# Bioinformatic Sequence Markup Language

# BSMLÔ 3.1 Tutorials

# Bioinformatic Sequence Markup Language

# BSML™ 3.1 Sequence Tutorials

## Table of Contents

# BSML™ 3.1 Tutorials

## 1. Introduction to the BSML Sequence Tutorials

Note: This manual contains material previously presented in "BSML 2.2 Sequence Tutorials." BSML 3.1 is backward compatible with BSML 2.2. A set of example BSML documents is included with the installation of the Genomic XML Viewer™.

BSML is primarily concerned with biological sequences – DNA, RNA, and protein. The information of interest falls into two general categories – **sequence data** (sequencing runs, clones, contigs, cDNA, etc.) and sequence **annotation** (assertions about the properties of sequences). BSML was developed under a 1997 grant from the National Human Genome Research Institute, with the goal of creating a public domain standard for representing this information. The purpose of these tutorials is to enable people in two communities – life sciences and information technology – to make effective use of BSML. The first few tutorials in this series provide a general answer to three questions:

- **What** is BSML?
- **Why** is BSML useful?
- **How** can BSML be used?

**BSML users.** Within the life sciences and information technology disciplines, there are two general audiences to which these tutorials are directed:

- **Producers** - people who are interested in creating BSML content
- **Consumers** - people who are interested in using information encoded in BSML

Of course, a single individual or organization may be both a producer and a consumer. A scientist may wish to interact as a consumer with data already encoded in BSML, and then as a producer to edit the existing content by adding new annotation. After the general introduction, we present tutorials on the use of BSML by producers and consumers.

**BSML, XML, and software.** BSML is an XML application. This means that any software that can read XML can read BSML, and any software that can render XML in a generic way can render BSML. For example, an XML-aware HTML browser is capable of displaying BSML as text. However, generic software will have little capability to visualize sequence information in the way that was intended for data encoded in BSML. Similarly, software that is not "bioinformatics-aware" will not be able to process the links to information resources contained in BSML documents (e.g., a cross-reference to a public database).

Sequences may be described at various levels: complete genome, chromosome, regulatory region, gene, transcript, gene product, and so forth. Each of these levels may

be thought of as defining a type of **biological object**. BSML is useful in simply capturing the semantics of biological objects, but it is especially useful in conjunction with appropriate software that enables interactive access to the full range of information that BSML can encode.

Why is software important? First, information technology makes it possible to render information in ways that are simply not available through print media. The ability to navigate through an information space using zooming and linking technologies brings new dimensions to information access. Second, life scientists are accustomed to visualizing biological objects and to communicating graphically about these objects and their annotations. Thus the capabilities of graphical rendering software are well suited to the communication needs of life scientists.

For these reasons, we devote several tutorials to the relationship between BSML and rendering software, with emphasis on the BSML browsers available from LabBook, Inc., including the free Genomic XML Viewer. (See ***Rendering BSML in the Genomic XML Viewer***.)

## 2. What Is BSML?

**XML and HTML.** XML (eXtensible Markup Language) was developed by the World Wide Web Consortium (W3C) as a language that builds on the strengths of HTML (HyperText Markup Language) and corrects some of its deficiencies. For example, XML also embodies the use of links to connect one document element with another. At the same time, XML presents a model for capturing semantic content rather than presentation style, the latter being the focus of HTML.

To illustrate this difference, consider the representation of "Jane Doe" in a document. In an HTML document, the name might be represented as:

```
<B>Jane Doe</B>
```

This notation tells the rendering software (HTML browser) to display the name using a bold **(<B>**) font. However, the HTML encoding does not truly distinguish this content as a person's name, and a search of HTML pages may have difficulty separating this "Doe" from "doe", a female deer, and "DOE", the Department of Energy. An XML encoding might state:

```
<Person>
  <LastName>Doe</LastName>
  <FirstName>Jane</FirstName>
</Person>
```

"Tags" (e.g., <**Person**>) are used in both cases to "mark up" the content, but only in the XML case do the tags capture the semantics of the content, i.e., that "Jane Doe" is a person's name consisting of a last name and a first name.

**BSML is XML.** Technically, BSML is an XML application that is represented by a Document Type Definition (DTD). This means that BSML conforms to the XML standard - a BSML document contains XML content that can be read by any XML parsing software and determined to be **well-formed** XML. Because of the BSML DTD, the parsing software may also **validate** the document to determine that it complies with the document rules specified in the BSML DTD.

**BSML documents.** BSML documents *are* documents. This means that they are readable by people as well as machines. A complete BSML document begins with the standard XML processing instruction and a reference to the BSML DTD:

```
<?XML VERSION="1.0"?>
<!DOCTYPE BSML PUBLIC "HTTP://LABBOOK.COM/DTD/BSML3_1.DTD">
```

The document content is enclosed between **Bsml** tags. This element is defined in the 3.1 DTD as:

```
<!ELEMENT Bsml
         (Attribute*, Info*,
```

```
Definitions?,
Research?,
Display?)>
```

General information (document author, creation date, etc.) may be attached to the entire document by using **Attribute** and **Info** elements.

**BSML and Bioinformatic Sequences.** BSML is *not* intended for the description of geological strata or subatomic particles. It is an XML application in the **domain** of **bioinformatic sequences**. BSML makes use of much of the functionality provided by XML and related standards, but to understand BSML requires understanding how it is applied in its domain.

