation to a subcomponent.
A ComponentDefinition representing the subcomponent

follows(*args)
follows(comparand_list) -> std::vector< SequenceAnnotation * >
```
### length()
`length() -> int`

The length of a SequenceAnnotation in base coordinates.

### overlaps(*args)
`overlaps(comparand_list) -> std::vector< SequenceAnnotation * >`

### precedes(*args)
`precedes(comparand_list) -> std::vector< SequenceAnnotation * >`

#### class SequenceAnnotationProperty(*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

#### add(new_value)
`add(new_value)`

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

#### addValidationRule(*args)
`addValidationRule(rule)`

#### clear()
`clear()`

Clear all property values.

#### copy(target_property)
`copy(target_property)`

Copy property values to a target object’s property fields.

#### find(query)
`find(query) -> bool`

Check if a value in this property matches the query.

#### getLowerBound()
`getLowerBound() -> char`

#### getOwner()
`getOwner() -> SBOLObject &`

#### getTypeURI()
`getTypeURI() -> rdf_type`

#### getUpperBound()
`getUpperBound() -> char`

#### remove(index=0)
`remove(index=0)`

Remove a property value.
set (*args)
    set(new_value)

    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
    • new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
    validate(arg=NULL)

write ()
    write()

class SequenceConstraint (*args)

    The SequenceConstraint class can be used to assert restrictions on the relative, sequence-based positions of pairs of Component objects contained by the same parent ComponentDefinition. The primary purpose of this class is to enable the specification of partially designed ComponentDefinition objects, for which the precise positions or orientations of their contained Component objects are not yet fully determined.

    • subject [ReferencedObject] The subject property is REQUIRED and MUST contain a URI that refers to a Component contained by the same parent ComponentDefinition that contains the SequenceConstraint.

    • object [ReferencedObject] The object property is REQUIRED and MUST contain a URI that refers to a Component contained by the same parent ComponentDefinition that contains the SequenceConstraint. This Component MUST NOT be the same Component that the SequenceConstraint refers to via its subject property.

    • restriction [URIProperty] The restriction property is REQUIRED and has a data type of URI. This property MUST indicate the type of structural restriction on the relative, sequence-based positions or orientations of the subject and object Component objects. The URI value of this property SHOULD come from the RECOMMENDED URIs in the following table. libSBOL Symbol

    Description
    SBOL_RESTRICTION_PRECEDES
    The position of the subject Component MUST precede that of the object Component.
    If each one is associated with a SequenceAnnotation, then the SequenceAnnotation associated with the subject Component MUST specify a region that starts before the region specified by the SequenceAnnotation associated with the object Component.

    SBOL_RESTRICTION_SAME_ORIENTATION_AS
    The subject and object Component objects MUST have the same orientation. If each one is associated with a SequenceAnnotation, then the orientation URIs of the Location objects of the first SequenceAnnotation MUST be among those of the second SequenceAnnotation, and vice versa.

    SBOL_RESTRICTION_OPPOSITE_ORIENTATION_AS
    The subject and object Component objects MUST have opposite orientations. If each one is associated with a SequenceAnnotation, then the orientation URIs of the Location objects of one SequenceAnnotation MUST NOT be among those of the other SequenceAnnotation.

    • persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

    • displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable,
but more human-readable than the full URI of an identity. If the displayId property is used, then its
String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed
of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version
numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This
convention represents versions as sequences of numbers and qualifiers that are separated by the char-
acters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full
explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of
URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from
which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an
SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version
of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own
wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property
and those of other SBOL objects. For example, the reference chain A was derived from B and B was
derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a de-
sign process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is
intended to be displayed to a human when visualizing an Identified object. If an Identified object
lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is
RECOMMENDED that software tools give users the ability to switch perspectives between name
properties that are human-readable and displayId properties that are less human-readable, but are
more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This
property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type
of URI. A given Identified objects identity URI MUST be globally unique among all other identity
URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain
over which the user has control. Namely, the user can guarantee uniqueness of identities within
this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification
document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/sequenceconstraint.h

class SequenceConstraintProperty (*args)

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class
converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter** : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add(new_value)

add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.
addValidationRule(*args)
    addValidationRule(rule)

clear()
clear()
    Clear all property values.

copy(target_property)
copy(target_property)
    Copy property values to a target object’s property fields.

find(query)
    find(query) -> bool
    Check if a value in this property matches the query.

getLowerBound()
getLowerBound() -> char

getOwner()
getOwner() -> SBOLObject &

getTypeURI()
getTypeURI() -> rdf_type

getUpperBound()
    getUpperBound() -> char

remove(index=0)
    remove(index=0)
    Remove a property value.

set(*args)
    set(new_value)
    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

    • new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
    validate(arg=NULL)

write()
write()

class SequenceProperty(*args)
    Member properties of all SBOL objects are defined using a Property object.

    The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

    • The [] SBOL specification currently supports string, URI, and integer literal values.

    • python_iter : std::vector<std::string>::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

dd(new_value)
    add(new_value)
    Appends the new value to a list of values, for properties that allow it.
• **new_value** [] A new string which will be added to a list of values.

```
addValidationRule (*args)
    addValidationRule(rule)
```

```
clear()
    clear()
    Clear all property values.
```

```
copy (target_property)
    copy(target_property)
    Copy property values to a target object’s property fields.
```

```
find (query)
    find(query) -> bool
    Check if a value in this property matches the query.
```

```
getLowerBound()
    getLowerBound() -> char
```

```
getOwner()
    getOwner() -> SBOLObject &
```

```
getTypeURI()
    getTypeURI() -> rdf_type
```

```
getUpperBound()
    getUpperBound() -> char
```

```
remove (index=0)
    remove(index=0)
    Remove a property value.
```

```
set (*args)
    set(new_value)
    Basic setter for SBOL IntProperty, but can be used with TextProperty as well.
```

• **new_value** [] A new integer value for the property, which is converted to a raw string during serialization.

```
validate (arg=None)
    validate(arg=NULL)
```

```
write()
    write()
```

**class SmallMoleculeActivationInteraction (uri, ligand, transcription_factor)**

• **ligand** : AliasedProperty< FunctionalComponent >

• **transcriptionFactor** : AliasedProperty< FunctionalComponent >

• **types** [URIProperty ] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

    The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See http://www.ebi.ac.uk/sbo/main/) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type
URI for SBO Term
LibSBOL symbol
Biochemical Reaction
http://identifiers.org/biomodels.sbo/SBO:0000176
SBO_BIOCHEMICAL_REACTION
Inhibition
http://identifiers.org/biomodels.sbo/SBO:0000169
SBO_INHIBITION
Stimulation
http://identifiers.org/biomodels.sbo/SBO:0000170
SBO_STIMULATION
Genetic Production
http://identifiers.org/biomodels.sbo/SBO:0000589
SBO_GENETIC_PRODUCTION
Non-Covalent Binding
http://identifiers.org/biomodels.sbo/SBO:0000177
SBO_NONCOVALENT_BINDING
Degradation
http://identifiers.org/biomodels.sbo/SBO:0000179
SBO_DEGRADATION
Control
http://identifiers.org/biomodels.sbo/SBO:0000168
SBO_CONTROL

• participations [OwnedObject< Participation > ] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer might want to specify that an Interaction will exist, even if its participants have not yet been determined.

• functionalComponents : OwnedObject< FunctionalComponent >

• persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

```python
class SmallMoleculeInhibitionInteraction (uri, ligand, transcription_factor):
    ligand : AliasedProperty< FunctionalComponent >
    transcriptionFactor : AliasedProperty< FunctionalComponent >
    types [URIProperty] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

    The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See http://www.ebi.ac.uk/sbo/main/) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type

    URI for SBO Term
    LibSBOL symbol
    Biochemical Reaction

    http://identifiers.org/biomodels.sbo/SBO:0000176
    SBO_BIOCHEMICAL_REACTION
    Inhibition
```
participations  [OwnedObject< Participation > ] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer might want to specify that an Interaction will exist, even if its participants have not yet been determined.

functionalComponents  : OwnedObject< FunctionalComponent >

persistentIdentity  [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

displayId  [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

version  [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

wasDerivedFrom  [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own
wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, e.g., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

class Test (*args)
A Test is a container for experimental data. A Test is the product of the third step of libSBOL’s formalized Design-Build-Test-Analyze workflow.

- **samples** [ReferencedObject] References to Builds which were tested in the experiment.

- **dataFiles** [ReferencedObject] References to file Attachments which contain experimental data sets.

- **members** [URIProperty] The members property of a Collection is OPTIONAL and MAY contain a set of URI references to zero or more TopLevel objects.

- **attachments** : ReferencedObject

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property
and those of other SBOL objects. For example, the reference chain A was derived from B and B was
derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a de-
sign process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is
intended to be displayed to a human when visualizing an Identified object. If an Identified object
lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is
RECOMMENDED that software tools give users the ability to switch perspectives between name
properties that are human-readable and displayId properties that are less human-readable, but are
more likely to be unique.

• **description** [TextProperty ] The description property is OPTIONAL and has a data type of String. This
property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type
of URI. A given Identified objects identity URI MUST be globally unique among all other identity
URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain
over which the user has control. Namely, the user can guarantee uniqueness of identities within
this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification
doucment.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/dbtl.h

copy (*args)
copy(ns="", version="") -> SBOLClass &

class TestProperty (*args)
Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class
converts SBOL data structures into RDF triples.

• **The** [] SBOL specification currently supports string, URI, and integer literal values.

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

add (new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

• **new_value** [] A new string which will be added to a list of values.

addValidationRule (*args)
addValidationRule(rule)

clear()
clear()

Clear all property values.

copy (target_property)
copy(target_property)

Copy property values to a target object’s property fields.

copy (query)
copy(query) -> bool
Check if a value in this property matches the query.

```cpp
getLowerBound()
getLowerBound() -> char

getOwner()
getOwner() -> SBOLObject &

getTypeURI()
getTypeURI() -> rdf_type

getUpperBound()
getUpperBound() -> char

remove(index=0)
remove(index=0)
Remove a property value.

set(*args)
set(new_value)
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

  * new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
validate(arg=NULL)

write()
write()
```

```cpp
class TextProperty(*args)
TextProperty objects are used to contain string literals.

