Submitting BrainML Models

1. Introduction

This page describes how to go about submitting a model to the BrainML repository as a series of steps. In general, a representative of a research community will submit a model for the data the community wishes to store in a repository. This model can reuse components of other models where characteristics of the data are shared in common, and define new components where needed to express distinct types of information.

A model consists of an XML Schema, defining XML tags and their data types, and referencing other schemas using standard mechanisms as well as those described in the specification. It is recommended that you read the materials elsewhere on this web site, browse the models currently available, and explore the tools hosted at neurodatabase.org (which are driven by a data model in a precursor format to BrainML; see extensions).

In particular, in addition to a knowledge of XML Schemas, users should read the following documents prior to creating a new model:

- Overview of BrainML
- BrainML Specification
- BrainML and BrainMetaL Components

Note that `BrainML.org` supports the use of XML Schema Compact Syntax (XSCS), a simpler format for editing XML schema. Models can be submitted (and browsed) in either compact or regular syntax. Use of compact syntax is recommended if you are editing schemas by hand.

2. Sign up for an Account

To submit a model, you must have a registered account at `BrainML.org`. This is to prevent abuse of the repository; the only registration requirement is that the user must be a neuroscientist with an institutional affiliation.

To sign up for an account:

1. Select Model Schemas at the upper left on this site.
2. Complete the fields on the lower left with your desired user ID and password.
3. Click the button for 'Sign Up'. This will direct you to a registration form.
4. Fill in and submit the form by pressing the 'Submit' button. An email will be sent to the

administrator of this site who will review your information. Approval generally requires 48 hours. If you have not heard from the administrator withing 48 hours, please email the Laboratory of Neuroinformatics at the following email address: neurodatabase AT med.cornell.edu

3. Copy an Existing Model

The easiest way to start building a model is to start with an existing schema in compact format and modify it to suit your needs. This helps jump start the process and assists with getting syntactic details right. A good schema to start with is this one, from the Cortical Neurophysiology model.

In the following, we will use this particular model as an example to illustrate points about building models in general.

4. Declare Model Namespace

The primary identifying component of any model is its *namespace*. This is a string unique to the model constructed according to certain conventions. The string is declared in the initial schema definition block as shown here for the cortical neurophysiology model (compact syntax):

```
targetNamespace "urn:bml/brainml.org:med.cornell.edu/Cortex/1"
namespace       "urn:bml/brainml.org:med.cornell.edu/Cortex/1"
namespace bml   "urn:bml/brainml.org:internal/BrainML/1"
namespace bmtl  "urn:bml/brainml.org:internal/BrainMetaL/1"
namespace xlink "http://www.w3.org/1999/xlink"
```

Here,

- `targetNamespace` and the first `namespace` entry describe the namespace for the model. The last three namespace references indicate models that this model depends on (incorporates parts of in its own definitions).
- `urn:bml/brainml.org:` is a fixed part that is common to all BrainML namespaces. It indicates that this is a BrainML model hosted at brainml.org.
- `med.cornell.edu` represents the domain name of the organization publishing the model. This should be replaced in your own model by your institution domain name (as specified in your account information).
- `Cortex` represents the name of the model itself. You should replace this with a short name, not containing spaces, that describes the model and is unique within your institution.
- `1` is the version number of the model. Version numbers start at 1, and are incremented whenever an augmented or modified version of the model is submitted. Earlier versions remain available in the repository and can be referenced by their version number to support data using them.

The same schema header above in full XML syntax is:

```
<xs:schema elementFormDefault="qualified"
    targetNamespace="urn:bml/brainml.org:med.cornell.edu/Cortex/1"
              xmlns="urn:bml/brainml.org:med.cornell.edu/Cortex/1"
           xmlns:bml="urn:bml/brainml.org:internal/BrainML/1"
          xmlns:bmtl="urn:bml/brainml.org:internal/BrainMetaL/1"
         xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

Notice that here the full syntax requires a couple of extra elements: the 'elementFormDefault' attribute, and the last namespace for XML Schema itself. These are fixed components required by the full syntax and can be ignored.

5. Import Referenced Models

In order to reuse components from other models given as namespaces above, they must be *imported*. This is done with lines of the following form (see original schema for full text).

