**Madsen C, McLaughlin JA, Misirli G, Pocock P, Flanagan K, Hallinan J, Wipat A.**

**The SBOL Stack: A Platform for Storing, Publishing, and Sharing Synthetic Biology Designs.**

***ACS Synthetic Biology* 2016, 5(6), 487-497**

ACS SyntheticBiology

Research Article

pubs.acs.org/synthbio

# The SBOL Stack: A Platform for Storing, Publishing, and Sharing Synthetic Biology Designs

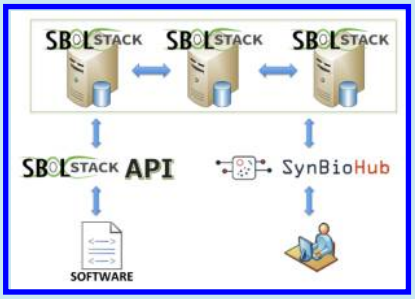
Curtis Madsen,[†,‡,∥] James Alastair McLaughlin,[†,∥] Göksel Mısırlı,[†] Matthew Pocock,[†,§] Keith Flanagan,[†] Jennifer Hallinan,[†,⊥] and Anil Wipat*[,†]

[†]School of Computing Science, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.

[‡]Department of Electrical & Computer Engineering, Boston University, Boston, Massachusetts 02215, United States

[§]Turing Ate My Hamster, LTD, Newcastle upon Tyne NE27 0RT, U.K.

**ABSTRACT:** Recently, synthetic biologists have developed the *Synthetic Biology Open Language* (SBOL), a data exchange standard for descriptions of genetic parts, devices, modules, and systems. The goals of this standard are to allow scientists to exchange designs of biological parts and systems, to facilitate the storage of genetic designs in repositories, and to facilitate the description of genetic designs in publications. In order to achieve these goals, the development of an infrastructure to store, retrieve, and exchange SBOL data is necessary. To address this problem, we have developed the SBOL Stack, a *Resource Description Framework* (RDF) database specifically designed for the storage, integration, and publication of SBOL data. This database allows users to define a library of synthetic parts and designs as a service, to share SBOL data with collaborators, and to store designs of biological systems locally. The database also allows external data sources to be integrated by mapping them to the SBOL data model. The SBOL Stack includes two Web interfaces: the SBOL Stack API and SynBioHub. While the former is designed for developers, the latter allows users to upload new SBOL biological designs, download SBOL documents, search by keyword, and visualize SBOL data. Since the SBOL Stack is based on semantic Web technology, the inherent distributed querying functionality of RDF databases can be used to allow different SBOL stack databases to be queried simultaneously, and therefore, data can be shared between different institutes, centers, or other users.




**KEYWORDS:** *Synthetic Biology Open Language (SBOL), synthetic biology, RDF, triplestore, database, data standards*

Synthetic biology combines ideas from biology and engineering and has the goal of designing and building novel, useful biological systems. However, using the extensive amount of existing biological data for the purposes of designing new, synthetic organisms can be a daunting task. This challenge is magnified by the fact that efforts are usually carried out in individual laboratories, by teams in different geographic locations. Moreover, the interests of researchers can vary greatly, and biological data relevant to the design of genetic circuits is often not exchanged. In order to enable this exchange and to facilitate the scaling of designs, computer applications require standards, repositories, and APIs to enable researchers to communicate the designs of genetic parts and circuits in a uniform manner. Systems are also required that allow these designs to be stored and shared.

The *Synthetic Biology Open Language* (SBOL)[1−3] is an emerging standard in synthetic biology. The goals of this standard are to allow researchers to exchange designs of biological parts and systems, to send and receive genetic designs, to facilitate the storage of genetic designs in repositories, and to include genetic designs in publications. Although the initial version of SBOL focused upon exchanging genetic descriptions of parts at the DNA level, SBOL version 2 supports other types of genetic circuit components including proteins, RNAs, and small molecules, in addition to other elements which help to specify design constraints between these components. For example, structural constraints can be used to understand the relative order of components in a final DNA construct, and molecular interactions between these components can also be described. SBOL version 2 also allows for the hierarchical specification of genetic circuit designs *via* the definition of inputs and outputs at the structural and functional level.

SBOL can be used to enable workflows involving different tools and repositories from multiple organizations. For instance, SBOL models can be constructed using *computer-aided design* (CAD) tools such as SynBad,[4] iBioSim,[5] and TinkerCell.[6] Sequence alignment can be performed on an SBOL document using tools such as Vector NTI Express Designer.[7] SBOL designs can be stored and retrieved from repositories such as the Flowers Virtual Parts Repository[8] and JBEI-ICE[9] either during construction or after the design has been completed. For example, six independent groups recently collaborated on the design of a set of genetic toggle switches using several SBOL enabled tools.[10]

**Special Issue:** IWBDA 2015

**Received:** October 21, 2015
**Published:** June 7, 2016

DOI: 10.1021/acssynbio.5b00210
*ACS Synth. Biol.* 2016, 5, 487−497

SBOL documents are serialized using the RDF/XML format[11] and can be stored in standard RDF repositories as a set of triples. RDF is a labeled, directed, graph-based data format for representing information in Web environments. Figure 1 shows a graphical example of the RDF definition of a

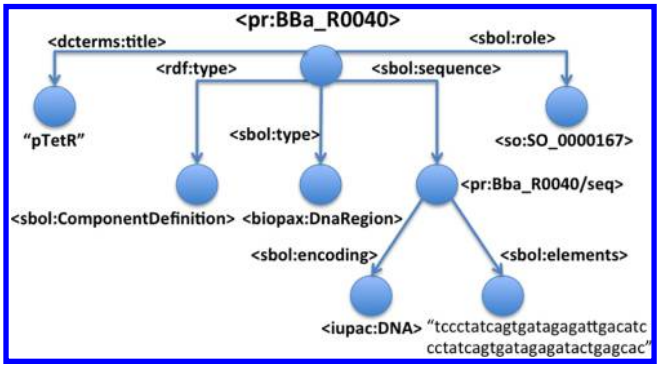



**Figure 1.** A graphical visualization of an SBOL version 2 `ComponentDefinition` and associated triples. Each edge, together with the two connected nodes, represents a triple. For simplicity, URIs are represented in angle brackets using qualifiers, such as `so` for the Sequence Ontology.

TetR repressible promoter using SBOL version 2. In this figure, each node and edge is labeled with either a *Uniform Resource Identifier* (URI) that can be used to link to more data, or with a literal data item. The promoter is represented with an SBOL `ComponentDefinition` entity and its role is set to the promoter term from the Sequence Ontology.[12] This entity is then associated with a `Sequence` entity in order to capture the nucleotide sequence of the promoter. This graph representation is ideal to execute complex queries using languages designed for navigating RDF graphs, such as *SPARQL Protocol and RDF Query Language* (SPARQL).[13] Using SPARQL with SBOL RDF enables the expression of complex queries, such as searching for all regulatory interactions for a given coding site or searching for a particular feature among multiple parts. Therefore, the use of RDF enables the application of a range of standard Semantic Web technology for computational synthetic biology applications. Given the rich promise of functionality afforded by the representation of SBOL in RDF form, there is an overarching need to provide a system that will allow SBOL version 2 to be stored, shared, and queried.

Previously, Galdzicki and co-workers[14] demonstrated the utility of SBOL version 1 by converting data from the iGEM Registry of Standard Biological Parts (http://parts.igem.org/Main_Page)[15] into SBOL version 1 and making the data available through a *triplestore*, a type of database specifically developed for storing and querying RDF. However, this system is currently not under further development.