They can be used as member objects inside custom SBOL Extension classes.

  * python_iter : std::vector<std::string>::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/properties.h

get()
get() -> std::string
Basic getter for all SBOL literal properties.

A string literal

getAll()
getAll() -> std::vector<std::string>
```

```cpp
class TopLevel(*args)
All SBOL classes derived from TopLevel appear as top level nodes in the RDF/XML document tree and SBOL files. An abstract class.

  * attachments : ReferencedObject

  * persistentIdentity [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

  * displayId [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable,
but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/toplevel.h

```cpp
void copy ( *args)
    copy(ns="", version="") -> SBOLClass &

initialize (uri)
    initialize(uri)
```

class TranscriptionalActivationInteraction (uri, activator, target_promoter)

- **activator** : AliasedProperty< FunctionalComponent >

- **targetPromoter** : AliasedProperty< FunctionalComponent >

- **types** [URIProperty] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See http://www.ebi.ac.uk/sbo/main/) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type
URI for SBO Term
LibSBOL symbol
Biochemical Reaction
http://identifiers.org/biomodels.sbo/SBO:0000176
SBO_BIOCHEMICAL_REACTION
Inhibition
http://identifiers.org/biomodels.sbo/SBO:0000169
SBO_INHIBITION
Stimulation
http://identifiers.org/biomodels.sbo/SBO:0000170
SBO_STIMULATION
Genetic Production
http://identifiers.org/biomodels.sbo/SBO:0000589
SBO_GENETIC_PRODUCTION
Non-Covalent Binding
http://identifiers.org/biomodels.sbo/SBO:0000177
SBO_NONCOVALENT_BINDING
Degradation
http://identifiers.org/biomodels.sbo/SBO:0000179
SBO_DEGRADATION
Control
http://identifiers.org/biomodels.sbo/SBO:0000168
SBO_CONTROL

- **participations** [OwnedObject< Participation >] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer might want to specify that an Interaction will exist, even if its participants have not yet been determined.

- **functionalComponents** [OwnedObject< FunctionalComponent >]

- **persistentIdentity** [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULd be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.
• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, e.g., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

class TranscriptionalRepressionInteraction (uri, repressor, target_promoter)

• repressor : AliasedProperty< FunctionalComponent >

• targetPromoter : AliasedProperty< FunctionalComponent >

• **types** [URIProperty] The types property is a REQUIRED set of URIs that describes the behavior represented by an Interaction.

  The types property MUST contain one or more URIs that MUST identify terms from appropriate ontologies. It is RECOMMENDED that at least one of the URIs contained by the types property refer to a term from the occurring entity branch of the Systems Biology Ontology (SBO). (See http://www.ebi.ac.uk/sbo/main/) The following table provides a list of possible SBO terms for the types property and their corresponding URIs. Type

URI for SBO Term
LibSBOL symbol
Biochemical Reaction
http://identifiers.org/biomodels.sbo/SBO:0000176
SBO_BIOCHEMICAL_REACTION
Inhibition
participations  [OwnedObject< Participation > ] The participations property is an OPTIONAL and MAY contain a set of Participation objects, each of which identifies the roles that its referenced FunctionalComponent plays in the Interaction. Even though an Interaction generally contains at least one Participation, the case of zero Participation objects is allowed because it is plausible that a designer might want to specify that an Interaction will exist, even if its participants have not yet been determined.

• functionalComponents : OwnedObject< FunctionalComponent >

• persistentIdentity  [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

• displayId  [TextProperty ] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• version  [VersionProperty ] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters . and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• wasDerivedFrom  [URIProperty ] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own
wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject ] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty ] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty ] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty ] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

```class URIProperty (*args)
A URIProperty may contain a restricted type of string that conforms to the specification for a Uniform Resource Identifier (URI), typically consisting of a namespace authority followed by an identifier.

A URIProperty often contains a reference to an SBOL object or may contain an ontology term.

- **python_iter** : std::vector< std::string >::iterator
C++ includes: /Users/bbartley/Dev/git/libSBOL/source/properties.h

```get ( )
get() -> std::string
Get first URI.
A string of characters used to identify a resource

```getAll ( )
getAll() -> std::vector< std::string >
```

```class Usage (*args)
How different entities are used in an Activity is specified with the Usage class, which is linked from an Activity through the qualifiedUsage relationship. A Usage is then linked to an Entity through the Entitys URI and the role of this entity is qualified with the hadRole property. When the wasDerivedFrom property is used together with the full provenance described here, the entity pointed at by the wasDerivedFrom property MUST be included in a Usage.

- **entity** [URIProperty ] The entity property is REQUIRED and MUST contain a URI which MAY refer to an SBOL Identified object.

- **roles** [URIProperty ] The hadRole property is REQUIRED and MAY contain a URI that refers to a particular term describing the usage of an entity referenced by the entity property.

- **persistentIdentity** [URIProperty ] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.
• **displayId** [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

• **version** [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This convention represents versions as sequences of numbers and qualifiers that are separated by the characters , and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

• **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

• **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

• **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

• **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

• **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/provo.h

```cpp
class UsageProperty (*args)
    Member properties of all SBOL objects are defined using a Property object.
    The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.
    • **The** [] SBOL specification currently supports string, URI, and integer literal values.
    • **python_iter**: std::vector< std::string >::iterator
```

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add(new_value)
    add(new_value)
    Appends the new value to a list of values, for properties that allow it.
```
• new_value [] A new string which will be added to a list of values.

addValidationRule(*args)
addValidationRule(rule)

clear()
clear()
Clear all property values.

copy(target_property)
copy(target_property)
Copy property values to a target object’s property fields.

find(query)
find(query) -> bool
Check if a value in this property matches the query.

getLowerBound()
getLowerBound() -> char

getOwner()
getOwner() -> SBOLObject &

getTypeURI()
getTypeURI() -> rdf_type

getUpperBound()
getUpperBound() -> char

remove(index=0)
remove(index=0)
Remove a property value.

set(*args)
set(new_value)
Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate(arg=None)
validate(arg=None)

write()
write()

class VariableComponent(*args)
The VariableComponent class can be used to specify a choice of ComponentDefinition objects for any new Component derived from a template Component in the template ComponentDefinition. This specification is made using the class properties variable, variants, variantCollections, and variantDerivations. While the variants, variantCollections, and variantDerivations properties are OPTIONAL, at least one of them MUST NOT be empty.

• variable [ReferencedObject] The variable property is REQUIRED and MUST contain a URI that refers to a template Component in the template ComponentDefinition. If the wasDerivedFrom property of a Component refers to this template Component, then the definition property of the derived Component MUST refer to one of the ComponentDefinition objects referred to by the variants property of the VariableComponent. If not, then this definition property MUST either (1) refer to one of the ComponentDefinition objects from a Collection referred to by the variantCollections property of the
VariableComponent, or (2) refer to a ComponentDefinition derived from a CombinatorialDerivation referred to by the variantDerivations property of the VariableComponent.

- **repeat**  [URIProperty] The repeat property is REQUIRED and has a data type of URI.
  This property specifies how many Component objects can be derived from the template Component during the derivation of a new ComponentDefinition. The URI value of this property MUST come from the REQUIRED operator URIs provided in the table below Operator URI Description

<table>
<thead>
<tr>
<th>Operator URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://sbols.org/v2#zeroOrOne">http://sbols.org/v2#zeroOrOne</a></td>
<td>No more than one Component in the derived ComponentDefinition SHOULD have a wasDerivedFrom property that refers to the template Component.</td>
</tr>
<tr>
<td><a href="http://sbols.org/v2#one">http://sbols.org/v2#one</a></td>
<td>Exactly one Component in the derived ComponentDefinition SHOULD have a wasDerivedFrom property that refers to the template Component.</td>
</tr>
<tr>
<td><a href="http://sbols.org/v2#zeroOrMore">http://sbols.org/v2#zeroOrMore</a></td>
<td>Any number of Component objects in the derived ComponentDefinition MAY have wasDerivedFrom properties that refer to the template Component.</td>
</tr>
<tr>
<td><a href="http://sbols.org/v2#oneOrMore">http://sbols.org/v2#oneOrMore</a></td>
<td>At least one Component in the derived ComponentDefinition SHOULD have a wasDerivedFrom property that refers to the template Component.</td>
</tr>
</tbody>
</table>

- **variants**  [ReferencedObject] The variants property is OPTIONAL and MAY contain zero or more URIs that each refer to a ComponentDefinition. This property specifies individual ComponentDefinition objects to serve as options when deriving a new Component from the template Component.

- **variantCollections**  [ReferencedObject] The variantCollections property is OPTIONAL and MAY contain zero or more URIs that each refer to a Collection. The members property of each Collection referred to in this way MUST NOT be empty and MUST refer only to ComponentDefinition objects. This property enables the convenient specification of existing groups of ComponentDefinition objects to serve as options when deriving a new Component from the template Component.

- **variantDerivations**  [ReferencedObject] The variantDerivations property is OPTIONAL and MAY contain zero or more URIs that each refer to a CombinatorialDerivation. This property enables the convenient specification of ComponentDefinition objects derived in accordance with another CombinatorialDerivation to serve as options when deriving a new Component from the template Component. The variantDerivations property of a VariableComponent MUST NOT refer to the CombinatorialDerivation that contains this VariableComponent (no cyclic derivations are allowed).

- **persistentIdentity**  [URIProperty] The persistentIdentity property is OPTIONAL and has a data type of URI. This URI serves to uniquely refer to a set of SBOL objects that are different versions of each other. An Identified object MUST be referred to using either its identity URI or its persistentIdentity URI.

- **displayId**  [TextProperty] The displayId property is an OPTIONAL identifier with a data type of String. This property is intended to be an intermediate between name and identity that is machine-readable, but more human-readable than the full URI of an identity. If the displayId property is used, then its String value SHOULD be locally unique (global uniqueness is not necessary) and MUST be composed of only alphanumeric or underscore characters and MUST NOT begin with a digit.