Within the <**Bsml**> tags, the document is divided into three major sections, each of which is optional:

- **Definitions** – encoding of genomes and sequences, data tables, sets, and networks
- **Research** - encoding of queries, searches, analyses and experiments
- **Display** – encoding of display widgets that represent graphical representations of biological objects

**Note:** BSML builds on various public standards for the representation of information, such as NCBI's (the U.S. National Center for Biotechnology Information) representation of GenBank content.

## 2.1 Biological Sequences

What characteristics of a biological sequence provide enough information to identify and describe the sequence? The information needed to accomplish these tasks is encoded in the **Sequence** element of BSML.

### 2.1.1 Basic Sequence Descriptions

The minimum information needed to describe a sequence is an indication of the type of **molecule** that is being represented and its **length** in residues. BSML represents all of this information using **attributes** and their values, which are encoded in the XML as:

```
attribute-name="value"
```

For example, the most basic description of a sequence might be:

```
<Sequence molecule="dna" length="1000"></Sequence>
```

This identifies a DNA sequence that is 1000 bp long. In the DTD, we define these attributes as follows:

```
molecule (mol-not-set | dna | rna | aa | na | other-mol) "dna"
length CDATA #IMPLIED
```

This means that **molecule** is an attribute that can only take on a limited set of values (a **controlled vocabulary**) and that the default value is **dna**. (The notation **(x|y|z)** means one of the members of the set of **x**, **y,** and **z**.)

The **length** attribute may theoretically be any text (**CDATA**=character data), but in practice must be an integer that represents the number of residues.

### 2.1.2 Representing Sequence Characteristics

In practice, we usually want to know more about a sequence than its molecule type and length, especially in the case of DNA.

**DNA strands and topology.** DNA is typically characterized as single- (**ss**) or double-stranded (**ds**), and its shape is typically characterized as **linear** or **circular**. BSML provides two attributes for presenting this information:

```
strand (std-not-set | ss | ds | mixed | std-other) #IMPLIED
topology (top-not-set | linear | circular | tandem | top-other)"linear"
```

Thus a double-stranded linear DNA fragment might be described as:

```
<Sequence molecule="dna" length="1000" strand="ds" topology="linear">
```

**Ends of Linear Double-stranded DNA.** If a DNA fragment is the result of a sequencing project, we are not likely to be concerned with characterizing the ends of the fragment, regardless of the procedure (PCR, restriction) used to manipulate the fragment during sequencing. On the other hand, if the fragment is the product of a cloning procedure, characterizing the ends of the fragment may be very important. BSML provides a set of tags for describing the 5' overhang on each end and the phosphorylization state of each end:

```
end5hang5 %integer; #IMPLIED
end3hang5 %integer; #IMPLIED
end5phos %yesorno; #IMPLIED
end3phos %yesorno; #IMPLIED
```

**Note:** These attribute definitions use **entities** that tell us how to interpret the text value of the attribute. Thus **%integer;** is defined to indicate that the text must be resolved to an integer value (positive or negative). Similarly, **%yesorno;** is defined to mean that **"1"** indicates "yes" (logical **true**) and **"0"** indicates "no" (logical **false**).

If a fragment has been cut at its 5' end by a blunt cutting restriction enzyme, that end has been kinased, and if the other end was produced by cutting with EcoRI, the fragment might be described as:

```
<Sequence molecule="dna" length="1000"
          end5hang5="0" end5phos="0" end3hang5="4" end3phos="1">
```

In this case, the **length** of **1000** refers to the length of the coding strand.

**Sequence representation.** In some cases, we are simply interested in stating that a particular sequence exists, without describing its sequence data. Such a representation may be defined as **virtual**. In other cases we wish to explicitly provide the sequence data as part of the definition of this sequence – a representation termed **raw**. Following the model of NCBI, BSML provides the following controlled vocabulary for this attribute:

```
representation (not-set | virtual | raw | segmented | constructed |
                reference | consensus | map | derived | other) "raw"
```

Thus a sequence representation that contains protein sequence data might be encoded as:

```
<Sequence molecule="aa" length="137" representation="raw">
```

**BSML 3.1 – new sequence attributes.** BSML 3.1 introduced several new attributes for describing sequences. **genomeref** is an attribute of type **IDREF** that may be used to point to a genome-level element (see below). **trans-table** names the translation table to be used with a DNA or RNA sequence (names are defined by NCBI). **segmenttype** indicates the type of segment represented by the sequence element. **dnatype** indicates whether a DNA sequence is **genomic** or **cdna**.

```
genomeref IDREF #IMPLIED
trans-table CDATA #IMPLIED
segmenttype (chromosome|chromosomal-contig|clone|
             sequence-contig|sequence|other) "sequence"
dnatype (genomic | cdna) #IMPLIED
```

For example:

```
<Sequence molecule="dna" length="1000" representation="raw"
          trans-table="human-mitrochondria" segmenttype="contig"
          dnatype="genomic" genomeref="GENOME1">
```

### 2.1.3 Sequence Identification

Beyond the description of a sequence, we typically want to **identify** the sequence. Identification may mean several things.