More generically, several systems for storing and sharing synthetic biology designs have been described. Most notable are the Joint BioEnergy Institute's Inventory of Composable Elements (JBEI-ICE)[9] and the iGEM Registry of Standard Biological Parts. Many other software tools also exist that offer the ability to store genetic designs, both commercial and noncommercial (*e.g.*, Plasmid repository, VectorNTI, and Addgene to name a few). All of these systems, except JBEI-ICE, do not provide open-source repositories that promote data sharing by allowing users to install their own systems that can be linked. JBEI-ICE is an open source registry that allows information about biological parts to be managed and shared. The repository is accessible *via* both a Web browser and through a Simple Object Access Protocol (SOAP) based Web application programming interface. ICE also introduced the concept of a Web of registries, providing support for ICE repositories to be connected to allow distributed and interconnected use. The system also supports the import and export of SBOL version 1 and SBOL version 2 files that describe the composition of parts at the genetic level. The JBEI-ICE system as a whole also provides an open source suite of tools to allow sequence annotation, manipulation, and analysis. However, the integration of ICE repositories is carried out using Universally Unique Indentifiers (UUIDs) which means that, currently, parts can only be shared between other ICE repositories. Furthermore, the data format used for representing the parts for integration is specific to ICE and not based on SBOL or a standardized format. ICE data is currently confined to parts metadata and the information about the genetic features in the parts. SBOL version 2 provides information about proteins, small molecules, RNA, and parts interactions which are not currently captured in ICE.

In this work, we describe the SBOL Stack, a platform to allow synthetic biology designs that have been represented in SBOL version 1 or SBOL version 2 to be computationally stored, retrieved, exchanged, and ultimately, published. The SBOL Stack is based on a Sesame RDF triplestore (http://rdf4j.org)[16] specifically designed for publishing a library of synthetic parts and designs as a service, for storing designs of biological systems locally, and for facilitating the sharing and integration of SBOL with collaborators using standardized Semantic Web technologies.

Unlike many previously developed repositories, the SBOL Stack is designed to automatically integrate SBOL data from other SBOL Stack installations as a result of the standardized, and therefore shared, semantics of SBOL version 2. The use of standardized, shared semantics also offers the opportunity to integrate a range of alternative data sources into the same repository with the potential to enrich SBOL designs and inform the design process. This integration process is enabled by the SPARQL query language that can already express queries across diverse data sources, whether the data is stored natively as RDF or exposed as RDF using middleware. The specification for executing queries distributed over different SPARQL end points has already been standardized by the W3C consortium (https://www.w3.org/TR/sparql11-federated-query). The SPARQL language also already includes functionality for merging the results of queries across multiple triplestore databases distributed across the Web. This inherent federated querying can be exploited and used to enable multiple SBOL Stacks to be integrated at query time, such that queries executed on one SBOL Stack will be automatically shared between a set of specified remote SBOL Stack instances (Figure 2).

The SBOL Stack is primarily designed to be part of the infrastructure available to synthetic biology tool-builders who need to implement systems that require the storage of SBOL and would also like to integrate further data sources. In order to aid developers we have developed a range of Web interfaces and developer tools, enabling developers to build their own software to read and write data available in the SBOL Stack. These tools include the SBOL Stack API, which provides a RESTful interface enabling programmatic access to SBOL Stack instances and allows data to be directly read, written, and


DOI: 10.1021/acssynbio.5b00210
ACS Synth. Biol. 2016, 5, 487−497

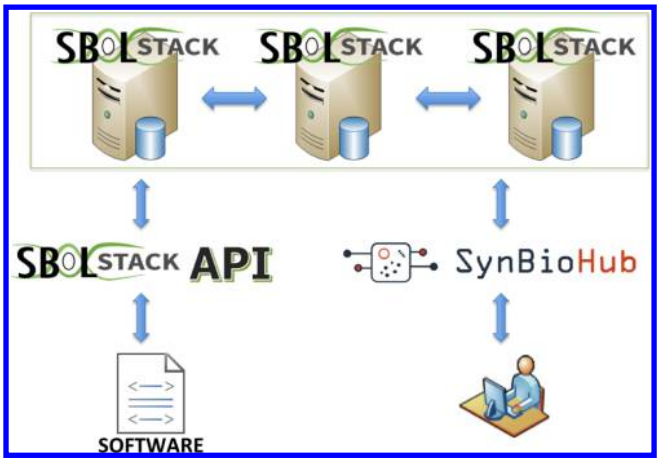



**Figure 2.** Synthetic biology designs can be browsed programmatically using the SBOL Stack API and client libraries or manually using SynBioHub. SPARQL queries can be used to facilitate data federation at the layer of the SBOL Stack. A SynBioHub instance is therefore able to query a collection of interconnected SBOL Stack instances simultaneously.

queried by other external software tools. Full, external, programmatic access for reading and writing SBOL version 2 synthetic biology designs is also a unique feature of the system. We also include Amazon cloud computing images and Docker container versions of the system to allow developers to install their own, local, SBOL Stack data stores (see the Methods section).

We have created a sample instance of the SBOL Stack that provides access to the complete iGEM Registry of Standard Biological Parts in SBOL version 2 format. This instance also includes semantically enriched integrated data from seven data sources describing *Bacillus subtilis*,[17] coupled with the SyBiOnt ontology[18] dealing with genetic features, gene products and their annotations, gene regulatory networks, metabolic pathways, and other useful information. The SyBiOnt ontology provides a useful framework for developers to integrate further biological data sets with SBOL version 2.

The SBOL Stack is primarily aimed at developers. In addition to the SBOL Stack we have also developed a simple Web-based tool, SynBioHub, primarily to demonstrate how the SBOL Stack can be used as a repository for SBOL. However, SynBioHub also provides a useful database for the everyday user wishing to store and visualize SBOL files. While SynBioHub does not facilitate the creation of SBOL files itself, it provides a useful repository and front end for storing, searching, and retrieving SBOL version 2 in the SBOL Stack. A number of tools for the creation of SBOL already exist, and it is envisaged that these will connect to the SBOL Stack directly using the tools and API we provide.

## 1. RESULTS

The system presented here allows the storage and querying of SBOL version 2. The system currently comprises two open source software tools. First, the SBOL Stack, which is a database for storing SBOL version 2 in a decomposed RDF format, where the individual data components are amenable to integration with other data sets. Second, SynBioHub, a Web based tool to allow the retrieval, visualization, and deposition of designs for engineered biological systems from the SBOL Stack in SBOL version 2 format. While the SBOL Stack is intended to be used by a synthetic biology tool builder, SynBioHub is aimed at an end user wishing to upload designs to the SBOL Stack database for storage and sharing and to browse and download existing designs that are already in the SBOL Stack.

**1.1. SBOL Stack Design, Architecture and Implementation.** The SBOL Stack platform is composed of a series of abstractions (Figure 3). At the core, it relies on a Sesame RDF

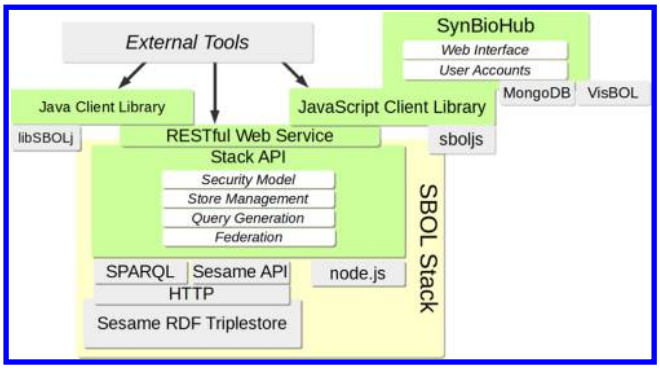



**Figure 3.** Architecture of the SBOL Stack. The Stack API provides an abstraction over the Sesame RDF triplestore specifically for the purpose of storing and searching SBOL designs. The API can then be accessed by external tools directly or through the client libraries available for Java and JavaScript. SynBioHub is an example of an external tool built upon the SBOL Stack and provides a further abstraction of dynamic user account management and the ability to upload and share designs through a Web interface.

triplestore. The communication with the triplestore is through HTTP and SPARQL, where possible, and the use of Sesame specific APIs is minimal; the Sesame API is used only to create new triplestores. Therefore, the SBOL Stack can easily be adapted to work with other triplestores in the future.