- **version**  [VersionProperty] If the version property is used, then it is RECOMMENDED that version numbering follow the conventions of semantic versioning, particularly as implemented by Maven. This
convention represents versions as sequences of numbers and qualifiers that are separated by the characters, and - and are compared in lexicographical order (for example, 1 < 1.3.1 < 2.0-beta). For a full explanation, see the linked resources.

- **wasDerivedFrom** [URIProperty] The wasDerivedFrom property is OPTIONAL and has a data type of URI. An SBOL object with this property refers to another SBOL object or non-SBOL resource from which this object was derived. If the wasDerivedFrom property of an SBOL object A that refers to an SBOL object B has an identical persistentIdentity, and both A and B have a version, then the version of B MUST precede that of A. In addition, an SBOL object MUST NOT refer to itself via its own wasDerivedFrom property or form a cyclical chain of references via its wasDerivedFrom property and those of other SBOL objects. For example, the reference chain A was derived from B and B was derived from A is cyclical.

- **wasGeneratedBy** [ReferencedObject] An Activity which generated this ComponentDefinition, eg., a design process like codon-optimization or a construction process like Gibson Assembly.

- **name** [TextProperty] The name property is OPTIONAL and has a data type of String. This property is intended to be displayed to a human when visualizing an Identified object. If an Identified object lacks a name, then software tools SHOULD instead display the objects displayId or identity. It is RECOMMENDED that software tools give users the ability to switch perspectives between name properties that are human-readable and displayId properties that are less human-readable, but are more likely to be unique.

- **description** [TextProperty] The description property is OPTIONAL and has a data type of String. This property is intended to contain a more thorough text description of an Identified object.

- **identity** [URIProperty] The identity property is REQUIRED by all Identified objects and has a data type of URI. A given Identified objects identity URI MUST be globally unique among all other identity URIs. The identity of a compliant SBOL object MUST begin with a URI prefix that maps to a domain over which the user has control. Namely, the user can guarantee uniqueness of identities within this domain. For other best practices regarding URIs see Section 11.2 of the SBOL specification document.

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/combinatorialderivation.h

```cpp
class VariableComponentProperty (*args)

Member properties of all SBOL objects are defined using a Property object.

The Property class provides a generic interface for accessing SBOL objects. At a low level, the Property class converts SBOL data structures into RDF triples.

- **The** [] SBOL specification currently supports string, URI, and integer literal values.

- **python_iter : std::vector< std::string >::iterator**

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/property.h

```cpp
add(new_value)
add(new_value)

Appends the new value to a list of values, for properties that allow it.