**Sequence title.** BSML provides a number of attributes that apply to most elements, including **title** and **comment**. The **title** attribute generally contains the *displayable* name of an element: the text that is shown to a user to identify the element. The **comment** attribute may contain additional information or a more extended version of the title. For example:

```
<Sequence title="SV40" comment="complete SV40 circular genome">
```

**Private sequence source.** One common need is to identify where a sequence came from. If the sequence was determined in a local laboratory, the identification may consist of describing the machine and sequencing run. There may well be some unique local identifier that may be captured in the following attribute:

```
local-acckey CDATA #IMPLIED
```

For example:

```
<Sequence local-acckey="SeqRun12345">
```

**Public sequence source.** Another principal source of sequences is the international consortium of public databases (GenBank, EMBL, and DDBJ). BSML provides three attributes to identify sequences from sources such as these:

```
db-source CDATA #IMPLIED
ic-acckey CDATA #IMPLIED
locus     CDATA #IMPLIED
```

For example:

```
<Sequence db-source="GenBank" ic-acckey="J02400" locus="SV4CG">
```

**Additional identification.** Although a great deal of identification is explicitly represented in the attributes defined by BSML for sequences, there always may be additional descriptions that the BSML creator wishes to associate with a sequence. The **Attribute** element is provided for this purpose. For example, the following elements might be added to a GenBank description:

```
<Attribute name="version" content="J02400.1  GI:965480"/>
<Attribute name="source" content="Simian virus 40"/>
<Attribute name="organism" content="Simian virus 40"/>
```

**XML element identification.** In the world of XML, an element may be identified by an attribute that is designated as its unique identifier, which means that the element must be uniquely identified in the XML document containing it. In the case of BSML, this attribute is the **id** attribute, which may be defined for almost all BSML elements. We may choose to use a unique public identifier (e. g., an EMBL accession key) as the **id**, for example:

```
<Sequence id="AB003468" db-source="EMBL" ic-acckey="AB003468">
```

Alternatively, we may wish to use an identifier that is unique to a private library of sequences, even if the source of the sequence is a public database:

```
<Sequence id="LB001SQ12345" db-source="EMBL" ic-acckey="AB003468">
```

In this example, the identifier **LB001SQ12345** might represent the 12,345 assigned sequence identifier at site LB001.

If a sequence in a BSML document is a fragement of a chromosome from a particular genome build, it is recommended that a consisting identification system be used for naming the chromosomes and the sequences. For example, the notation **database..build.start.length** may be be used for single chromosome genomes, and the **notation database.build.chromosome.start.length** may be used for multichromosomal genomes.

```
<Genome title="UCSC">
  <Chromosome title="1" id="UCSC.2002.10.C1" … >
  …
</Genome>
<Sequence title="UCSC Chromosome 1, bp 200000001 to 200000500"
                  id="UCSC.2002.10.C1.200000001.500" … >
```

**Sequence positional context and identification.** To be more explicit about the position of a sequence on a genome, we may use the **Numbering** element to relate a particular sequence to a larger context.

Internally, the length of every sequence is expressed as a positive integer. Sequences and other sequence-related elements may specify their unit of measurement (bp, cM, etc.) through the use of a **Numbering** element, which also may define a relative numbering basis or a linear transformation rule for converting internal sequence positions to display values. By using various reference attributes, a **Numbering** element may indicate the location of a sequence in a larger context, e.g., the location of a sequence on a clone or the location of a clone on a chromosome. By chained references, a sequence may be placed in more than one positional context. The following example illustrates how a sequence (**id='SEQ1"**) may be located on the complementary strand of a clone (**id="CLONE1"**), which is, in turn, located on the coding strand of a chromosome (**id="CHR1"**), thus establishing the positional context of the sequence (note that BSML 3.1 includes the **seqref** attribute to point to a reference sequence; the **refs** attribute was used in BSML 2.2):

```
<Sequence title="Chromosome 1" id="CHR1"
          representation="virtual" length="253000000">
</Sequence>
<Sequence title="Clone 1" id="CLONE1"
          representation="virtual" length="200000" seqref="CHR1">
  <Numbering type="continuous" refnum="143000001" ascending="1"/>
</Sequence>
<Sequence title="Sequence 1" id="SEQ1"
          representation="raw" length="50" seqref="CLONE1">
  <Numbering type="continuous" refnum="123456" ascending="0"/>
</Sequence>
```

### 2.1.4 Sequence Segments of Interest

A segment of interest is any portion of a sequence, identified by its starting position (on the coding strand in the case of a double stranded nucleotide sequence), length, and, if

applicable, strand. For example, a region around a gene might be a segment of interest for the selection of PCR primers.

To provide persistent representation of segments of interest, the **Segment-set** element may be used, with each **Segment** within a set referring to a segment of interest. The **seg-id** attribute of a **Segment** indicates the **Sequence** for which the region is defined. If a number of **Segment** elements are used for the same **Sequence**, it is recommended that all **Segment** elements in a **Segment-set** refer to the same **Sequence** and the **refs** attribute of the **Sequence** may then be used to point to the **id** of the **Segment-set**.

```
<Sequence id="user.SQ20020801153012" refs="user.SS20020801154514" … >
 ...

<Segment-set id="user.SS20020801154514" seg-set-type="segment">

  <Segment title="GEN1 gene region"  id="user.SG20020801154515"
    seg-source-type="Sequence"
    seg-id="user.SQ20020801153012"  seg-role="segment"
    seg-start="1001" seg-end="2000" seg-on-complement="0">

  <Segment title="GEN2 gene region"  id="user.SG20020801154516"
    seg-source-type="Sequence"
    seg-id="user.SQ20020801153012"  seg-role="segment"
    seg-start="3001" seg-end="4000" seg-on-complement="1">

</Segment-set>
```

This usage links the **Sequence** to the **Segment-set** by pointing the **refs** attribute to the **id** of the **Segment-set**. Each **Segment** in the set points to the **Sequence** by its **seg-id** attribute and may be referred to by its own **id**. If a segment of interest is sent to an analysis and identified by its identifier, that identifier may then be used to locate the relative position of the fragment on the sequence and to annotate the sequence with the results of the analysis.