The triplestore is completely encapsulated by the SBOL Stack API, a RESTful Web service built on node.js.[19] While the RESTful interface alone makes the Stack accessible from any programming language or environment with HTTP client functionality, the Web service is also encapsulated by client libraries available for Java and JavaScript, in order to facilitate integration of the Stack with existing software on the client side. SynBioHub is an example of software built using the JavaScript client library, providing a user-friendly Web interface by combining the SBOL Stack with a MongoDB database for user accounts and the VisBOL[20] visualization library for producing diagrams of part composition.

The SBOL Stack implements a security model by which users are identified by a username and password. Each user then may be assigned individual permissions, such as the ability to create new stores, upload data, perform SPARQL queries, and search using Stack API calls. Additionally, a set of permissions can be allocated for anonymous users, enabling an instance of the Stack to provide varying degrees of public access.

This security model makes the SBOL Stack suitable for a range of use cases. For example, the SBOL Stack can be used to provide a publicly accessible data set where anyone can query the data, but only authenticated users can modify it.

**1.2. SBOL Stack API and Client Library.** The SBOL Stack API can be integrated with software using either the RESTful API or the client libraries available for Java and JavaScript. The API encapsulates a backend triplestore with a "black box" approach, abstracting away the details of setting up triplestores


DOI: 10.1021/acssynbio.5b00210
ACS Synth. Biol. 2016, 5, 487−497

and generating SPARQL queries. The API is lightweight and HTTP-based. Standard GET and POST requests are used to submit queries, and the results are returned to users in the SBOL version 2 format.

Using the API, structural information such as DNA sequences or functional information such as protein−protein interactions about biological system designs can be retrieved. SBOL version 2 allows detailed information for biological components, molecular interactions, and different types of biological constraints to be captured. Several SBOL entities are used to capture different elements of a design. Some of these entities are called top-level entities and are used as containers to embed all the relevant information. At the structural level, a `ComponentDefinition` entity can be used to represent the biological components such as DNA, proteins, and their composition using sub components. `ComponentDefinition` entities can be associated with sequence information such as nucleotides or amino acids. `ModuleDefinition` entities, on the other hand, are used to capture functional information about a particular design which may have sub designs. Modules can be defined with inputs and outputs in order to join together with other `ModuleDefinition` entities and to scale up target designs. The SBOL Stack API allows both `ComponentDefinition` and `ModuleDefinition` entities to be retrieved as complete SBOL documents containing both the requested entity and any other referenced entities.

`ComponentDefinition` entities, for example, can be simply searched for using their names. The search functionality provides different options, such as whether to search for an exact match or whether to use a wildcard search. URIs uniquely identifying these entities can also be used to retrieve particular `ComponentDefinition` entities. It is also possible to submit an `SBOLDocument` containing a `ComponentDefinition` as a template. The SBOL Stack API then generates a SPARQL query and returns an `SBOLDocument` containing every matching `ComponentDefinition` entity. As the amount of data in the SBOL Stack increases, it may become challenging to retrieve large documents. Moreover, users may wish to gain information about whether particular types of components exist or not. To overcome these issues, the API provides methods to return counts of SBOL entities in the SBOL Stack. Search results can also be retrieved using an offset, indicating the start index, and number of entities to retrieve, in order to return a subset of results. This approach allows users to iterate through data as necessary.

Though SBOL is well-suited as a rich format to describe components, its verbosity is not always desirable when describing a very large list of parts. For example, when searching an entire genome for coding sites, the overhead of representing each gene as a `ComponentDefinition` can quickly produce a very large result. For this reason, the SBOL Stack also provides an end point to search metadata, returning only the name, description and URI of matching `ComponentDefinition` entities as a JavaScript Object Notation (JSON)[21] array. The JSON format is intentionally lightweight with minimal markup, which makes the metadata end point useful for returning a large list of parts. As the URI is provided for each `ComponentDefinition` in the JSON result, the API can later be used to retrieve the complete SBOL document.

Table 1 shows some of the SBOL Stack end points. Although these end points can be accessed directly, we have developed Java and JavaScript client APIs, which provide methods for each of these end points for easier programmatic access. For example, the Java client library uses object representations for SBOL documents and other related entities by embedding the libSBOLj library[22] (Figure 4). Detailed, specific examples of querying using the Stack API are given below in Section 1.3.

**Table 1. A Description of the SBOL Stack API End Points**

| | |
|---|---|
| `POST/store/create` | Create a new store in the Stack |
| `POST/` | Upload new data to the Stack |
| `GET/sparql` | Query the Stack using SPARQL |
| `POST/sparql` | As `GET/sparql`, but the query is specified as a POST parameter |
| `GET/component/count` | Returns the total number of `ComponentDefinition` entities in the Stack |
| `GET/component/search/sbol` | Search the stack for `ComponentDefinition` entities, returning an SBOL document |
| `GET/component/search/metadata` | Search the stack for `ComponentDefinition` entities, returning metadata as JSON |
| `GET/component/search/template` | Search the stack for `ComponentDefinition` entities using an SBOL document as the query |
| `GET/module/count` | Returns the total number of `ModuleDefinition` entities in the Stack |
| `GET/module/search/sbol` | Search the stack for `ModuleDefinition` entities, returning an SBOL document |
| `GET/module/search/metadata` | Search the stack for `ModuleDefinition` entities, returning metadata as JSON |
| `POST/module/search/template` | Search the stack for `ModuleDefinition` entities using an SBOL document as the query |

**1.3. SBOL Stack Data Integration, Semantics, and the Example Data Set.** In addition to storing SBOL, the SBOL Stack can act as a data warehouse for synthetic biology designs allowing for the integration of multiple resources to aid in the design process. In this approach, data from external resources are converted either into SBOL version 2 format or another compatible semantic model and uploaded into an SBOL Stack instance. Since SBOL is RDF-based and the data are stored in triplestores without any further transformation, data can be aggregated using standard URIs and predicates. For example, uploading the definition of a promoter component and subsequently uploading custom annotations about the same promoter would merge all of the data into the same graph.

As an example, we built an instance of the SBOL Stack (see the Methods Section) from several data sources. This instance provides access to multiple data sets: seven integrated data sources relating to *Bacillus subtilis*;[17] data from the iGEM Registry of Standard Biological Parts (2016 data set); features from the complete *Escherichia coli* K12 substr. MG1655 genome from GenBank;[23] and a collection of SBOL version 2 examples from recent publications.[22,24,25] Data available in the provided instance of the SBOL Stack gives an example of how it is possible to include other custom ontologies in the SBOL Stack, providing they can be expressed in RDF. For instance, the SyBiOnt ontology[18] was used to mediate the integration of the *Bacillus subtilis* data sets and to enable integration with the SBOL version 2 data model to include information describing how genetic features, gene products, and transcription factors interact to form gene regulatory networks and metabolic pathways.