- **new_value** [] A new string which will be added to a list of values.

```cpp
addValidationRule (*args)
addValidationRule(rule)

Clear all property values.
copy (target_property)
copy(target_property)

Copy property values to a target object’s property fields.

find (query)
find(query) -> bool

Check if a value in this property matches the query.

getLowerBound ()
getLowerBound() -> char

getOwner ()
getOwner() -> SBOLObject &

getTypeURI ()
getTypeURI() -> rdf_type

getUpperBound ()
getUpperBound() -> char

remove (index=0)
remove(index=0)

Remove a property value.

set (*args)
set(new_value)

Basic setter for SBOL IntProperty, but can be used with TextProperty as well.

• new_value [] A new integer value for the property, which is converted to a raw string during serialization.

validate (arg=None)
validate(arg=NULL)

write ()
write()

class VersionProperty (property_owner, type_uri, lower_bound, upper_bound, initial_value)

Contains a version number for an SBOL object.

The VersionProperty follows Maven versioning semantics and includes a major, minor, and patch version number.

• python_iter : std::vector< std::string >::iterator

C++ includes: /Users/bbartley/Dev/git/libSBOL/source/properties.h

decrementMajor ()
decrementMajor()  

Decrement major version.

decrementMinor ()
decrementMinor()  

Decrement major version.

decrementPatch ()
decrementPatch()  

Decrement major version.
incrementMajor()
incrementMajor()
Increment major version.

incrementMinor()
incrementMinor()
Increment minor version.

incrementPatch()
incrementPatch()
Increment patch version.

major()
major() -> int
Get major version.
The major version as an integer Splits the version string by a delimiter and returns the major version number.

minor()
minor() -> int
Get minor version.
The minor version as an integer Splits the version string by a delimiter and returns the minor version number.

patch()
patch() -> int
Get patch version.
The patch version as an integer Splits the version string by a delimiter and returns the patch version.

getFileVersion()
getFileFormat() -> std::string SBOL_DECLSPEC
Returns currently accepted file format.

getHomespace()
getHomespace() -> SBOL_DECLSPEC std::string
Get the current default namespace for autocreation of URIs when a new SBOL object is created.

hasHomespace()
hasHomespace() -> SBOL_DECLSPEC int
Checks if a valid default namespace has been defined.

is_alphanumeric_or_underscore(c)
is_alphanumeric_or_underscore(c) -> bool

is_not_alphanumeric_or_underscore(c)
is_not_alphanumeric_or_underscore(c) -> bool

libsbol_rule_1 (sbol_obj, arg)
libsbol_rule_1 (sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_10 (sbol_obj, arg)
libsbol_rule_10 (sbol_obj, arg) -> SBOL_DECLSPEC void
libsbol_rule_11(sbol_obj, arg)
libsbol_rule_11(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_12(sbol_obj, arg)
libsbol_rule_12(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_13(sbol_obj, arg)
libsbol_rule_13(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_14(sbol_obj, arg)
libsbol_rule_14(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_15(sbol_obj, arg)
libsbol_rule_15(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_16(sbol_obj, arg)
libsbol_rule_16(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_17(sbol_obj, arg)
libsbol_rule_17(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_18(sbol_obj, arg)
libsbol_rule_18(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_19(sbol_obj, arg)
libsbol_rule_19(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_2(sbol_obj, arg)
libsbol_rule_2(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_20(sbol_obj, arg)
libsbol_rule_20(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_21(sbol_obj, arg)
libsbol_rule_21(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_22(sbol_obj, arg)
libsbol_rule_22(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_24(sbol_obj, arg)
libsbol_rule_24(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_3(sbol_obj, arg)
libsbol_rule_3(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_4(sbol_obj, arg)
libsbol_rule_4(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_5(sbol_obj, arg)
libsbol_rule_5(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_6(sbol_obj, arg)
libsbol_rule_6(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_7(sbol_obj, arg)
libsbol_rule_7(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_8(sbol_obj, arg)
libsbol_rule_8(sbol_obj, arg) -> SBOL_DECLSPEC void

libsbol_rule_9(sbol_obj, arg)
libsbol_rule_9(sbol_obj, arg) -> SBOL_DECLSPEC void
sbolRule10101(sbol_obj, arg)
sbolRule10101(sbol_obj, arg) -> SBOL_DECLSPEC void

sbolRule10102(sbol_obj, arg)
sbolRule10102(sbol_obj, arg) -> SBOL_DECLSPEC void

sbol_rule_10202(sbol_obj, arg)
sbol_rule_10202(sbol_obj, arg) -> SBOL_DECLSPEC void

sbol_rule_10204(sbol_obj, arg)
sbol_rule_10204(sbol_obj, arg) -> SBOL_DECLSPEC void

setFileFormat(file_format)
setFileFormat(file_format) -> SBOL_DECLSPEC void

Sets file format to use.

setHomespace(ns)
setHomespace(ns) -> SBOL_DECLSPEC void

Global methods.

Set the default namespace for autocreation of URIs when a new SBOL object is created

testRoundTrip()
Function to run test suite for pySBOL

testSBOL()
Function to test pySBOL API
CHAPTER 9

Indices and tables

- genindex
- modindex
- search
S
sbol.libsbol,31
A

Activity (class in sbol.libsbol), 31
ActivityProperty (class in sbol.libsbol), 33
add() (ActivityProperty method), 33
add() (AgentProperty method), 35
add() (AliasedOwnedFunctionalComponent method), 36
add() (AnalysisProperty method), 38
add() (AssociationProperty method), 40
add() (AttachmentProperty method), 42
add() (BuildProperty method), 45
add() (CollectionProperty method), 47
add() (CombinatorialDerivationProperty method), 49
add() (ComponentDefinitionProperty method), 58
add() (ComponentProperty method), 61
add() (DesignProperty method), 65
add() (ExperimentalDataProperty method), 75
add() (FunctionalComponentProperty method), 79
add() (ImplementationProperty method), 85
add() (InteractionProperty method), 88
add() (LocationProperty method), 90
add() (MapsToProperty method), 93
add() (MeasurementProperty method), 94
add() (ModuleProperty method), 97
add() (ModuleDefinitionProperty method), 101
add() (ModuleProperty method), 102
add() (OwnedActivity method), 103
add() (OwnedAgent method), 105
add() (OwnedAnalysis method), 108
add() (OwnedAssociation method), 110
add() (OwnedAttachment method), 112
add() (OwnedBuild method), 114
add() (OwnedCollection method), 117
add() (OwnedCombinatorialDerivation method), 119
add() (OwnedComponent method), 121
add() (OwnedComponentDefinition method), 123
add() (OwnedDesign method), 126
add() (OwnedExperiment method), 128

add() (OwnedExperimentalData method), 130
add() (OwnedFunctionalComponent method), 133
add() (OwnedImplementation method), 135
add() (OwnedInteraction method), 137
add() (OwnedLocation method), 139
add() (OwnedMapsTo method), 142
add() (OwnedMeasurement method), 144
add() (OwnedModel method), 146
add() (OwnedModule method), 148
add() (OwnedModuleDefinition method), 151
add() (OwnedParticipation method), 153
add() (OwnedPlan method), 155
add() (OwnedSampleRoster method), 158
add() (OwnedSequence method), 160
add() (OwnedSequenceAnnotation method), 162
add() (OwnedSequenceConstraint method), 164
add() (OwnedTest method), 167
add() (OwnedUsage method), 169
add() (OwnedVariableComponent method), 171
add() (ParticipationProperty method), 177
add() (PlanProperty method), 179
add() (ReferencedObject method), 182
add() (SampleRosterProperty method), 186
add() (SequenceAnnotationProperty method), 193
add() (SequenceConstraintProperty method), 195
add() (SequenceProperty method), 196
add() (TestProperty method), 202
add() (UsageProperty method), 209
add() (VariableComponentProperty method), 212
addComponentDefinition() (Document method), 67

daddDownstreamFlank() (ComponentDefinition method), 55
addModel() (Document method), 67
addModuleDefinition() (Document method), 67
addNamespace() (Document method), 67
addPropertyValue() (SBOLObject method), 183
addReference() (ReferencedObject method), 182
addSequence() (Document method), 67
addSynBioHubAnnotations() (PartShop method), 173
addUpstreamFlank() (ComponentDefinition method), 55
addValidationRule() (ActivityProperty method), 34
addValidationRule() (AgentProperty method), 35
addValidationRule() (AnalysisProperty method), 38
addValidationRule() (AssociationProperty method), 40
addValidationRule() (AttachmentProperty method), 42
addValidationRule() (BuildProperty method), 45
addValidationRule() (CollectionProperty method), 47
addValidationRule() (CombinatorialDerivationProperty method), 49
addValidationRule() (ComponentDefinitionProperty method), 58
addValidationRule() (ComponentProperty method), 61
addValidationRule() (DesignProperty method), 65
addValidationRule() (ExperimentalDataProperty method), 75
addValidationRule() (ExperimentProperty method), 74
addValidationRule() (FunctionalComponentProperty method), 79
addValidationRule() (ImplementationProperty method), 85
addValidationRule() (InteractionProperty method), 88
addValidationRule() (LocationProperty method), 90
addValidationRule() (MapsToProperty method), 93
addValidationRule() (MeasurementProperty method), 94
addValidationRule() (ModuleDefinitionProperty method), 101
addValidationRule() (ModuleProperty method), 102
addValidationRule() (ParticipationProperty method), 177
addValidationRule() (PlanProperty method), 179
addValidationRule() (SampleRosterProperty method), 186
addValidationRule() (SequenceAnnotationProperty method), 193
addValidationRule() (SequenceConstraintProperty method), 195
addValidationRule() (SequenceProperty method), 197
addValidationRule() (TestProperty method), 202
addValidationRule() (UsageProperty method), 210
addValidationRule() (VariableComponentProperty method), 212
adjoints() (Range method), 181
Agent (class in sbol.libsbol), 34
AgentProperty (class in sbol.libsbol), 35
AliasedOwnedFunctionalComponent (class in sbol.libsbol), 36
Analysis (class in sbol.libsbol), 37
AnalysisProperty (class in sbol.libsbol), 38
append() (Document method), 67
apply() (SBOLObject method), 183
assemble() (ComponentDefinition method), 56
assemble() (ModuleDefinition method), 100
assemble() (Sequence method), 190
assemblePrimaryStructure() (ComponentDefinition method), 56
Association (class in sbol.libsbol), 39
AssociationProperty (class in sbol.libsbol), 40
attachFile() (PartShop method), 173
Attachment (class in sbol.libsbol), 41
AttachmentProperty (class in sbol.libsbol), 42

B
Build (class in sbol.libsbol), 43
build() (ComponentDefinition method), 56
BuildProperty (class in sbol.libsbol), 44

C