### 2.1.5 General Segment-set Uses

The **Segment-set** is defined the content model for **Sequences** in BSML 3.1. **Segment-set** and **Segment** elements may be used generally to represent relationships among the segments of the same or different sequences. The following notation is recommended.

Each **Segment-set** may contain any number of **Segment** elements, each of which describes a region of a sequence of chromosome. These elements may be used to define the equivalence of **Sequence** element regions to another sequence region or a region of a chromosome. Note that the two regions do not have to be identical in size.

**Segment Use Cases**. In each use case, the **seg-set-ty**pe attribute is set to a value to indicate the type of equivalence. The following values are used:

| | |
|---|---|
| segment | a region of interest on one sequence |

| | |
|---|---|
| equivalent | two regions are equivalent (no other conditions implied) |
| copy | all sequences are complete or region copies of each other |
| translated | one sequence is a translated region of the other |
| expressed | one sequence is a cdna derived from the other, genomic sequence |
| gapped | one sequence is a gapped (aligned) version of the other |
| aligned | the sequences are aligned (pairwise or multiple) regions |

**Simple Sequence Equivalence.** This example defines equivalence between bases 251-750 of the first sequence and bases 1-500 of the second sequence:

```
<Sequences>
  <Sequence title="seq1" id="LB101SQ0000001"
            molecule="dna" length="1000">
  </Sequence>
  <Sequence title="seq2" id="LB101SQ0000002"
            molecule="dna" length="500">
  </Sequence>
  <Segment-set seg-set-type="equivalent">
    <Segment seg-source-type="source" seg-id="LB101SQ0000001"
             seg-start="251" seg-end="750" seg-on-complement="0">
    </Segment>
    <Segment seg-source-type="source" seg-id="LB101SQ0000002"
             seg-start="1" seg-end="500" seg-on-complement="0">
    </Segment>
  <Segment-set>
</Sequences>
```

**Derived Sequence Equivalence.** This example indicates that the second sequence was created by copying a region of the first sequence (bases 251-750).

```
<Sequences>
  <Sequence title="original" id="LB101SQ0000001"
            molecule="dna" length="1000">
  </Sequence>
  <Sequence title="copy" id="LB101SQ0000002"
            molecule="dna" length="500">
  </Sequence>
  <Segment-set seg-set-type="copy">
    <Segment seg-source-type="source" seg-id="LB101SQ0000001"
             seg-start="251" seg-end="750" seg-on-complement="0">
    </Segment>
    <Segment seg-source-type="copy" seg-id="LB101SQ0000002"
             seg-start="1" seg-end="500" seg-on-complement="0">
    </Segment>
  <Segment-set>
</Sequences>
```

**Translated Sequence Equivalence.** This example indicates that the second sequence was created as the translation of a region of the first, using the complementary strand.

```
<Sequences>
```

```
  <Sequence title="original" id="LB101SQ0000001"
            molecule="dna" length="900">
  </Sequence>
  <Sequence title="translation" id="LB101SQ0000002"
            molecule="aa" length="200">
  </Sequence>
  <Segment-set seg-set-type="translated">
    <Segment seg-source-type="source" seg-id="LB101SQ0000001"
             seg-start="201" seg-end="800" seg-on-complement="1">
    </Segment>
    <Segment seg-source-type="translation" seg-id="LB101SQ0000002"
             seg-start="1" seg-end="200">
    </Segment>
  <Segment-set>
</Sequences>
```

**Expressed Sequence Equivalence.** This example indicates that the first sequence is a cdna aligned to a region of a genomic DNA segment.

```
<Sequences>
  <Sequence title="cdna" id="LB101SQ0000001"
            molecule="dna" length="500">
  </Sequence>
  <Sequence title="genomicdna" id="LB101SQ0000002"
            molecule="dna" length="10000000">
  </Sequence>
  <Segment-set seg-set-type="expressed">
    <Segment seg-source-type="cdna" seg-id="LB101SQ0000001"
             seg-start="1" seg-end="500" seg-on-complement="0">
    </Segment>
    <Segment seg-source-type="genomicdna" seg-id="LB101SQ0000002"
             seg-start="5133201" seg-end="5148222"
             seg-on-complement="0">
    </Segment>
  <Segment-set>
</Sequences>
```

**Gapped.** This example indicates that the second sequence is gapped (i.e., has - insertions) and is otherwise equivalent to the first.

```
<Sequences>
  <Sequence title="original" id="LB101SQ0000001"
            molecule="dna" length="1000">
  </Sequence>
  <Sequence title="aligned" id="LB101SQ0000002"
            molecule="dna" length="1100">
  </Sequence>
  <Segment-set seg-set-type="gapped">
    <Segment seg-source-type="source" seg-id="LB101SQ0000001"
             seg-start="1" seg-end="1000" seg-on-complement="0">
    </Segment>
    <Segment seg-source-type="aligned" seg-id="LB101SQ0000002"
             seg-start="1" seg-end="1100" seg-on-complement="0">
    </Segment>
  <Segment-set>
```

```
</Sequences>
```