The SBOL Stack instance providing SBOL version 2 access to the iGEM Registry of Standard Biological Parts could be valuable to the synthetic biology community, since exposing


DOI: 10.1021/acssynbio.5b00210
ACS Synth. Biol. 2016, 5, 487−497

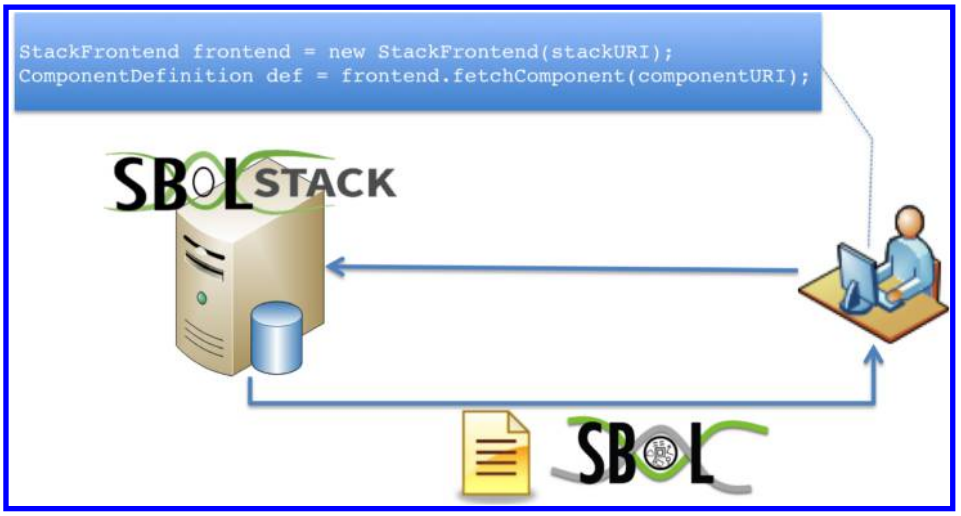



**Figure 4.** SBOL Stack can easily be queried programmatically using the client APIs. This example of the Java client API shows a `ComponentDefinition` entity identified by URI being retrieved from the SBOL Stack.

```
template.createComponentDefinition("constitutive_promoters",
    ComponentDefinition.DNA);
template.addRole(new URI("http://identifiers.org/so/SO:0000167"));
SBOLDocument results = frontend.searchComponents(template, 0, 1000);
```

**Figure 5.** Retrieving all promoters. A template `ComponentDefinition` is created with the `role` property set to the promoter term from SO. As a result, the first 1000 `ComponentDefinitions` that match the template, in this case promoter definitions, are returned. This snippet of code is written in Java for use with the Java API.

```
template.addRole(new URI("http://parts.igem.org/category/promoter"));
template.addRole(new
    URI("http://parts.igem.org/category/regulation/constitutive"));
SBOLDocument results = frontend.searchComponents(template, 0, 1000);
```

**Figure 6.** Retrieving all promoters. A template `ComponentDefinition` is created with the role property set to the promoter term from SO. As a result, the first 1000 `ComponentDefinitions` that match the template, in this case promoter definitions, are returned. This snippet of code is written in Java for use with the Java API.

```
PREFIX sbol: <http://sbols.org/v2#>
PREFIX ncbi: <http://www.ncbi.nlm.nih.gov#>
  SELECT ?ComponentDefinition WHERE {
      ?ComponentDefinition a sbol:ComponentDefinition;
        ncbi:taxid <http://identifiers.org/taxonomy/1423> .
  }
```

**Figure 7.** Retrieving all promoters. A template `ComponentDefinition` is created with the role property set to the promoter term from SO. As a result, the first 1000 `ComponentDefinitions` that match the template, in this case promoter definitions, are returned.

this data set allows users to immediately query and download BioBricks in the SBOL version 2 format. We have taken care to preserve links back to the iGEM registry including links to the part characterization data in the experience section of the registry. These parts can be used in SBOL workflows to design, build, and exchange new systems which are enriched with metadata integrated from a range of tools and data sets. Because the SBOL Stack is built upon an RDF graph, data is automatically integrated and queried alongside other data contained within the SBOL Stack. This integration can potentially provide a more enriched result than which a user would receive if the part was retrieved directly from the iGEM Registry of Standard Biological Parts or another repository without a graph-based store.

The provided SBOL Stack instance can be queried programmatically using the API or through SPARQL queries. For example, the list of all promoters from all the different sources in this SBOL Stack instance can be retrieved using a single query. The role property of a `ComponentDefinition` entity indicates the genetic feature represented and is specified with Sequence Ontology[12] (SO) terms. In Figure 5, this property is used together with the SO promoter term (`SO:0000167`) to filter for promoter definitions only.

Any URI can be used as a role, even without an associated ontology. In the iGEM parts registry, each part is associated with a set of categories. For example, constitutive promoters are associated with the `//promoter` category and the `//regulation/constitutive` category. In the SBOL Stack representation of the parts registry, categories are



DOI: 10.1021/acssynbio.5b00210
ACS Synth. Biol. 2016, 5, 487−497

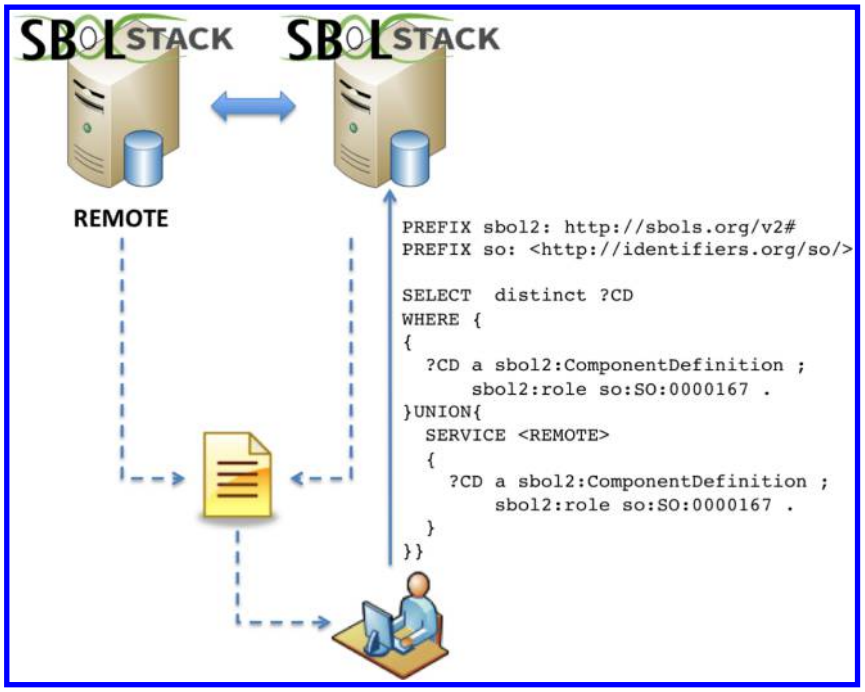



**Figure 8.** Using SPARQL queries to facilitate data federation. Remote SBOL Stack end points are indicated with the REMOTE keyword. The query result is the union of `ComponentDefinition` entities representing promoters from both the local and the remote repositories.

represented using the SBOL `role` property. This enables iGEM parts to be queried by category using standard SBOL Stack functionality to search by role (Figure 6).

Different types of SBOL properties can also be used in searches, including custom annotations. For example, the *Bacillus subtilis* data set provided includes annotations about taxonomy identifiers. This predicate can be used in an SBOL Stack query to retrieve all `ComponentDefinitions` for parts derived from *Bacillus subtilis*, which has taxonomy identifier 1423 (Figure 7).