```
import "http://brainml.org/schemas/internal/BrainML/1/brainml.xsd"
    namespace "urn:bml/brainml.org:internal/BrainML/1"
```

In full XML syntax, this is:

```
<xs:import namespace="urn:bml/brainml.org:internal/BrainML/1"
         schemaLocation="http://brainml.org/schemas/internal/
BrainML/1/brainml.xsd"/>
```

6. Define Components

Components are defined using the standard mechanisms of XML schema: simpleTypes, complexTypes, attributes, and elements. By convention in BrainML schemas, all major types are defined as named top-level complexTypes with a name ending in "-type". The element is then defined with the same name without the "-type". This makes it easy to reuse individual components in other models, following the "salami slice" schema design paradigm. For example:

```
/* Experimental protocol. */
complexType cortical_protocol-type extends bml:protocol-type {
   (  preparation,
      description,
      stimulus_nudge?,
      bml:link_experiment? )
}
```

This defines a type for describing an experimental protocol. The first line is a comment, and is mandatory by BrainML convention, for it provides the main documentation for the type. The second line names the type, and declares it to inherit from `protocol-type` in the BrainML base model. The next set of lines declare the contents of this type: each term refers to another local definition in this model (cortical neurophysiology), except for the last, which refers to a definition in the BrainML base model. The commas indicate sequential ordering of the contents, and the question marks indicate optional occurrence.

Two of the local definitions are shown here:

```
/* Preparation type used in this Protocol. */
element preparation restricts bmtl:vocab-type {
   attribute domain {xs:token} = "protocol.preparation"
}

/* Short verbal description of Protocol. */
element description { xs:string }
```

The first definition here represents a controlled vocabulary field, marked by "`restricts bmtl:vocab-type`". The 'domain' attribute is given a fixed value, declaring the domain a valid term for this field should come from (see specification).

Finally, the cortical_protocol element itself is declared:

```
/* Experimental protocol. */
element cortical_protocol substitutes bml:protocol
   { cortical_protocol-type }
```

The "`substitutes` ..." here declares that this element may be used in places where a `protocol` element is called for in models. This is analogous to the ability in object-oriented programming languages to use a subclass where a superclass is called for. Unlike in OO languages, however, this capability must be declared separately in XML schema from the declaration of inheritance.

7. Leverage Existing Model Structures

The BrainMetaL and BrainML base packages offer many components that a model-builder can leverage to create a customized data model. These components include general-purpose structures for:

- containing data in specific structures such as unbinned histograms, X-Y datasets, etc;
- metadata to sufficiently describe the data;
- indicating the author(s) of data;
- linking the data to other sets of data;

In each instance, you have the option of using the base components directly, creating new definitions inheriting from them, or coming up with completely new, independent definitions. (These cases are all illustrated in the example above.)

BrainMetaL includes a set of five top-level classes we refer to as the *Quintessence* elements. These are highly generic entities that all major components in the BrainMetaL and BrainML base packages descend from by inheritance. We recommend that all models follow this convention. The Quintessence is used by software to determine basic structure in a data document, and it also provides a helpful initial level of organization for model semantics.

Below is an example of defining an entity directly descendant from a Quintessence element. (In most cases you will not define something directly descending from the Quintessence, but the idea is the same.) Here, `data_element-type` is the Quintessence class, and `view-type` is the descending entity.

```
/* Base content for a View */
complexType view-type extends bmtl:data_element-type {
   ( link_recording_site*, label )
   /* Ordering position of this View within its containing experiment. */
   required attribute seq { xs:int }
}
```

8. Validate the New Model

You can check the new model for validity as an XML schema and compatibility with any existing BrainML components it references by clicking the Validate link on the left side navigation bar. Instructions are given on that page.

9. Publish the Model

Using this model server, publishing the model is easy. From the View Models page (also available through the "Model Schemas" link at top left), select "Submit new model...". (You must be logged in to perform this action.) Then follow the screen there to create a Model. This just creates a placeholder. You then submit the schema itself as a *version* of the model (the first version, to be exact).

Before a model is published, it must currently be reviewed by the BrainML.org staff. This process should take place within 2-3 days, at which point you will be notified of acceptance. At this time the model and its first version will become visible on the site. Submitting subsequent versions does not require this review step.