cast() (SBOLObject method), 183
clear() (ActivityProperty method), 34
clear() (AgentProperty method), 36
clear() (AnalysisProperty method), 38
clear() (AssociationProperty method), 40
clear() (AttachmentProperty method), 43
clear() (BuildProperty method), 45
clear() (CollectionProperty method), 47
clear() (CombinatorialDerivationProperty method), 49
clear() (ComponentDefinitionProperty method), 58
clear() (ComponentProperty method), 61
clear() (DesignProperty method), 65
clear() (ExperimentalDataProperty method), 75
clear() (ExperimentProperty method), 74
clear() (FunctionalComponentProperty method), 79
clear() (ImplementationProperty method), 85
clear() (ModuleDefinitionProperty method), 101
clear() (ModuleProperty method), 102
clear() (ParticipationProperty method), 177
clear() (PlanProperty method), 179
clear() (SampleRosterProperty method), 186
clear() (SequenceAnnotationProperty method), 193
clear() (SequenceConstraintProperty method), 195

222
| clear() (BuildProperty method), 45 |
| copy() (Activity method), 32 |
| ComponentDefinition (class in sbol.libsbol), 52 |
| ComponentDefinitionProperty (class in sbol.libsbol), 58 |
| ComponentInstance (class in sbol.libsbol), 59 |
| ComponentProperty (class in sbol.libsbol), 60 |
| Config (class in sbol.libsbol), 61 |
| Config_getOption() (in module sbol.libsbol), 62 |
| Config_setOption() (in module sbol.libsbol), 62 |
| connect() (FunctionalComponent method), 78 |
| connect() (ModuleDefinition method), 100 |
| contains() (Range method), 181 |
| contains() (SequenceAnnotation method), 192 |
| copy() (ActivityProperty method), 34 |
| copy() (Agent method), 35 |
| copy() (AgentProperty method), 36 |
| copy() (Analysis method), 38 |
| copy() (AnalysisProperty method), 38 |
| copy() (AssociationProperty method), 41 |
| copy() (Attachment method), 42 |
| copy() (AttachmentProperty method), 43 |
| copy() (Build method), 44 |
| copy() (BuildProperty method), 45 |
| copy() (Collection method), 46 |
| copy() (CollectionProperty method), 47 |
| copy() (CombinatorialDerivation method), 49 |
| copy() (CombinatorialDerivationProperty method), 49 |
| copy() (ComponentDefinition method), 56 |
| copy() (ComponentDefinitionProperty method), 59 |
| copy() (ComponentProperty method), 61 |
| copy() (Design method), 64 |
| copy() (DesignProperty method), 65 |
| copy() (Document method), 67 |
| copy() (Experiment method), 73 |
| copy() (ExperimentalData method), 75 |
| copy() (ExperimentalDataProperty method), 75 |
| copy() (ExperimentProperty method), 74 |
| copy() (FunctionalComponentProperty method), 79 |
| copy() (Implementation method), 84 |
| copy() (ImplementationProperty method), 85 |
| copy() (Interaction method), 88 |
| copy() (LocationProperty method), 90 |
| copy() (MapsTo method), 93 |
| copy() (MeasurementProperty method), 94 |
| copy() (Model method), 96 |
| copy() (ModelProperty method), 97 |
| copy() (ModuleDefinition method), 100 |
| copy() (ModuleDefinitionProperty method), 101 |
| copy() (ModuleProperty method), 102 |
| copy() (ParticipationProperty method), 107 |
| copy() (Plan method), 179 |
| copy() (PlanProperty method), 179 |
| copy() (SampleRoster method), 186 |
| copy() (SampleRosterProperty method), 186 |
| compare() (SBOLObject method), 183 |
| Component (class in sbol.libsbol), 50 |
| ComponentDefinitionProperty (class in sbol.libsbol), 52 |
| ComponentProperty (class in sbol.libsbol), 58 |
| ComponentInstance (class in sbol.libsbol), 59 |
| ComponentProperty (class in sbol.libsbol), 60 |
| Config (class in sbol.libsbol), 61 |
| Config_getOption() (in module sbol.libsbol), 62 |
| Config_setOption() (in module sbol.libsbol), 62 |
| connect() (FunctionalComponent method), 78 |
| connect() (ModuleDefinition method), 100 |
| contains() (Range method), 181 |
| contains() (SequenceAnnotation method), 192 |
| copy() (Activity method), 32 |
| copy() (ActivityProperty method), 34 |
| copy() (Agent method), 35 |
| copy() (AgentProperty method), 36 |
| copy() (Analysis method), 38 |
| copy() (AnalysisProperty method), 38 |
| copy() (AssociationProperty method), 41 |
| copy() (Attachment method), 42 |
| copy() (AttachmentProperty method), 43 |
| copy() (Build method), 44 |
| copy() (BuildProperty method), 45 |
| copy() (Collection method), 46 |
| copy() (CollectionProperty method), 47 |
| copy() (CombinatorialDerivation method), 49 |
| copy() (CombinatorialDerivationProperty method), 49 |
| copy() (ComponentDefinition method), 56 |
| copy() (ComponentDefinitionProperty method), 59 |
| copy() (ComponentProperty method), 61 |
| copy() (Design method), 64 |
| copy() (DesignProperty method), 65 |
| copy() (Document method), 67 |
| copy() (Experiment method), 73 |
| copy() (ExperimentalData method), 75 |
| copy() (ExperimentalDataProperty method), 75 |
| copy() (ExperimentProperty method), 74 |
| copy() (FunctionalComponentProperty method), 79 |
| copy() (Implementation method), 84 |
| copy() (ImplementationProperty method), 85 |
| copy() (Interaction method), 88 |
| copy() (LocationProperty method), 90 |
| copy() (MapsTo method), 93 |
| copy() (MeasurementProperty method), 94 |
| copy() (Model method), 96 |
| copy() (ModelProperty method), 97 |
| copy() (ModuleDefinition method), 100 |
| copy() (ModuleDefinitionProperty method), 101 |
| copy() (ModuleProperty method), 102 |
| copy() (ParticipationProperty method), 107 |
| copy() (Plan method), 179 |
| copy() (PlanProperty method), 179 |
| copy() (SampleRoster method), 186 |
| copy() (SampleRosterProperty method), 186 |
createGenericLocation() (OwnedLocation method), 140
createRange() (OwnedLocation method), 140
createRange() (OwnedModule method), 149
createRange() (OwnedCombinatorialDerivation method), 171
createRange() (OwnedCombinatorialDerivation method), 171
createRange() (OwnedVariableComponent method), 169
createRange() (OwnedUsage method), 169
createRange() (OwnedPlan method), 156
createGenericLocation() (OwnedSampleRoster method), 158
createRange() (OwnedSampleRoster method), 158
createRange() (OwnedFunctionComponent method), 133
createGenericLocation() (OwnerModule method), 146
createGenericLocation() (OwnerModule method), 149
createGenericLocation() (OwnerModuleDefinition method), 151
createGenericLocation() (OwnerSequence method), 160
createRange() (OwnerSequence method), 160
createRange() (OwnerSequenceAnnotation method), 162
createRange() (OwnerSequenceConstraint method), 165
createRange() (OwnerTest method), 167
createRange() (OwnerUsage method), 169
Cut (class in sbol.libsbol), 62

D
DateTimeProperty (class in sbol.libsbol), 63
decrementMinor() (VersionProperty method), 213
decrementPatch() (VersionProperty method), 213
define() (AliasedOwnedFunctionalComponent method), 36
define() (OwnedActivity method), 103
define() (OwnedAgent method), 106
define() (OwnedAnalysis method), 108
define() (OwnedAssociation method), 110
define() (OwnedAttachment method), 112
define() (OwnedBuild method), 115
define() (OwnedCollection method), 117
define() (OwnedCombinatorialDerivation method), 119
define() (OwnedComponent method), 121
define() (OwnedComponentDefinition method), 124
define() (OwnedDesign method), 126
define() (OwnedExperiment method), 128
define() (OwnedExperimentalData method), 131
define() (OwnedFunctionComponent method), 133
define() (OwnedImplementation method), 135
define() (OwnedInteraction method), 137
define() (OwnedLocation method), 140
define() (OwnedMeasurement method), 144
define() (OwnedModel method), 146
define() (OwnedMapsTo method), 149
define() (OwnedModule method), 149
define() (OwnerComponentDefinition method), 153
define() (OwnerCombinatorialDerivation method), 153
define() (OwnerComponent method), 156
define() (OwnerComponentDefinition method), 156
define() (OwnerDesign method), 158
define() (OwnerFunctionComponent method), 158
define() (OwnerFunctionComponent method), 160
define() (OwnerFunctionComponent method), 162
createRange() (OwnerFunctionComponent method), 162
createRange() (OwnerFunctionComponent method), 165
createRange() (OwnerFunctionComponent method), 167
createRange() (OwnerFunctionComponent method), 169
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
createRange() (OwnerFunctionComponent method), 171
define() (OwnedModule method), 149
define() (OwnedModuleDefinition method), 151
define() (OwnedParticipation method), 153
define() (OwnedPlan method), 156
define() (OwnedSampleRoster method), 158
define() (OwnedSequence method), 160
define() (OwnedSequenceAnnotation method), 162
define() (OwnedSequenceConstraint method), 165
define() (OwnedTest method), 167
define() (OwnedUsage method), 169
define() (OwnedVariableComponent method), 171
define() (Participation method), 177
Design (class in sbol.libsbol), 63
DesignProperty (class in sbol.libsbol), 64
disassemble() (ComponentDefinition method), 56
Document (class in sbol.libsbol), 65
downloadAttachment() (PartShop method), 174
end() (Document method), 67
end() (SearchResponse method), 189
EnzymeCatalysisInteraction (class in sbol.libsbol), 72
Experiment (class in sbol.libsbol), 73
ExperimentalData (class in sbol.libsbol), 75
ExperimentalDataProperty (class in sbol.libsbol), 75
ExperimentProperty (class in sbol.libsbol), 74
extend() (SearchResponse method), 189
extract() (SequenceAnnotation method), 192
find() (ActivityProperty method), 34
find() (AgentProperty method), 36
find() (AnalysisProperty method), 39
find() (AssociationProperty method), 41
find() (AttachmentProperty method), 43
find() (BuildProperty method), 45
find() (CollectionProperty method), 47
find() (CombinatorialDerivationProperty method), 50
find() (ComponentDefinitionProperty method), 59
find() (ComponentProperty method), 61
find() (DesignProperty method), 65
find() (Document method), 67
find() (ExperimentalDataProperty method), 75
find() (ExperimentProperty method), 74
find() (FunctionalComponentProperty method), 79
find() (ImplementationProperty method), 85
find() (InteractionProperty method), 88
find() (LocationProperty method), 90
find() (MapsToProperty method), 93
find() (MeasurementProperty method), 94
find() (ModelProperty method), 97
find() (ModuleDefinitionProperty method), 101
find() (ModuleProperty method), 102
find() (OwnedActivity method), 104
find() (OwnedAgent method), 106
find() (OwnedAnalysis method), 108
find() (OwnedAssociation method), 110
find() (OwnedAttachment method), 113
find() (OwnedBuild method), 115
find() (OwnedCollection method), 117
find() (OwnedCombinatorialDerivation method), 119
find() (OwnedComponent method), 122
find() (OwnedComponentDefinition method), 124
find() (OwnedDesign method), 126
find() (OwnedExperiment method), 129
find() (OwnedExperimentalData method), 131
find() (OwnedFunctionalComponent method), 133