**1.4. SBOL Stack Data Federation.** Data federation is another form of data integration that is supported by the SBOL Stack. The SBOL Stack supports this approach through its adoption of the existing Semantic Web technology, SPARQL, and HTTP access to remote triplestores. Using this approach, data about synthetic biology designs can be retrieved from multiple SBOL Stack instances without a need to move data from one instance to another. One of the acknowledged challenges in data federation is the agreement on shared semantics, since data repositories often store data using different semantics. The SBOL Stack uses SBOL and SyBiOnt to provide common semantics and, therefore, a query constructed using SBOL and SyBiOnt terms can be executed over multiple SBOL Stack instances.

The SBOL Stack supports executing federated queries through SPARQL queries. These queries themselves are graphs and are used to find matching patterns over large graph structures. Parts of these SPARQL queries can be constructed in a way that looks for data from remote repositories, while the rest of the query can be used to search for data from the local repository. Therefore, as well as querying data contained within an SBOL Stack instance, the SBOL Stack API can also be used to integrate data from external sources through such queries. These sources can be other instances of the SBOL Stack or any other SPARQL end point. This support for query federation allows multiple, disjoint data sets to be joined. Figure 8 shows a SPARQL query to select all promoters (SO term `SO:0000167`) both from the local data set and a remote data set.

The SBOL Stack can also automatically federate queries across other SBOL Stack instances without manually writing a federated SPARQL query. The SBOL Stack configuration file contains an `otherStacks` property, which lists the URLs of other SBOL Stack instances to aggregate data from. When a request is received by the SBOL Stack API, these other instances will be queried in parallel and the results collated into a single SBOL document to deliver back to the client. This federation also applies to the end points for counting components or modules, which will query other stacks and sum the total value of matching entities. We have tested this approach in a simple three-way querying exercise between Newcastle University (UK), Boston University (US), and Macquarie University (Australia). In this test the genetic toggle switch design from Galdzicki *et al.*[10] was built by retrieving promoters, CDSs, and cloning vectors from these three instances of the SBOL Stack. In the future, we envisage more complex examples will be developed when the number of SBOL Stack instances increases and there are data available on a wider variety of parts.

**1.5. SynBioHub Functionality and Use Cases.** SynBioHub is a Web application to demonstrate the use of the SBOL Stack as an SBOL version 2 repository and to support the end user in this respect. It is designed to allow a user to view and manage SBOL version 2 designs manually. SynBioHub is user-centric and each group of users (*e.g.*, a lab or institute) can be assigned a private repository. Users can upload new designs in the form of GenBank or SBOL documents and track them privately. Submissions are also associated with metadata, which is used to add information about the upload process itself. SynBioHub also demonstrates the suitability of the SBOL Stack for a more complicated scenario by providing an abstraction over the security model, implementing a dynamic user


DOI: 10.1021/acssynbio.5b00210
*ACS Synth. Biol.* 2016, 5, 487−497

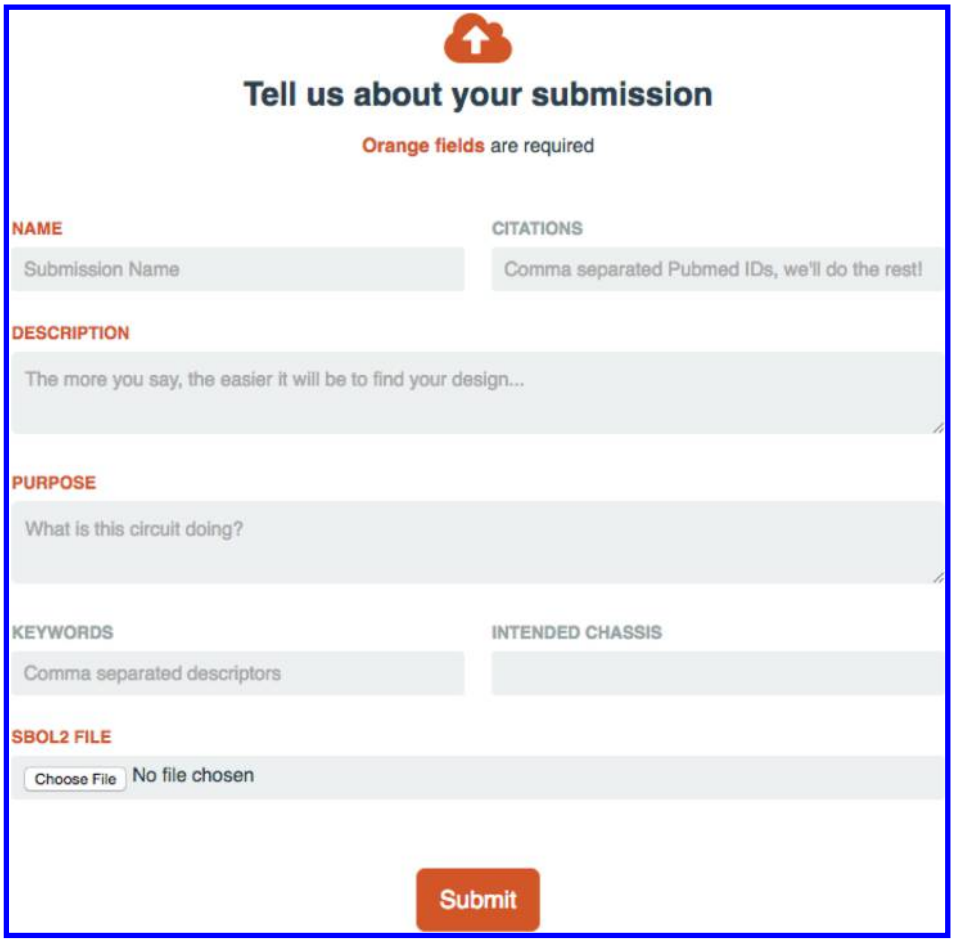



**Figure 9.** SynBioHub users can provide more information about the submission such as a name and a description, what the design is about, associated citations, the intended chassis, and a list of keywords.

registration system where each user is associated with their own unique store.

*1.5.1. Uploading and Sharing Synthetic Biology Designs.* Hub users must first register in order to upload new data and can then access their private repositories with a username and password. When logged in, a user can upload, download, and modify designs in their private repository, share designs with other users, and make designs public. Once public, the designs can be accessed *via* the Web by any visitor without the requirement of a SynBioHub account. Sharing designs is a key focus of SynBioHub, and the infrastructure provided by the SBOL Stack enables users to share designs both through the public repository and privately through their uniquely assigned URL.

When submitting a new design to SynBioHub, users are presented with a set of fields with which they can specify metadata for their design (Figure 9). Among these fields are name; list of citations; description; purpose; list of keywords; and intended chassis. The user can then upload an SBOL file or can upload a GenBank file to be converted into SBOL for storage in the triplestore. During the submission process, the metadata is converted into top level annotations that are given newly minted URIs in the SynBioHub namespace and attached to the SBOL file. If the design is later downloaded, this metadata stays associated with the SBOL file and is downloaded as part of the serialisation. It should be noted that no new URIs are produced for files uploaded in SBOL format, since the triplestore is able to use the URIs found in the SBOL files. New URIs are created for newly converted GenBank files that are tagged as being created by SynBioHub.

Users can search for designs using information from the metadata such as title, description, and keywords. When searching or browsing, designs in SynBioHub can be viewed on the submission page, where a list of designs owned by a user is presented along with designs from the public repository (Figure 10). Users can then choose particular designs that are private to be made public. Currently, each SynBioHub instance is connected to a single instance of the SBOL Stack. However, this instance of the SBOL Stack can be connected to other instances using the data federation paradigm described in Section 1.4. As a result, queries performed on a SynBioHub instance will be automatically federated to other connected instances of the SBOL Stack. The results will then be collated into a single SBOL document and returned to the user.