**Aligned.** This example illustrates the equivalent aligned regions of two sequences that were aligned to each other.

```
<Sequences>
  <Sequence title="seq1" id="LB101SQ0000001"
            molecule="dna" length="1000">
  </Sequence>
  <Sequence title="seq2" id="LB101SQ0000002"
            molecule="dna" length="950">
  </Sequence>
  <Segment-set seg-set-type="aligned">
    <Segment seg-source-type="aligned" seg-id="LB101SQ0000001"
             seg-start="25" seg-end="750" seg-on-complement="0">
    </Segment>
    <Segment seg-source-type="aligned" seg-id="LB101SQ0000002"
             seg-start="53" seg-end="800" seg-on-complement="0">
    </Segment>
  <Segment-set>
</Sequences>
```

**Chromosome Expressed.** This example shows the expressed region of a chromosome.

```
<Genomes>
  <Genome ...>
    <Chromosome id="USCS.CHR1" length="253000000"></Chromosome>
  </Genome>
</Genomes>
<Sequences>
  <Sequence title="original" id="LB101SQ0000001"
            molecule="dna" length="1000">
  </Sequence>
  <Segment-set seg-set-type="expressed">
    <Segment seg-source-type="cdna" seg-id="LB101SQ0000001"
             seg-start="1" seg-end="1000" seg-on-complement="0">
    </Segment>
    <Segment seg-source-type="genomicdna" seg-id="UCSC.CHR1"
             seg-start="2000001" seg-end="2002001"
             seg-on-complement="0">
    </Segment>
  <Segment-set>
</Sequences>
```

### 2.1.6 Sequence Identifiers and Sequence Instances

The same sequence (as denoted by both its data and identifiers) may be present in more than one BSML document. In fact, a sequence may be "virtual" with respect to BSML documents, existing only when it is requested from a database. To differentiate a particular instance of a sequence, which may be virtual and may include particular annotation, a URI may also be associated with the sequence identifier in references to the sequence that are outside the context of a particular BSML document.

Within an application using BSML to handle sequence content, it is important to be able to identify particular sequence instances. For example, suppose that the sequence data are are extracted and sent to an analysis. The analysis results are to be used to annotate the sequence. It is recommended that a consistent identification system be used. Many analyses accept the FASTA format, and it is commonplace to use the | separator in FASTA headers to separate fields. The recommended format for BSML sequences is:

```
> URI | id
```

where the URI may be any of the following:

- Document accession key in a document storage system
- Network URL
- Local filename

The id may be a **Sequence** element id, **Segment** id, or **genome/chromosome** identifier. For example:

```
>LB101AC00001|user.SQ20021001124521
>http://www.mysite.com/sequences|user.SG101
>c:/mysequences/seq1.bsml|UCSC.CHR1.2453452.1000
```

An alternate approach is to identify a region as part of the header and to identify fields by name:

```
>URI|c:/myseq/sq.bsml|id|user.SQ2002100|start|1001|length|500|strand|+
```

It is obviously import to use a consistent naming scheme. Note that some identifiers may be use used independently of a particular document, wherease others, unless registered in some persistent fashion, must include a URI reference.

## 2.2 Sequence Data

If the **representation** of a sequence is **raw**, the sequence data are contained within the **Sequence** element, either directly or by reference to an external file containing the sequence data.

**Direct inclusion of sequence data.** Sequence data are included directly using the content of a **Seq-data** element. For example:

```
<Sequence molecule="aa" length="8">
  <Seq-data>
    akffglkl
  </Seq-data>
</Sequence>
```

The data are represented by single-character IUPAC codes and the representation is not case-sensitive (e.g., **AKFFGLKL**=**akffglkl**). Any whitespace (blanks, etc.) and

characters not included in the IUPAC alphabets for amino acids and nucleotides are ignored; thus the following is valid:

```
<Sequence molecule="dna" length="100">
  <Seq-data>
     1 acgtacgtac gtacgtactt acgtcgcgaa cgccgtaact aaaacccctt
    51 acgtacgtac acgtacgtac acgtacgtac acgtacgtac ggatatcgtc
  </Seq-data>
</Sequence>
```

**External file storage of sequence data.** The **Seq-data-import** element is used to refer to an external URL that may be accessed to obtain the sequence data. In addition to an ASCII text file containing IUPAC codes that may be interspersed with whitespace and non-sequence characters, the **format** may be one of the following:

```
  IUPACna: IUPAC 1 letter codes, no spaces
  IUPACaa: IUPAC 1 letter codes, no spaces
  NCBI2na: 00=A, 01=C, 10=G, 11=T (4 bases per byte)
  NCBI4na: 1 bit each for agct (2 bases per byte)
             0001=A, 0010=C, 0100=G, 1000=T/U
             0101=Purine, 1010=Pyrimidine, etc.
```

For example:

```
<Seq-data-import format="IUPACaa"
  source="c:\labbook\genomicbrowser\library\lib1\query.fcgi?id=sv40"
</Seq-data-import>
```

If the sequence data comprise only part of the external file, the **start-pos** and **length** attributes may be used to specify the start location (byte) in the file and the length of the segment to be read. For example, the following gets the sequence data from **myseq.8na** and then extracts the 500 bases beginning at position 1001:

```
 <Seq-data-import format="IUPACaa"
  source="c:\labbook\library\myseq.8na" start-pos="1001" length="500"
</Seq-data-import>
```

## 2.3 Sequence Annotation

The term "sequence annotation" is not rigorously defined within the bioinformatics community. In BSML, the term is used to refer to a variety of elements that provide information about a sequence beyond the sequence data. Some annotation refers to the sequence as a whole or to characteristics that are not associated with any particular region of the sequence. This annotation is considered **nonpositional**. Other annotation refers explicitly to a region of the complete sequence and is considered **positional**.