find() (OwnedImplementation method), 135
find() (OwnedInteraction method), 138
find() (OwnedLocation method), 140
find() (OwnedMapsTo method), 142
find() (OwnedMeasurement method), 144
find() (OwnedModel method), 147
find() (OwnedModule method), 149
find() (OwnedModuleDefinition method), 151
find() (OwnedParticipation method), 154
find() (OwnedPlan method), 156
find() (OwnedSampleRoster method), 158
find() (OwnedSequence method), 160
find() (OwnedSequenceAnnotation method), 163
find() (OwnedSequenceConstraint method), 165
find() (OwnedTest method), 167
find() (OwnedUsage method), 169
find() (OwnedVariableComponent method), 172
find() (ParticipationProperty method), 177
find() (PlanProperty method), 179
find() (SampleRosterProperty method), 186
find() (SBOLObject method), 183
find() (SequenceAnnotationProperty method), 193
find() (SequenceConstraintProperty method), 196
find() (SequenceProperty method), 197
find() (TestProperty method), 202
find() (UsageProperty method), 210
find() (VariableComponentProperty method), 213
find_property() (Document method), 68
find_property() (SBOLObject method), 183
find_property_value() (SBOLObject method), 183
find_reference() (Document method), 68
find_reference() (SBOLObject method), 183
FloatProperty (class in sbol.libsbol), 76
follows() (Range method), 181
follows() (SequenceAnnotation method), 192
FunctionalComponent (class in sbol.libsbol), 76
FunctionalComponentProperty (class in sbol.libsbol), 78
GeneProductionInteraction (class in sbol.libsbol, 80)
generate() (Document method, 68)
generateAnalysis() (Activity method, 32)
generateBuild() (Activity method, 33)
generateDesign() (Activity method, 33)
generateTest() (Activity method, 33)
GenericLocation (class in sbol.libsbol, 81)
get() (AliasedOwnedFunctionalComponent method), 37
get() (FloatProperty method), 76
get() (IntProperty method), 86
get() (OwnedActivity method), 104
get() (OwnedAgent method), 106
get() (OwnedAnalysis method), 108
get() (OwnedAssociation method), 111
get() (OwnedAttachment method), 113
get() (OwnedBuild method), 115
get() (OwnedCollection method), 117
get() (OwnedCombinatorialDerivation method), 120
get() (OwnedComponent method), 122
get() (OwnedComponentDefinition method), 124
get() (OwnedDesign method), 126
get() (OwnedExperiment method), 129
get() (OwnedExperimentalData method), 131
get() (OwnedFunctionComponent method), 133
get() (OwnImplementation method), 136
get() (OwnedInteraction method), 138
get() (OwnedLocation method), 140
get() (OwnedMapsTo method), 143
get() (OwnedMeasurement method), 145
get() (OwnedModel method), 147
get() (OwnedModule method), 149
get() (OwnedModuleDefinition method), 152
get() (OwnedParticipation method), 154
get() (OwnedPlan method), 156
get() (OwnedSampleRoster method), 158
get() (OwnedSequence method), 161
get() (OwnedSequenceAnnotation method), 163
get() (OwnedSequenceConstraint method), 165
get() (OwnedTest method), 167
get() (OwnedUsage method), 170
get() (OwnedVariableComponent method), 172
get() (TextProperty method), 203
get() (URIProperty method), 208
getAnalysis() (Document method), 68
getAnnotation() (SBOObject method), 184
getAttachment() (Document method), 68
getBuild() (Document method), 68
getClassName() (SBOObject method), 184
getCollection() (Document method), 69
getCombinatorialDerivation() (Document method), 69
getComponentDefinition() (Document method), 69
getCut() (OwnedActivity method), 104
getCut() (OwnedAgent method), 106
getCut() (OwnedAnalysis method), 109
getCut() (OwnedAssociation method), 111
getCut() (OwnedAttachment method), 113
getCut() (OwnedBuild method), 115
getCut() (OwnedCollection method), 118
getCut() (OwnedCombinatorialDerivation method), 120
getCut() (OwnedComponent method), 122
getCut() (OwnedComponentDefinition method), 124
getCut() (OwnedDesign method), 127
getCut() (OwnedExperiment method), 129
getCut() (OwnedExperimentalData method), 131
getCut() (OwnedFunctionComponent method), 133
getCut() (OwnedImplementation method), 136
getCut() (OwnedInteraction method), 138
getCut() (OwnedLocation method), 140
getCut() (OwnedMapsTo method), 143
getCut() (OwnedMeasurement method), 145
getCut() (OwnedModel method), 147
getCut() (OwnedModule method), 149
getCut() (OwnedModuleDefinition method), 152
getCut() (OwnedParticipation method), 154
getCut() (OwnedPlan method), 156
getCut() (OwnedSampleRoster method), 158
getCut() (OwnedSequence method), 161
getCut() (OwnedSequenceAnnotation method), 163
getCut() (OwnedSequenceConstraint method), 165
getCut() (OwnedTest method), 167
getCut() (OwnedUsage method), 170
getCut() (OwnedVariableComponent method), 172
getCut() (TextProperty method), 203
getCut() (URIProperty method), 208
getCut() (Document method), 68
getCut() (OwnedActivity method), 104
getCut() (OwnedAgent method), 106
getCut() (OwnedAnalysis method), 109
getCut() (OwnedAssociation method), 111
getCut() (OwnedAttachment method), 113
getCut() (OwnedBuild method), 115
getCut() (OwnedCollection method), 118
getCut() (OwnedCombinatorialDerivation method), 120
getCut() (OwnedComponent method), 122
getCut() (OwnedComponentDefinition method), 124
getCut() (OwnedDesign method), 127
getCut() (OwnedExperiment method), 129
getCut() (OwnedExperimentalData method), 131
getCut() (OwnedFunctionComponent method), 133
getCut() (OwnedImplementation method), 136
getCut() (OwnedInteraction method), 138
getCut() (OwnedLocation method), 140
getCut() (OwnedMapsTo method), 143
getCut() (OwnedMeasurement method), 145
getCut() (OwnedModel method), 147
getCut() (OwnedModule method), 149
getCut() (OwnedModuleDefinition method), 152
getCut() (OwnedParticipation method), 154
getCut() (OwnedPlan method), 156
getCut() (OwnedSampleRoster method), 158
getCut() (OwnedSequence method), 161
getCut() (OwnedSequenceAnnotation method), 163
getCut() (OwnedSequenceConstraint method), 165
getCut() (OwnedTest method), 167
getCut() (OwnedUsage method), 170
getCut() (OwnedVariableComponent method), 172
getCut() (TextProperty method), 203
getCut() (URIProperty method), 208
getCut() (Document method), 68
getCut() (OwnedActivity method), 104
getCut() (OwnedAgent method), 106
getCut() (OwnedAnalysis method), 109
getCut() (OwnedAssociation method), 111
getCut() (OwnedAttachment method), 113
getCut() (OwnedBuild method), 115
getCut() (OwnedCollection method), 118
getCut() (OwnedCombinatorialDerivation method), 120
getCut() (OwnedComponent method), 122
getCut() (OwnedComponentDefinition method), 124
getCut() (OwnedDesign method), 127
getCut() (OwnedExperiment method), 129
getCut() (OwnedExperimentalData method), 131
getCut() (OwnedFunctionComponent method), 133
getCut() (OwnedImplementation method), 136
getCut() (OwnedInteraction method), 138
getCut() (OwnedLocation method), 140
getCut() (OwnedMapsTo method), 143
getCut() (OwnedMeasurement method), 145
getCut() (OwnedModel method), 147
getCut() (OwnedModule method), 149
getCut() (OwnedModuleDefinition method), 152
getCut() (OwnedParticipation method), 154
getCut() (OwnedPlan method), 156
getCut() (OwnedSampleRoster method), 158
getCut() (OwnedSequence method), 161
getCut() (OwnedSequenceAnnotation method), 163
getCut() (OwnedSequenceConstraint method), 165
getCut() (OwnedTest method), 167
getCut() (OwnedUsage method), 170
getCut() (OwnedVariableComponent method), 172
getCut() (TextProperty method), 203
getCut() (URIProperty method), 208
getCut() (Document method), 68
getCut() (OwnedActivity method), 104
getCut() (OwnedAgent method), 106
getCut() (OwnedAnalysis method), 109
getCut() (OwnedAssociation method), 111
getCut() (OwnedAttachment method), 113
getCut() (OwnedBuild method), 115
getCut() (OwnedCollection method), 118
getCut() (OwnedCombinatorialDerivation method), 120
getCut() (OwnedComponent method), 122
getCut() (OwnedComponentDefinition method), 124
getCut() (OwnedDesign method), 127
getCut() (OwnedExperiment method), 129
getCut() (OwnedExperimentalData method), 131
getCut() (OwnedFunctionComponent method), 133
getCut() (OwnedImplementation method), 136
getCut() (OwnedInteraction method), 138
getCut() (OwnedLocation method), 140
getCut() (OwnedMapsTo method), 143
getCut() (OwnedMeasurement method), 145
getCut() (OwnedModel method), 147
getCut() (OwnedModule method), 149
getCut() (OwnedModuleDefinition method), 152
getCut() (OwnedParticipation method), 154
getCut() (OwnedPlan method), 156
getCut() (OwnedSampleRoster method), 158
getCut() (OwnedSequence method), 161
getCut() (OwnedSequenceAnnotation method), 163
getCut() (OwnedSequenceConstraint method), 165
getCut() (OwnedTest method), 167
getCut() (OwnedUsage method), 170
getCut() (OwnedVariableComponent method), 172
getCut() (TextProperty method), 203
getCut() (URIProperty method), 208
getCut() (Document method), 68
getCut() (OwnedActivity method), 104
getCut() (OwnedAgent method), 106
getCut() (OwnedAnalysis method), 109
getCut() (OwnedAssociation method), 111
getCut() (OwnedAttachment method), 113
getCut() (OwnedBuild method), 115
getCut() (OwnedCollection method), 118
getCut() (OwnedCombinatorialDerivation method), 120
getCut() (OwnedComponent method), 122
getCut() (OwnedComponentDefinition method), 124
getCut() (OwnedDesign method), 127
getCut() (OwnedExperiment method), 129
getCut() (OwnedExperimentalData method), 131
getCut() (OwnedFunctionComponent method), 133
getCut() (OwnedImplementation method), 136

getCut() (OwnedFunctionalComponent method), 134
getCut() (OwnedImplementation method), 136
getCut() (NamedInteraction method), 138
getCut() (OwnedLocation method), 140
getCut() (OwnedModel method), 143
getCut() (OwnedMeasurement method), 145
getCut() (OwnedModule method), 147
getCut() (OwnedModuleDefinition method), 152
getCut() (OwnedParticipation method), 154
getCut() (OwnedPlan method), 156
getCut() (OwnedSampleRoster method), 159
getCut() (OwnedSequence method), 161
getCut() (OwnedSequenceAnnotation method), 163
getCut() (OwnedSequenceConstraint method), 165
getCut() (OwnedTest method), 168
getCut() (OwnedUsage method), 170
getCut() (OwnedVariableComponent method), 172
gDesign() (Document method), 69
gDownstreamComponent() (ComponentDefinition method), 56
gExperiment() (Document method), 69
gExperimentalData() (Document method), 69
gFileFormat() (in module sbol.libsbol), 214
getFirstComponent() (ComponentDefinition method), 56
gGenericLocation() (OwnedActivity method), 104
gGenericLocation() (OwnedAgent method), 107
gGenericLocation() (OwnedAnalysis method), 109
gGenericLocation() (OwnedAssociation method), 111
gGenericLocation() (OwnedAttachment method), 113
gGenericLocation() (OwnedBuild method), 116
gGenericLocation() (OwnedCollection method), 118
gGenericLocation() (OwnedCombinatorialDerivation method), 120
gGenericLocation() (OwnedComponent method), 122