*1.5.2. Browsing Data from the SBOL Stack.* SynBioHub users can either browse the entire data from a configured SBOL Stack instance or list their submissions. When viewing a single design, a user is presented with a page listing all of the metadata associated with the design, as well as options to download the SBOL file. Additionally, for DNA components with functional assignments, a graphical representation of the design that conforms to the SBOL Visual standard[26] is generated using VisBOL[20] and is presented to the user. This graphical representation can be exported in several different graphical file formats including SVG and PNG. Designs can then be included on slides in a presentation or as a figure on a poster or



DOI: 10.1021/acssynbio.5b00210
*ACS Synth. Biol.* 2016, 5, 487−497

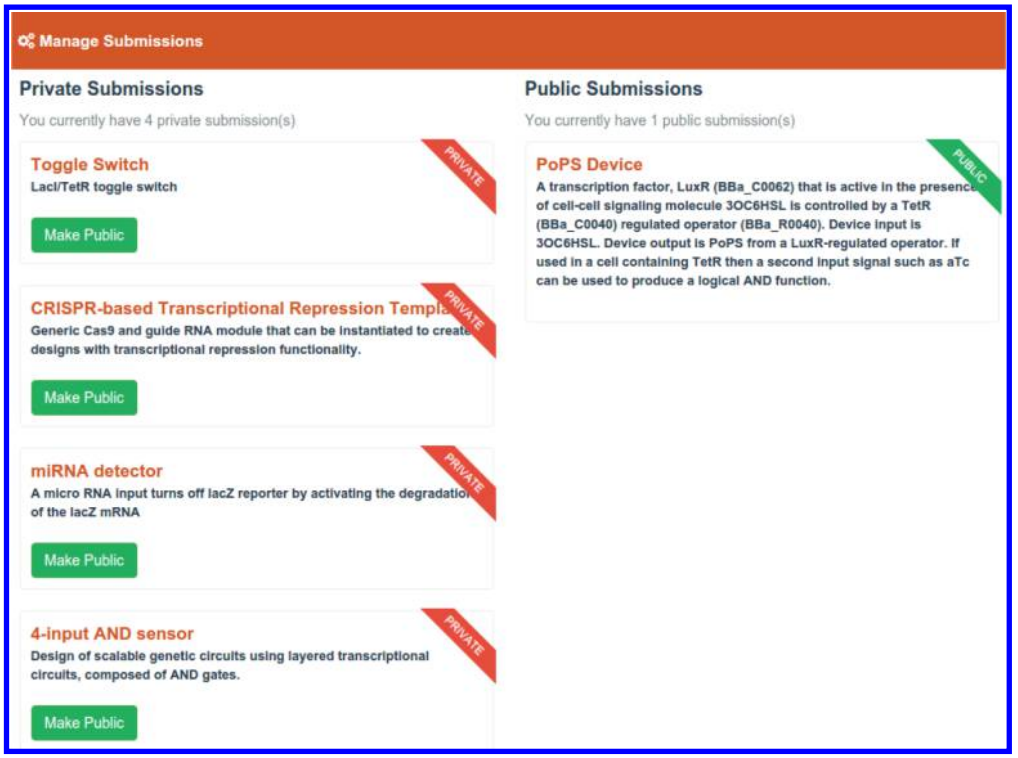



**Figure 10.** Hub users can track both their private and public designs using the Manage Submissions page. Private submissions belonging to the user are shown on the left, and public submissions belonging to the user are shown on the right. The user can make a private submission public by clicking the Make Public button.

publication. Figure 11 depicts a screen shot of the SynBioHub page for a design of a genetic toggle switch.

*1.5.3. Hosting a SynBioHub Instance.* The Hub can be configured to connect to any SBOL Stack instance, whether hosted on the same machine or remotely. This configuration is carried out when the Hub is accessed for the first time by an administrator. Other users can then start registering and using the Hub instance (Figure 12).

## 2. DISCUSSION

SBOL has been developed to facilitate a standard approach to sharing designs in the field of synthetic biology. The SBOL Stack fulfills the need for an open-source database that allows SBOL designs to be stored in their native RDF format that is amenable to sharing, composition, annotation, and integration. The development of SBOL as an emerging standard for the interchange of biological designs also brings new opportunities for enabling collaborative workflows between scientists and commercial entities across geographical, institutional, and political boundaries. This distributed approach to synthetic biology also requires repositories where designs can be programmatically stored, retrieved, and shared during the execution of the workflows. The SBOL Stack has also been designed to meet this need.

It has been recognized that the methods for the storage and sharing of parts and designs have not been developed sufficiently to meet the demands of the growing number of parts and the need to share and compose these parts.[9] Ham and co-workers produced the ICE system in response to this challenge and first introduced the concept of a web of registries that would allow users to maintain their own repository that could be linked to repositories of other users, applications, and institutions. In the work presented here, we have built on this concept to introduce a repository that not only allows parts to be stored, retrieved, and shared in the emerging standard format SBOL version 2, but also facilitates an integrated approach to their use and application.

The SBOL Stack allows data to be stored in a flexible manner since the data is broken down into the component RDF triples and stored in a triplestore. At a basic level this approach has the advantage of leveraging already available standard libraries and the SPARQL querying language for data retrieval and federation. As it is based on Sesame, the system is also cross-platform and easy to deploy on systems running a Java Web server. The SPARQL end point for the SBOL Stack enables computational access to the Stack through SPARQL queries for developers who wish to take this route. However, the API and the client library also allow computational tools to programmatically access the SBOL Stack using several search options. Triplestores have been developed to support the Semantic Web concept (Web 2.0) which has an integrated Web of resources at the heart of its concepts.[27] As such, the use of Semantic Web technology allows the use of standardized approaches to integrating data from multiple databases in the Web in a dynamic fashion—data federation. Using this approach, users can query multiple databases simultaneously and combine the search results. This approach will allow the collaborative development of systems where parts are not only retrieved from multiple repositories but are in a form that also allows their integration and modular combination by software applications such as CAD tools. In the future, we plan to build composition functionality into the SBOL Stack API itself.

In addition to operating in a federated mode, each SBOL Stack instance can also act as a data warehouse capable of storing and integrating data from other sources with the SBOL formatted parts data. A large amount of information about