### 2.3.1 Positional Annotation

**Features.** Positional annotation in BSML is handled mainly through a **Feature-table** element, which is contained within **Feature-tables** and may contain any number of **Feature** elements:

```
<Sequence ...>
  <Feature-tables>
    <Feature-table>
      <Feature>...</Feature>
      <Feature>...</Feature>
    </Feature-table>
  </Feature-tables>
</Sequence>
```

The **Feature** element itself defines the position of a feature (with an **Interval-loc** or **Point-loc** element) and may add a variety of information in the form of **Qualifier** elements. For example:

```
<Feature id="FTR4" title="Leucine TNRA" class="GENE">
 <Qualifier value-type="gene"/>
 <Interval-loc startpos="1513" endpos="1962" complement="0"/>
</Feature>
```

The purpose of allowing many **Feature-table** elements is to allow features to be grouped so that **Feature-table** elements may be selected for particular views of a sequence. Thus, the following organization might be used:

```
<Sequence>
  <Feature-tables>
    <Feature-table title="CpG Islands" class="CPG">
    …
    </Feature-table>
    <Feature-table title="Mouse homologies" class="MOUSE">
    …
    </Feature-table>
  </Feature-tables>
</Sequence>
```

#### 2.3.1.1 Positional Context for Features

The numbering used for features in **Interval-loc** or **Point-loc** elements is defined relative to the start of the **Sequence** on which these features are located. The following point is defined as the 10th base of the sequence relative to base **1** of this particular sequence.

```
<Point-loc sitepos="10"/>
```

#### 2.3.1.2 Numbering Basis

Of course, a particular feature may be viewed in a larger context. Base "1" on a fragment prepared by PCR from a complete genome has little meaning. As noted above, **Numbering** elements and **Sequence** attributes may be used to place a sequence in the larger context of a clone or chromosome. This context is transferred automatically to the

features on the sequence, so that these features may be referenced relative to the start of the fragment, the start of the clone, or the start of the chromosome.

### 2.3.2 Qualitative Annotation

Qualitative annotation typically means asserting properties about a **Feature** in the form of **Qualifiers**. In BSML, the **value-type** attribute is used to indicate the type of a qualifier, and the **value** attribute is used to indicate the specific content of the qualifier. For example, a **Qualifier** element may be used to indicate the translation of a region:

```
<Feature title="EXON 1" class="EXON">
  <Qualifier value-type="translation"
  value="MSIQHFRVALIPFFAAFCLPVFAHPETLVKVKDAEDQLGARVGYIELDLNSGKILESFR"/>
</Feature>
```

### 2.3.3 Quantitative Annotation

Quantitative annotation may be supplied in a variety of ways in BSML.

#### 2.3.3.1 Quantitative Annotation and Qualifier Value

The **value** attribute may be used to convey a quantitative index that is interpreted in terms of the **value-type** attribute. For example:

```
<Feature id="X12345" class="BLAST">
  <Interval-loc startpos="100" endpos="200"/>
  <Qualifier value-type="score" value="380"/>
</Feature>
```

### 2.3.4 Referential Annotation

Referential annotation refers to the use of linking and cross-reference elements to associate a sequence or its features with an external object. The external object may be an entry in a public database, a graphic image, a text, etc. Referential annotation may also be used to place a biological object in the context of an ontology that defines such characteristics as function and cellular location.

#### 2.3.4.1 Cross-reference Referential Annotation

A cross-reference may be added to a **Feature** by adding a **Qualifier** with the **value-type** attribute set to **db_xref** and the **value** attribute set to the database and identifier, as follows:

```
<Feature title="TetR" class="GENE">
  <Qualifier value-type="db_xref" value="GenBank:AB003468"/>
</Feature>
```

**Note:** The processing of such a cross-reference is software-dependent and depends on the interpretation of the **database:identifier** pairs. For example, the Genomic XML Viewer currently supports the following databases:

| | | | | | |
|---|---|---|---|---|---|
| **GENBANK** | **GI** | **NID** | **PID** | **PIR** | **SP** |
| **UNIGENE** | **OMIM** | **MIM** | **LOCUSLINK** | **LOCUS** | **PFAM** |
| **REFSEQ** | **EMBL** | **SWISS-PROT** | **SPTREMBL** | **HUGO** | **PDB** |
| **PRF** | **ENSEMBL** | **SNP** | **MEDLINE** | **PMID** | **UCSC** |
| **KEGG** | **MapViewer** | **SGD** | **WormDB** | | |

#### 2.3.4.2 Link-based Referential Annotation

The various linking elements may be used to annotate by reference to a resource contained in the current document or elsewhere. For example:

```
<View ...>
 <Link title="sv40" role="virus source"
  href="http://www.ncbi.nlm.nih.org/entrez/viewer.cgi?id=sv4cg">
```

## 2.4 Additional Sequence Descriptors: Genomes and Isoforms

BSML 3.1 offers a large set of new elements for describing sequence-related content, in addition to the **Sequence** element provided by BSML 2.2.