gGenericLocation() (OwnedComponentDefinition method), 125
gGenericLocation() (OwnedDesign method), 127
gGenericLocation() (OwnedExperiment method), 129
gGenericLocation() (OwnedExperimentalData method), 132
gGenericLocation() (OwnedFunctionalComponent method), 134
gGenericLocation() (OwnedImplementation method), 136
gGenericLocation() (OwnedInteraction method), 138
gGenericLocation() (OwnedLocation method), 141
gGenericLocation() (OwnedMapsTo method), 143
gGenericLocation() (OwnedMeasurement method), 145
gGenericLocation() (OwnedModel method), 147
gGenericLocation() (OwnedModule method), 150
gGenericLocation() (OwnedModuleDefinition method), 152
gGenericLocation() (OwnedParticipation method), 154
gGenericLocation() (OwnedPlan method), 157
gGenericLocation() (OwnedSampleRoster method), 159
gGenericLocation() (OwnedSequence method), 161
gGenericLocation() (OwnedSequenceAnnotation method), 163
gGenericLocation() (OwnedSequenceConstraint method), 166
gGenericLocation() (OwnedTest method), 168
gGenericLocation() (OwnedUsage method), 170
gGenericLocation() (OwnedVariableComponent method), 172
gHomespace() (in module sbol.libsbol), 214
gImplementation() (Document method), 70
gInSequentialOrder() (ComponentDefinition method), 56
gLastComponent() (ComponentDefinition method), 57
gLowerBound() (ActivityProperty method), 34
gLowerBound() (AgentProperty method), 36
gLowerBound() (AnalysisProperty method), 39
gLowerBound() (AssociationProperty method), 41
gLowerBound() (AttachmentProperty method), 43
gLowerBound() (BuildProperty method), 45
gLowerBound() (CollectionProperty method), 47
gLowerBound() (CombinatorialDerivationProperty method), 50
gLowerBound() (ComponentDefinitionProperty method), 59
gLowerBound() (ComponentProperty method), 61
gLowerBound() (DesignProperty method), 65
gLowerBound() (ExperimentalDataProperty method), 75
gLowerBound() (ExperimentProperty method), 74
gLowerBound() (FunctionalComponentProperty
getLowerBound() (ImplementationProperty method), 85
getLowerBound() (InteractionProperty method), 88
getLowerBound() (LocationProperty method), 90
getLowerBound() (MapsToProperty method), 93
getLowerBound() (MeasurementProperty method), 94
getLowerBound() (ModuleProperty method), 97
getLowerBound() (ModelProperty method), 97
getLowerBound() (ModuleDefinitionProperty method), 101
getLowerBound() (ParticipationProperty method), 177
getLowerBound() (PlanProperty method), 179
getLowerBound() (SampleRosterProperty method), 186
getLowerBound() (SequenceAnnotationProperty method), 193
getLowerBound() (SequenceConstraintProperty method), 196
getLowerBound() (SequenceProperty method), 197
getLowerBound() (TestProperty method), 203
getLowerBound() (UsageProperty method), 210
getLowerBound() (VariableComponentProperty method), 213
getOwner() (ModuleProperty method), 102
getOwner() (ParticipationProperty method), 177
getOwner() (PlanProperty method), 179
getOwner() (SampleRosterProperty method), 186
getOwner() (SequenceAnnotationProperty method), 193
getOwner() (SequenceConstraintProperty method), 196
getOwner() (SequenceProperty method), 197
getOwner() (TestProperty method), 203
getOwner() (UsageProperty method), 210
getOwner() (VariableComponentProperty method), 213
getPlan() (Document method), 70
getPrimaryStructure() (ComponentDefinition method), 57
getProperties() (SBOLObject method), 184
getPropertyValue() (SBOLObject method), 184
getPropertyValues() (SBOLObject method), 184
getRange() (OwnedActivity method), 105
getRange() (OwnedAgent method), 107
getRange() (OwnedAnalysis method), 109
getRange() (OwnedAssociation method), 111
getRange() (OwnedAttachment method), 114
getRange() (OwnedBuild method), 116
getRange() (OwnedCollection method), 118
getRange() (OwnedCombinatorialDerivation method), 120
getRange() (OwnedComponent method), 123
getRange() (OwnedComponentDefinition method), 125
getRange() (OwnedDesign method), 127
getRange() (OwnedExperiment method), 130
getRange() (OwnedExperimentalData method), 132
getRange() (OwnedFunctionalComponent method), 134
getRange() (OwnedImplementation method), 136
getRange() (OwnedInteraction method), 139
getRange() (OwnedLocation method), 141
getRange() (OwnedMapsTo method), 143
getRange() (OwnedMeasurement method), 145
getRange() (OwnedModel method), 148
getRange() (OwnedModule method), 150
getRange() (OwnedModuleDefinition method), 152
getRange() (OwnedParticipation method), 155
getRange() (OwnedPlan method), 157
getRange() (OwnedSampleRoster method), 159
getRange() (OwnedSequence method), 161
getRange() (OwnedSequenceAnnotation method), 164
getRange() (OwnedSequenceConstraint method), 166
getRange() (OwnedTest method), 168
getRange() (OwnedUsage method), 170
getRange() (OwnedVariableComponent method), 173
getSampleRoster() (Document method), 70
getSequence() (Document method), 70
getTest() (Document method), 70
getTypeURI() (ActivityProperty method), 34
getTypeURI() (AgentProperty method), 36
getTypeURI() (AnalysisProperty method), 39
getTypeURI() (AssociationProperty method), 41
getTypeURI() (AttachmentProperty method), 43
getTypeURI() (BuildProperty method), 45
getTypeURI() (CollectionProperty method), 47
getTypeURI() (CombinatorialDerivationProperty method), 50
getTypeURI() (ComponentDefinitionProperty method), 59
getTypeURI() (ComponentProperty method), 61
getTypeURI() (DesignProperty method), 65
getTypeURI() (ExperimentalDataProperty method), 75
getTypeURI() (ExperimentProperty method), 74
getTypeURI() (FunctionalComponentProperty method), 79
getTypeURI() (ImplementationProperty method), 85
getTypeURI() (InteractionProperty method), 88
getTypeURI() (LocationProperty method), 90
getTypeURI() (MapsToProperty method), 93
getTypeURI() (MeasurementProperty method), 94
getTypeURI() (ModelProperty method), 97
getTypeURI() (ModuleDefinitionProperty method), 101
getTypeURI() (ModuleProperty method), 102
getTypeURI() (ParticipationProperty method), 177
getTypeURI() (PlanProperty method), 179
getTypeURI() (PartShop method), 174
getUpperBound() (ComponentProperty method), 61
getUpperBound() (DesignProperty method), 65
getUpperBound() (ExperimentalDataProperty method), 75
getUpperBound() (ExperimentProperty method), 74
getUpperBound() (InteractionProperty method), 88
getUpperBound() (LocationProperty method), 90
getUpperBound() (MapsToProperty method), 93
getUpperBound() (MeasurementProperty method), 94
getUpperBound() (ModelProperty method), 97
getUpperBound() (ModuleDefinitionProperty method), 101
getUpperBound() (ModuleProperty method), 102
getUpperBound() (ParticipationProperty method), 177
getUpperBound() (PlanProperty method), 179
getUpperBound() (SampleRosterProperty method), 186
getUpperBound() (SequenceAnnotationProperty method), 193
getUpperBound() (SequenceConstraintProperty method), 196
getUpperBound() (SequenceProperty method), 197
getUpperBound() (TestProperty method), 203
getUpperBound() (UsageProperty method), 210
getUpperBound() (VariableComponentProperty method), 213
getUpstreamComponent() (ComponentDefinition method), 57
getURL() (PartShop method), 174
hasDownstreamComponent() (ComponentDefinition method), 57
hasHomespace() (in module sbol.libsbol), 214
hasUpstreamComponent() (ComponentDefinition method), 57
hasUpstreamComponent() (in module sbol.libsbol), 214
Identifier (class in sbol.libsbol), 82
Implementation (class in sbol.libsbol), 83
ImplementationProperty (class in sbol.libsbol), 84
incrementMajor() (VersionProperty method), 213
incrementMinor() (VersionProperty method), 214
incrementPatch() (VersionProperty method), 214
initialize() (TopLevel method), 204
insertDownstream() (ComponentDefinition method), 57
insertUpstream() (ComponentDefinition method), 57
Interaction (class in sbol.libsbol), 86
InteractionProperty (class in sbol.libsbol), 88
IntProperty (class in sbol.libsbol), 85
is_alphanumeric_or_underscore() (in module sbol.libsbol), 214
is_not_alphanumeric_or_underscore() (in module sbol.libsbol), 214
isComplete() (ComponentDefinition method), 57
isMasked() (FunctionalComponent method), 78
isRegular() (ComponentDefinition method), 58

L
length() (Sequence method), 181
length() (SequenceProperty method), 181
length() (SequenceAnnotation method), 192
libsbol_rule_1() (in module sbol.libsbol), 214
libsbol_rule_10() (in module sbol.libsbol), 214
libsbol_rule_11() (in module sbol.libsbol), 214
libsbol_rule_12() (in module sbol.libsbol), 215
libsbol_rule_13() (in module sbol.libsbol), 215
libsbol_rule_14() (in module sbol.libsbol), 215
libsbol_rule_15() (in module sbol.libsbol), 215
libsbol_rule_16() (in module sbol.libsbol), 215
libsbol_rule_17() (in module sbol.libsbol), 215
libsbol_rule_18() (in module sbol.libsbol), 215
libsbol_rule_19() (in module sbol.libsbol), 215
libsbol_rule_2() (in module sbol.libsbol), 215
libsbol_rule_20() (in module sbol.libsbol), 215
libsbol_rule_21() (in module sbol.libsbol), 215
libsbol_rule_22() (in module sbol.libsbol), 215
libsbol_rule_24() (in module sbol.libsbol), 215
libsbol_rule_3() (in module sbol.libsbol), 215
libsbol_rule_4() (in module sbol.libsbol), 215
libsbol_rule_5() (in module sbol.libsbol), 215
libsbol_rule_6() (in module sbol.libsbol), 215
libsbol_rule_7() (in module sbol.libsbol), 215
libsbol_rule_8() (in module sbol.libsbol), 215
libsbol_rule_9() (in module sbol.libsbol), 215
linearize() (ComponentDefinition method), 58
Location (class in sbol.libsbol), 89
LocationProperty (class in sbol.libsbol), 90
login() (PartShop method), 174

M
major() (VersionProperty method), 214
MapsTo (class in sbol.libsbol), 91
MapsToProperty (class in sbol.libsbol), 92
mask() (FunctionalComponent method), 78
Measurement (class in sbol.libsbol), 93
MeasurementProperty (class in sbol.libsbol), 94
minor() (VersionProperty method), 214
Model (class in sbol.libsbol), 95
ModelProperty (class in sbol.libsbol), 96
Module (class in sbol.libsbol), 97
ModuleDefinition (class in sbol.libsbol), 98
ModuleDefinitionProperty (class in sbol.libsbol), 101
ModuleProperty (class in sbol.libsbol), 102

O
overlaps() (Range method), 181
overlaps() (SequenceAnnotation method), 193
override() (FunctionalComponent method), 78
override() (ModuleDefinition method), 100
OwnedActivity (class in sbol.libsbol), 103
OwnedAgent (class in sbol.libsbol), 105
OwnedAnalysis (class in sbol.libsbol), 107
OwnedAssociation (class in sbol.libsbol), 110
OwnedAttachment (class in sbol.libsbol), 112
OwnedBuild (class in sbol.libsbol), 114
OwnedCollection (class in sbol.libsbol), 116
OwnedCombinatorialDerivation (class in sbol.libsbol), 119
OwnedComponent (class in sbol.libsbol), 121
OwnedComponentDefinition (class in sbol.libsbol), 123
OwnedDesign (class in sbol.libsbol), 125
OwnedExperiment (class in sbol.libsbol), 128