DOI: 10.1021/acssynbio.5b00210
ACS Synth. Biol. 2016, 5, 487−497

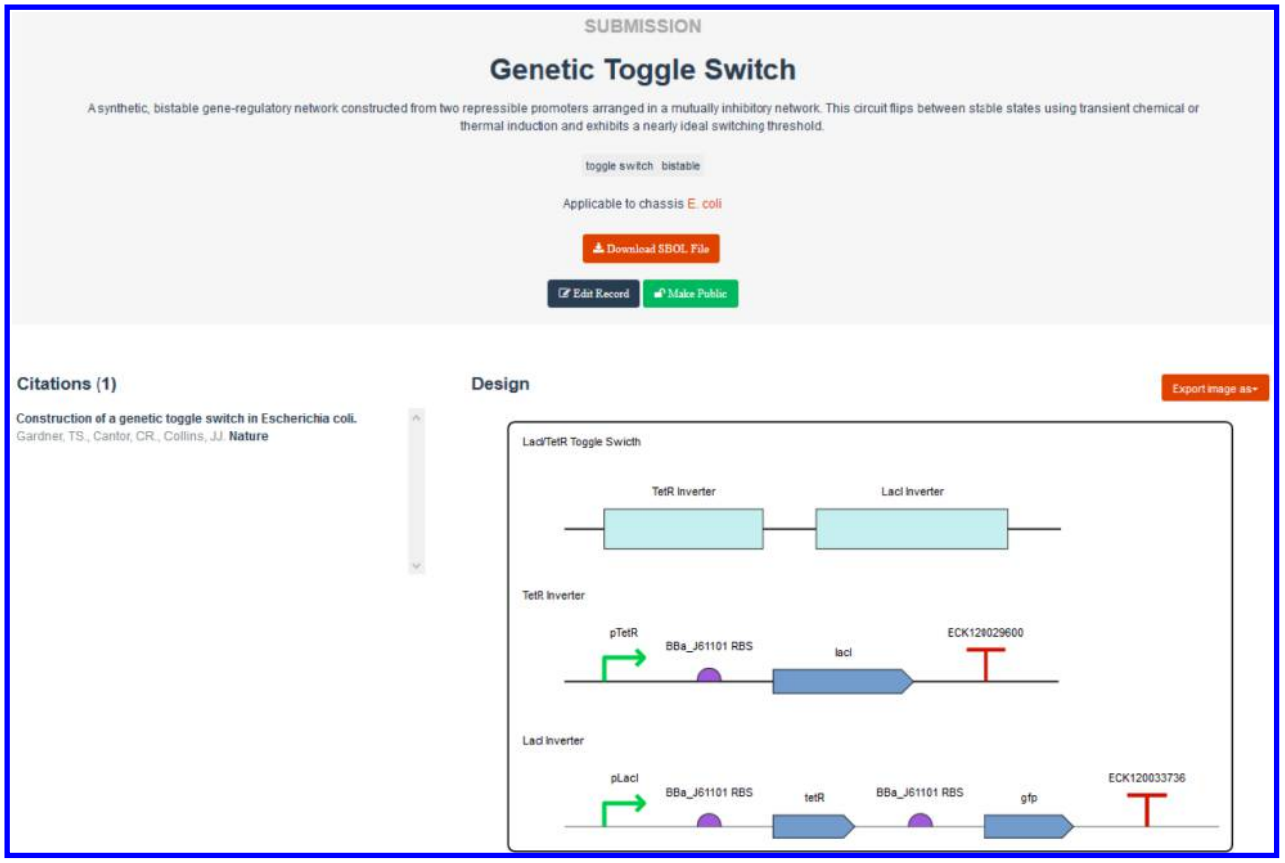



**Figure 11.** A screen shot of the SynBioHub page for a genetic toggle switch. This page includes a title, a description, a list of keywords, an intended chassis, a list of citations, and a visual representation of the design rendered using VisBOL. With the buttons below the description, any user can download an SBOL file from the submission, and the owning user can edit metadata corresponding to the submission or make the submission public if currently private.

genetic features, their biological role, and their functional interactions is now freely available in a myriad of databases. However, these databases often have different file formats and different semantics, making it difficult to automatically assimilate the information necessary for biological system design. The RDF orientation of SBOL makes it extremely flexible, and facilitates data integration, both with RDF and non-RDF data. SBOL data can, for example, be integrated with annotations from ontologies such as the Sequence Ontology and the Gene Ontology,[28] making it possible to add semantic information to biological part data. Where the SBOL semantics are not sufficient for data representation, additional ontologies such as SyBiOnt[18] can be used to mediate the integration of data. This approach offers the potential to enrich the designs for parts with external information and opens up the possibility of a more informed approach to parts design. Moreover, it should be possible in the future to store the characterization data relating to parts in the SBOL Stack by extension of the SBOL vocabulary using many existing ontologies and data standards. The inclusion of information about small molecules, interactions, RNA, proteins, *etc.* as supported by SBOL version 2 also opens up the possibility of designing biological systems at higher levels in the molecular biology hierarchy than the genetic level at which most current designs are composed. In the future, we aim to include phenotypic and physiological data in SBOL Stack databases to further facilitate the high-level design of biological systems.

In addition to the SBOL Stack, we have also developed a web-based tool, SynBioHub, that allows the end user to access the SBOL designs in the SBOL Stack. SynBioHub acts as a demonstration for other developers on how to programmatically access the SBOL Stack. However, SynBioHub is also a fully functional system that will allow users to upload their parts in the SBOL or GenBank format and to retrieve parts from an SBOL Stack installation. If that SBOL Stack instance is connected to other SBOL Stack installations using the federated querying system, then parts from those other SBOL Stack instances will also be accessible from that SynBioHub instance. The SynBioHub Web client allows users to upload entire SBOL documents to a private database and to share those records with other users or make them public. In the future, SynBioHub could be a useful tool in the academic journal submission process so that synthetic biology designs can be evaluated and made available at the same time as the papers that refer to them.

The release of the SBOL Stack and SynBioHub described here offer a useful set of functions that we hope will be of benefit to the wider scientific community, not just SBOL software developers. Moreover, there are many more new functions and features that could be added to these systems. It is our hope that the open source nature of the tools will encourage the synthetic biology developer community to adopt and extend this software.

## 3. METHODS

The SBOL Stack and SynBioHub were written in JavaScript. The Web service client API was developed in Java. The Sesame



DOI: 10.1021/acssynbio.5b00210
*ACS Synth. Biol.* 2016, 5, 487−497

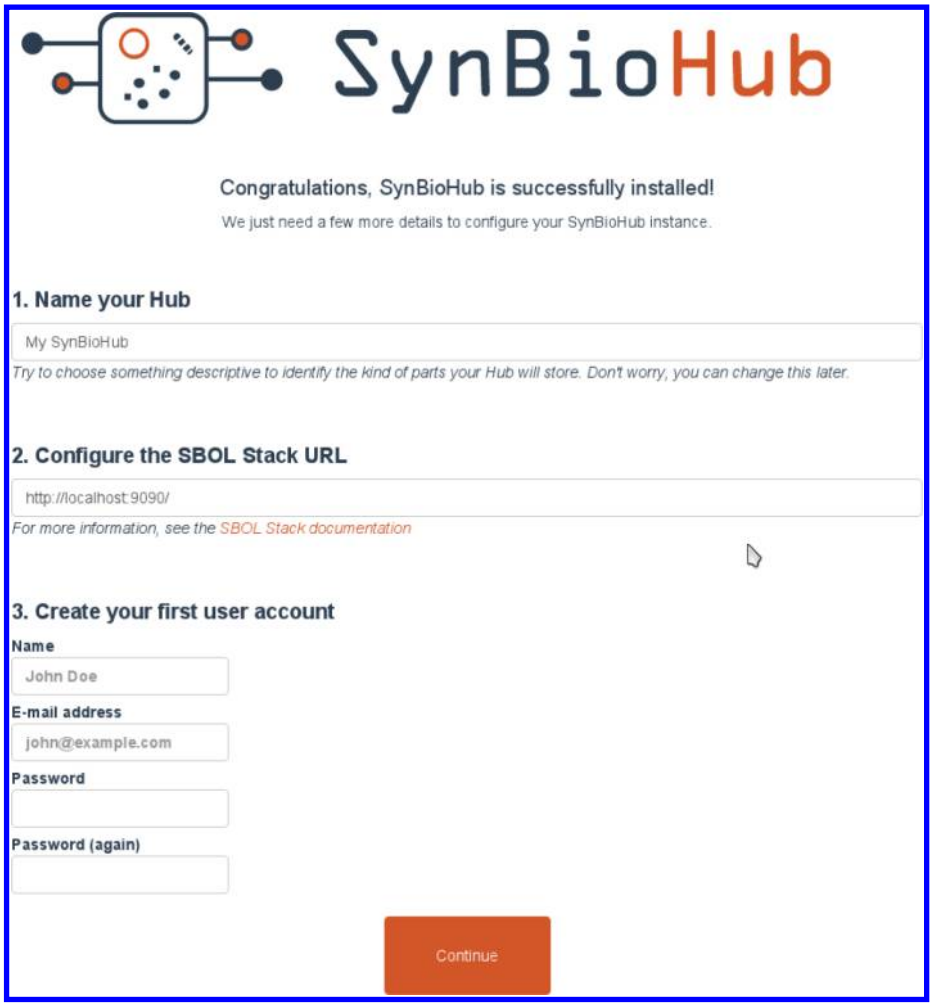



**Figure 12.** Setup page shown when SynBioHub is accessed for the first time. The user can give the new SynBioHub a name, configure the URL of an SBOL Stack instance to connect to, and create the first user account in order to login after setup.

triplestore was used as an RDF database. Sesame is an open-source implementation of an RDF triplestore written in the Java language.[16] In the future, the SBOL Stack will be migrated to Eclipse rdf4j, a recently released fork of Sesame.