### 2.4.1 Genomes

Generally, a genome will be described by reference to a "build" or sequence project. To do so, **Resource** elements are used to provide the appropriate metadata (see below). For example, a description of a build of the human genome might be begin as:

```
<Definitions>
  <Genomes>
   <Genome title="human" autosomal-chromosome-count="22"
    sex-chromosome-count="2" ploidy-count="2"
    distinct-chromosome-count="24" total-chromosome-count="46">
    <Organism genus="Homo" species="sapiens" taxon-num="9660">
    </Organism>
    <Resource title="Genome build">
     <Version title="UCSC Golden Path"
      description="bug fix for Aug 2001 release">
      <Date datetime="2001-10" role="created"/>
     </Version>
    </Resource>
    ...
```

The description might then include chromosomes, and the descriptions of chromosomes might include cytobands (see the ***BSML Reference Manual*** for further details).

```
    <Chromosome name="1" number="1" circular="0" autosomal="1"
                length="287033928">
```

```
<Cytoband band-name="p36.33" major-band="p36" minor-band="33"
 band-color="gneg" band-type="euchromatin" band-start-pos="0"
 band-end-pos="2100000"/>
 ...
```

### 2.4.2 Isoforms

**Isoform** elements are used to define particular changes in sequences related to alleles, SNPs, and mutations. To allow for various ways of specifying the position of a change, the attribute **location**, defined as character data, is used rather than an integer positional reference. This allows positions to be specified in general ways and in any relevant manner (e.g., in a cytoband). Another element in this section, **Case**, is used to refer to a set of changes associated with a particular organism (patient, clone, etc.). The **Phenotype** element is used to describe phenotypes associated with particular isoforms or cases. The **Pedigree-set** element contains references to any number of **Pedigree** elements. These may group **cases** by lineage and indicate standard pedigree information. For more details, refer to the ***BSML Reference Manual***.

The following example encodes a mutation located at position 1001 on the sequence pointed to by the **seqref** IDREF attribute ("SEQ1").

```
<Isoforms>
  <Isoform-set>
    <Isoform id="ISO1001" seqref="SEQ1"
             title="mutation 168B"
             location-type="physical" location="1001"
             change="cac" replaces "ctc">
    </Isoform>
  </Isoform-set>
</Isoforms>
```

A set of mutations may be associated with a **Case** (patient, cell-line, etc.) by using the following encoding, which relates the case to an isoform with the ID "ISO1001".

```
<Isoforms>
  <Case-set>
    <Case title="John Doe" isoforms="ISO1001"></Case>
  <Case-set>
</Isoforms>
```

## 2.5 Representing Data Tables

BSML 3.1 provides a number of elements in the **Tables** section under **Definitions**. Some of these elements (**Table**, **Motif-table**, **Seq-pair-alignment, PCR-primer-table**, **Alignment-point-set**, and **Multiple-alignment-table**) provide encodings of tabular data that are completely contained in the BSML document. Other elements are used to refer to external data tables (**Sequence-search-table** and **Table-import**). Some of these elements

provide attributes that are **IDREF** pointers to the sequence being analyzed. In other cases, **Link** elements may be used to point to data sources, as appropriate. The following encoding represents an external data file containing hydrophobicity data associated with a sequence in the current document. A **Link** element is used in the **Table-import** element to point to the sequence used in the analysis.

```
<Bsml>
 <Definitions>
  <Sequences>
   <Sequence id="SEQ1" title="myseq" …>
   …
   </Sequence>
  </Sequences>
  <Tables>
   <Table-import source="c:\mydata\hydro.dat"
                 title="myseq hydrophobicity">
    <Link title="myseq" href="#SEQ1"/>
   </Table-import>
  </Tables>
```

## 2.6 Describing Research

The content of a data table is often created by applying software to generate queries, searches, or analyses. The conditions of the research are obviously important in the interpretation of the data. For this reason, BSML 3.1 introduces a new high-level element named **Research** that contains descriptions of searches, queries, analyses, and experiments (refer to the ***BSML Reference Manual*** for complete details).

These elements are often used in combination with a table and sequence to provide a set of related elements. For example, suppose that a BLAST search result set is stored in an external data file (**url** below for **Sequence-search-table**).

```
<Bsml>
 <Definitions>
  <Sequences>
   <Sequence id="SEQ1" title="myseq" …>
   …
   </Sequence>
  </Sequences>
  <Tables>
   <Sequence-search-table search-type="BLAST"
      url="c:\mydata\myblast.xml" analysis-title="BLAST against nr"
      analysisref="ANA1"
      queryseqref="SEQ1">
  </Sequence-search-table>
  </Tables>
 </Definitions>
 <Research>
  <Analyses>
   <Analysis id="ANA1" title="BLAST against nr for myseq">
    <Parameter name="matrix" value="pam250"/>
     ...
```

```
    </Analysis>
  </Analyses>
 </Research>
</Bsml>
```

This encoding links to the external data table using the **Sequence-search-table** element. This element, in turn, links the analysis to the sequence by its **queryseqref** attribute and to the analysis description by its **analysisref** attribute.

## 2.7 Biological Objects and Graphical Objects

BSML provides alternative ways for representing and presenting biological content. This section discusses the data model of BSML as it pertains to biological objects and graphical objects and the relationship between the two.

### 2.7.1 Biological Object

A biological object may be thought of as an abstraction representing some reasonably well-defined biological entity or concept that is encoded in some form (relational database schema, XML schema, UML notation, etc.). In BSML, for example, the **Sequence** element and all of its attributes and child elements represent the notion of a **sequence** as a biological object. The element definition represents the abstract definition of the object and defines a subset of all properties that are potentially of interest. An instance of the element in a BSML document represents an instance of the object, with values supplied for the defining attributes (shape, length, etc.).