OwnedExperimentalData (class in sbol.libsbol), 130
OwnedFunctionalComponent (class in sbol.libsbol), 132
OwnedImplementation (class in sbol.libsbol), 135
OwnedInteraction (class in sbol.libsbol), 137
OwnedLocation (class in sbol.libsbol), 139
OwnedMapsTo (class in sbol.libsbol), 141
OwnedMeasurement (class in sbol.libsbol), 144
OwnedModel (class in sbol.libsbol), 146
OwnedModule (class in sbol.libsbol), 148
OwnedModuleDefinition (class in sbol.libsbol), 150
OwnedParticipation (class in sbol.libsbol), 153
OwnedPlan (class in sbol.libsbol), 155
OwnedSampleRoster (class in sbol.libsbol), 157
OwnedSequence (class in sbol.libsbol), 160
OwnedSequenceAnnotation (class in sbol.libsbol), 162
OwnedSequenceConstraint (class in sbol.libsbol), 164
OwnedTest (class in sbol.libsbol), 166
OwnedUsage (class in sbol.libsbol), 169
OwnedVariableComponent (class in sbol.libsbol), 171

P
participate() (ComponentDefinition method), 58
Participation (class in sbol.libsbol), 175
ParticipationProperty (class in sbol.libsbol), 177
PartShop (class in sbol.libsbol), 173
patch() (VersionProperty method), 214
Plan (class in sbol.libsbol), 178
PlanProperty (class in sbol.libsbol), 179
precedes() (Range method), 182
precedes() (SequenceAnnotation method), 193
pull() (PartShop method), 174
pullCollection() (PartShop method), 174
pullComponentDefinition() (PartShop method), 174
pullSequence() (PartShop method), 174
Q
query_repository() (Document method), 71

R
Range (class in sbol.libsbol), 180
read() (Document method), 71
readString() (Document method), 71
ReferencedObject (class in sbol.libsbol), 182
remove() (ActivityProperty method), 34
remove() (AgentProperty method), 36
remove() (AnalysisProperty method), 39
remove() (AssociationProperty method), 41
remove() (AttachmentProperty method), 43
remove() (BuildProperty method), 45
remove() (CollectionProperty method), 47
remove() (CombinatorialDerivationProperty method), 50
remove() (ComponentDefinitionProperty method), 59
remove() (ComponentProperty method), 61
remove() (DesignProperty method), 65
remove() (ExperimentalDataProperty method), 75
remove() (ExperimentalProperty method), 74
remove() (FunctionalComponentProperty method), 79
remove() (ImplementationProperty method), 85
remove() (InteractionProperty method), 88
remove() (LocationProperty method), 90
remove() (MapsToProperty method), 93
remove() (MeasurementProperty method), 94
remove() (ModelProperty method), 97
remove() (ModuleDefinitionProperty method), 101
remove() (ModuleProperty method), 102
remove() (OwnedActivity method), 105
remove() (OwnedAgent method), 107
remove() (OwnedAnalysis method), 109
remove() (OwnedAssociation method), 112
remove() (OwnedAttachment method), 114
remove() (OwnedBuild method), 116
remove() (OwnedCollection method), 118
remove() (OwnedCombinatorialDerivation method), 121
remove() (OwnedComponent method), 123
remove() (OwnedComponentDefinition method), 125
remove() (OwnedDesign method), 128
remove() (OwnedExperiment method), 130
remove() (OwnedExperimentalData method), 132
remove() (OwnedFunctionalComponent method), 134
remove() (OwnedImplementation method), 137
remove() (OwnedInteraction method), 139
remove() (OwnedLocation method), 141
remove() (OwnedMapsTo method), 143
remove() (OwnedMeasurement method), 146
remove() (OwnedModel method), 148
remove() (OwnedModule method), 150
remove() (OwnedModuleDefinition method), 153
remove() (OwnedParticipation method), 155
remove() (OwnedPlan method), 157
remove() (OwnedSampleRoster method), 159
remove() (OwnedSequence method), 162
remove() (OwnedSequenceAnnotation method), 164
remove() (OwnedSequenceConstraint method), 166
remove() (OwnedTest method), 168
remove() (OwnedUsage method), 171
remove() (OwnedVariableComponent method), 173
remove() (ParticipationProperty method), 177
remove() (PlanProperty method), 179
remove() (SampleRosterProperty method), 186
remove() (SequenceAnnotationProperty method), 193
remove() (SequenceConstraintProperty method), 196
remove() (SequenceProperty method), 197
remove() (TestProperty method), 203
remove() (UsageProperty method), 210
remove() (VariableComponentProperty method), 213
reportAmbiguity() (Analysis method), 38
reportCoverage() (Analysis method), 38
reportError() (Analysis method), 38
reportIdentity() (Analysis method), 38
request_comparison() (Document method), 71
request_validation() (Document method), 71

S
SampleRoster (class in sbol.libsbol), 185
SampleRosterProperty (class in sbol.libsbol), 186
sbol.libsbol (module), 31
sbol_rule_10202 () (in module sbol.libsbol), 216
sbol_rule_10204 () (in module sbol.libsbol), 216
SBOLObject (class in sbol.libsbol), 182
sbolRule10101 () (in module sbol.libsbol), 215
sbolRule10102 () (in module sbol.libsbol), 216
search() (PartShop method), 174
search_metadata() (Document method), 71
searchCount() (PartShop method), 174
SearchQuery (class in sbol.libsbol), 187
SearchResponse (class in sbol.libsbol), 188
searchRootCollections() (PartShop method), 175
searchSubCollections() (PartShop method), 175
Sequence (class in sbol.libsbol), 189
SequenceAnnotation (class in sbol.libsbol), 191
SequenceAnnotationProperty (class in sbol.libsbol), 193
SequenceConstraint (class in sbol.libsbol), 194
SequenceConstraintProperty (class in sbol.libsbol), 195
SequenceProperty (class in sbol.libsbol), 196
set () (ActivityProperty method), 34
set () (AliasedOwnedFunctionalComponent method), 37
set () (AnalysisProperty method), 39
set () (AssociationProperty method), 41
set () (AttachmentProperty method), 43
set () (BuildProperty method), 45
set () (CollectionProperty method), 47
set () (CombinatorialDerivationProperty method), 50
set () (ComponentDefinitionProperty method), 59
set () (ComponentProperty method), 61
set () (DesignProperty method), 65
set () (ExperimentalDataProperty method), 75
set () (ExperimentProperty method), 74
set () (FunctionalComponentProperty method), 79
set () (ImplementationProperty method), 85
set () (InteractionProperty method), 88
set () (LocationProperty method), 91
set () (MapsToProperty method), 93
set () (MeasurementProperty method), 94
set () (ModelProperty method), 97
set () (ModuleDefinitionProperty method), 101
set () (ModuleProperty method), 103
set () (OwnedActivity method), 105
set () (OwnedAgent method), 107
set () (OwnedAnalysis method), 109
set () (OwnedAssociation method), 112
set () (OwnedAttachment method), 114
set () (OwnedBuild method), 116
set () (OwnedCollection method), 118
set () (OwnedCombinatorialDerivation method), 121
set () (OwnedComponent method), 123
set () (OwnedComponentDefinition method), 125
set () (OwnedDesign method), 128
set () (OwnedExperiment method), 130
set () (OwnedExperimentalData method), 132
set () (OwnedFunctionalComponent method), 134
set () (OwnedImplementation method), 137
set () (OwnedInteraction method), 139
set () (OwnedLocation method), 141
set () (OwnedMapsTo method), 143
set () (OwnedMeasurement method), 146
set () (OwnedModel method), 148
set () (OwnedModule method), 150
set () (OwnedModuleDefinition method), 153
set () (OwnedParticipation method), 155
set () (OwnedPlan method), 157
set () (OwnedSampleRoster method), 159
set () (OwnedSequence method), 162
set () (OwnedSequenceAnnotation method), 164
set () (OwnedSequenceConstraint method), 166
set () (OwnedTest method), 168
set () (OwnedUsage method), 171
set () (OwnedVariableComponent method), 173
set () (ParticipationProperty method), 178
set () (PlanProperty method), 180
set () (ReferencedObject method), 182
set () (SampleRosterProperty method), 186
set () (SequenceAnnotationProperty method), 193
set () (SequenceConstraintProperty method), 196
set () (SequenceProperty method), 197
set () (TestProperty method), 203
set () (UsageProperty method), 210
set () (VariableComponentProperty method), 213
setAnnotation() (SBOLObject method), 184
setFileFormat() (in module sbol.libsbol), 216
setHomespace() (in module sbol.libsbol), 216
setInput() (ModuleDefinition method), 100
setOption() (Config static method), 62
setOutput() (ModuleDefinition method), 101
setPropertyValue() (SBOLObject method), 184
setReference() (ReferencedObject method), 182
SmallMoleculeActivationInteraction (class in sbol.libsbol), 197
SmallMoleculeInhibitionInteraction (class in sbol.libsbol), 199
stampTime() (DateTimeProperty method), 63
submit() (PartShop method), 175
summary() (Document method), 71
synthesize() (Sequence method), 191
T
Test (class in sbol.libsbol), 201
TestProperty (class in sbol.libsbol), 202
testRoundTrip() (in module sbol.libsbol), 216
testSBOL() (in module sbol.libsbol), 216
TextProperty (class in sbol.libsbol), 203
TopLevel (class in sbol.libsbol), 203
TranscriptionalActivationInteraction (class in sbol.libsbol), 204
TranscriptionalRepressionInteraction (class in sbol.libsbol), 206
U
update_uri() (SBOLObject method), 185
validate() (ComponentDefinition method), 58
URIProperty (class in sbol.libsbol), 208
Usage (class in sbol.libsbol), 208
UsageProperty (class in sbol.libsbol), 209
validate() (ActivityProperty method), 34
validate() (AgentProperty method), 36
validate() (AnalysisProperty method), 39
validate() (AssociationProperty method), 41
validate() (AttachmentProperty method), 43
validate() (BuildProperty method), 45
validate() (CollectionProperty method), 47
validate() (CombinatorialDerivationProperty method), 50
validate() (ComponentDefinitionProperty method), 59
validate() (ComponentProperty method), 61
validate() (DesignProperty method), 65
validate() (Document method), 71
validate() (ExperimentalDataProperty method), 76
validate() (ExperimentProperty method), 79
validate() (FunctionalComponentProperty method), 79
validate() (ImplementationProperty method), 85
validate() (InteractionProperty method), 89
validate() (LocationProperty method), 91
validate() (MapsToProperty method), 93
validate() (MeasurementProperty method), 95
validate() (ModuleProperty method), 97
validate() (ModuleDefinitionProperty method), 102
validate() (ModuleProperty method), 103
validate() (ParticipationProperty method), 178
validate() (PlanProperty method), 180
validate() (SampleRosterProperty method), 187
validate() (SequenceAnnotationProperty method), 194
validate() (SequenceConstraintProperty method), 196
validate() (SequenceProperty method), 197
validate() (TestProperty method), 203
validate() (UsageProperty method), 210
validate() (VariableComponentProperty method), 213
VariableComponent (class in sbol.libsbol), 210
VariableComponentProperty (class in sbol.libsbol), 212
verifyTarget() (Analysis method), 38
VersionProperty (class in sbol.libsbol), 213
write() (ActivityProperty method), 34
write() (AgentProperty method), 36

W
write() (ActivityProperty method), 34
write() (AgentProperty method), 36