**3.1. Software Availability.** The SBOL Stack can be accessed *via* www.sbolstack.org. This site contains a link to the source code for the SBOL Stack triplestore, the SynBioHub Web client, and Docker images to enable deployment of new Stack instances. It also contains a link to our example instance of the SBOL Stack with a front-facing SynBioHub Web interface that can be accessed *via* www.synbiohub.org. This instance contains the data from the aforementioned *Bacillus subtilis* data set, the iGEM Registry of Standard Biological Parts, and SBOL version 2 example files from recent publications. The entire source code for the SBOL Stack and the SynBioHub are available under the BSD license.

## AUTHOR INFORMATION


### Corresponding Author

*E-mail: anil.wipat@ncl.ac.uk.

### Present Address

[⊥]Macquarie University, Sydney NSW 2109, Australia.

### Author Contributions

[‖]C.M. and J.A.M. contributed equally to this work.


### Notes

The authors declare no competing financial interest.

## ACKNOWLEDGMENTS

The authors thank iGEM Headquarters for providing access to the Registry of Standard Biological Parts dataset. C.M. has been supported by the Engineering and Physical Sciences Research Council grant EP/K039083/1; A.W. and G.M. have been supported by the Engineering and Physical Sciences Research Council grant EP/J02175X/1. J.A.M. is supported by FUJIFILM Diosynth biotechnologies.

## REFERENCES

(1) Galdzicki, M. (2011) *Synthetic Biology Open Language (SBOL) Version 1.0.0. BBF RFC 84*, DOI: 1721.1/66172.

(2) Galdzicki, M. (2012) *Synthetic Biology Open Language (SBOL) Version 1.1.0. BBF RFC 87*, DOI: 1721.1/73909.

(3) Bartley, B., Beal, J., Clancy, K., Misirli, G., Roehner, N., Oberortner, E., Pocock, M., Bissell, M., Madsen, C., Nguyen, T., Zhang, Z., Gennari, J. H., Myers, C., Wipat, A., and Sauro, H. (2015) Synthetic Biology Open Language (SBOL) Version 2.0.0. *J. Integr. Bioinf. 12*, 272.

(4) Gilfellon, O., Misirli, G., Madsen, C., Hallinan, J., Zuliani, P., and Wipat, A. (2015) SynBad: An SVP Design Framework. *7th International Workshop on Bio-Design Automation (IWBDA)*.



DOI: 10.1021/acssynbio.5b00210
*ACS Synth. Biol.* 2016, 5, 487−497

(5) Myers, C. J., Barker, N. A., Jones, K. R., Kuwahara, H., Madsen, C., and Nguyen, N.-P. D. (2009) iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics 25*, 2848−2849.
(6) Chandran, D., Bergmann, F., and Sauro, H. (2009) TinkerCell: modular CAD tool for synthetic biology. *J. Biol. Eng. 3*, 19.
(7) Lu, G., and Moriyama, E. N. (2004) Vector NTI, a balanced all-in-one sequence analysis suite. *Briefings Bioinf. 5*, 378−388.
(8) Misirli, G., Hallinan, J., and Wipat, A. (2014) Composable Modular Models for Synthetic Biology. *ACM J. Emerg Technol. Comput. Syst 11*, 1−19.
(9) Ham, T. S., Dmytriv, Z., Plahar, H., Chen, J., Hillson, N. J., and Keasling, J. D. (2012) Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res. 40*, e141.
(10) Galdzicki, M., et al. (2014) The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol. 32*, 545−550.
(11) Manola, F., and Miller, E., Eds. (2004) *RDF Primer*, W3C Recommendation, World Wide Web Consortium.
(12) Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M., Stein, L., Durbin, R., and Ashburner, M. (2005) The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biol. 6*, R44.
(13) Prud'hommeaux, E., and Seaborne, A. (2008) *SPARQL Query Language for RDF*. W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/.
(14) Galdzicki, M., Rodriguez, C., Chandran, D., Sauro, H. M., and Gennari, J. H. (2011) Standard Biological Parts Knowledgebase. *PLoS One 6*, e17005.
(15) Peccoud, J., Blauvelt, M. F., Cai, Y., Cooper, K. L., Crasta, O., DeLalla, E. C., Evans, C., Folkerts, O., Lyons, B. M., Mane, S. P., Shelton, R., Sweede, M. A., and Waldon, S. A. (2008) Targeted Development of Registries of Biological Parts. *PLoS One 3*, 1−7.
(16) Broekstra, J., Kampman, A., and Harmelen, F. v. (2002) Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *Proceedings of the First International Semantic Web Conference on The Semantic Web*, London, UK, pp 54−68.
(17) Misirli, G., Wipat, A., Mullen, J., James, K., Pocock, M., Smith, W., Allenby, N., and Hallinan, J. (2013) BacillOndex: An Integrated Data Resource for Systems and Synthetic Biology. *J. Integr. Bioinf. 10*, 224.
(18) Misirli, G., Hallinan, J., Pocock, M., Lord, P., McLaughlin, J. A., Sauro, H., and Wipat, A. (2016) Data Integration and Mining for Synthetic Biology Design. *ACS Synth. Biol.*, DOI: 10.1021/acssynbio.5b00295.
(19) Tilkov, S., and Vinoski, S. (2010) Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing 14*, 80.
(20) McLaughlin, J. A., Pocock, M., Misirli, G., Madsen, C., and Wipat, A. (2016) VisBOL: Web-based tools for synthetic biology design visualization. *ACS Synth. Biol.*, DOI: 10.1021/acssynbio.5b00244.
(21) Ecma International (2013) ECMA-404: The JSON Data Interchange Format.
(22) Zhang, Z., Nguyen, T., Roehner, N., Misirli, G., Pocock, M., Oberortner, E., Samineni, M., Zundel, Z., Beal, J., Clancy, K., Wipat, A., and Myers, C. J. (2016) libSBOLj 2.0: A Java Library to Support SBOL 2.0. *IEEE Life Sci. Lett. 1*, 34−37.
(23) Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. (2013) GenBank. *Nucleic Acids Res. 41*, D36−D42.
(24) Nguyen, T., Roehner, N., Zundel, Z., and Myers, C. J. (2015) A Converter from the Systems Biology Markup Language to the Synthetic Biology Open Language. *ACS Synth. Biol.*, DOI: 10.1021/acssynbio.5b00212.
(25) Roehner, N. (2016) Sharing Structure and Function in Biological Design with SBOL 2.0. *ACS Synth. Biol.*, DOI: 10.1021/acssynbio.5b00215.
(26) Quinn, J. Y., et al. (2015) SBOL Visual: A Graphical Language for Genetic Designs. *PLoS Biol. 13*, e1002310.
(27) Lewis, D. (2006) What is Web 2.0? *Crossroads 13*, 3−3.
(28) Ashburner, M., et al. (2000) Gene Ontology: tool for the unification of biology. *Nat. Genet. 25*, 25−29.


DOI: 10.1021/acssynbio.5b00210
*ACS Synth. Biol.* 2016, 5, 487−497