#### 2.7.1.1 Individual Objects

Some biological objects may be considered individually, without reference to other objects. For example, an RNA molecule may be considered as a separate **Sequence** entity.

#### 2.7.1.2 Sets of Objects

Other biological objects are best considered as sets. For example, the cleavage sites for a restriction enzyme on a particular sequence are usually considered as a set. The sequence fragments produced by a digest with the restriction enzyme are defined by the set of cleavage sites, even though a researcher may only be interested in one fragment. For similar reasons, sets of features (e.g., exons) have a collective meaning that is not available from any particular member of the set.

### 2.7.2 Graphical Object

A graphical object may be considered at several levels of abstraction: the pixels turned on or off on a display, the drawing primitives (line, arc, etc.) used to render the object, etc. In BSML, each graphical object is termed a **display widget**. Rather than emphasizing the drawing routines, each display widget is defined in terms of its **function**.

Some of the functions represented by display widgets are not particularly biological. For example, a **Caption-widget** element represents text to be displayed on the page. Similarly, **Line-pointer** and **Shape** widgets provide graphical objects that are not particularly biological.

### 2.7.3 Biological Object as a Graphical Object

The vast majority of the display widgets defined in BSML are graphical objects with biological meaning. In many cases, the display widget represents a **graphical metaphor** that is commonly used by molecular biologists to describe their subject matter. For example, a **View** element depicts a biological sequence. Together with its children elements (**Interval-widget**, **Point-widget**, etc.), the **View** provides a graphical representation of an annotated sequence.

In other cases, the display widget represents an abstract rendering of a research product such as an electrophoresis gel image (**Gel-plot-widget**). Still other widgets represent data in commonly accepted display formats (e.g., **Dot-plot-widget**).

In most cases, underlying biological data are not encoded directly into the graphical widget. Instead, the data are associated with the object by reference to elements in the **Definitions** portion of the document. This method provides for clean separation between graphical objects, contained in the **Display** section of a BSML document, and the data elements contained in the **Definitions** section. For example, a **View** element uses its **seqref** attribute to refer to a **Sequence** element.

### 2.7.4 Encoding and Rendering

The representation of graphical objects may be specified in terms of a number of attributes controlling such display characteristics as line widths, colors, and text fonts. When these attributes are specified, the **author** of the BSML document has control over how the graphical object appears when rendered with suitable software. If these attributes are left unspecified (all are optional), the rendering will be determined by the software using default or style sheet assignments provided by the **recipient**.

### 2.7.5 Graphical Objects and Rendering

There are certain assumptions made in the encoding methods defined by BSML for graphical widgets, but they are not too restrictive. It should be noted that there are no constraints that make the rendering of BSML content specific to any particular language, machine, or operating system. Any rendering agent that can implement basic text and drawing routines can render BSML graphical widgets.

#### 2.7.5.1 Paged vs. Unpaged Media

BSML is geared toward paged media, but there is no stipulation of page size, so an arbitrarily long page could be used.

#### 2.7.5.2 Objects on a Page

The graphical display objects (widgets) are all nested under a **Page** element. Some elements are nested within others (for example, a **View** element may contain an **Aligned-chart-widget**).

#### 2.7.5.3 Viewers and Data

Because BSML is able to represent a wide range of data types (hierarchies, sets, tables, textual, numeric, etc.), appropriate rendering software ("viewers") must be associated with each data type (e.g., a BLAST result viewer).

## 2.8 Annotating with Metadata

### 2.8.1 Background

The content of a BSML document may be associated with an author (or any other information) by adding **Attribute** elements to most elements that might require such identification (entire document, particular sequence, etc.). In addition, some attributes are specific to an element and define properties of it (e.g., the **dbsource** attribute of a **Sequence** element indicates the source of the sequence, such as GenBank).

### 2.8.2 Resources and Metadata

BSML 3.1 introduces the **Resource** element, which is included in the content models of many data and research elements. The **Resource** element includes in its content model the fifteen elements defined for metadata by the **Dublin Core Metadata Initiative**:

http://dublincore.org

The **Resource** element has a content model that allows this element to be used quite flexibly to describe a wide variety of resource types (refer to the ***BSML Reference Manual*** for complete details).

```
<!ELEMENT Resource (Attribute | Coverage | Description | Type |
Relation | Source | Subject | Title | Contributor | Creator | Publisher
| Rights | Date | Format | Identifier | Language | Content | Version |
History | Authority | %links;)* )>
```

By itself, a **Resource** element may be used to designate a remote source of information, using the **url** attribute, much like a **Link** element.

```
<Sequence ...
 <Resource title="Analysis Center" url="http://www.analysiscenter.com">
</Sequence>
```

In conjunction with the children in its content model, the **Resource** element may be used to describe authorship, content, and many other characteristics of other content. For example, say that a new **Feature** element is added to a feature table. The following encoding would indicate the author:

```
<Sequence ...>
 <Feature-tables>
  <Feature-table>
   <Feature ...>
   </Feature>
    ...
   <Feature id="FTR1" title="new feature">
    <Interval-loc start-pos="1001" end-pos="2000"/>
    <Resource>
     <Contributor>
      <Person fullname="Charles Darwin"></Person>
     </Contributor>
    </Resource>
   </Feature>
  </Feature-table>
 </Feature-tables>
</Sequence>
```

### 2.8.3 Metadata "Inheritance"

The intent of the design of BSML 3.1 is that rules of inheritance be applied to metadata **Resource** elements. For example, if a **Resource** indicating authorship is attached to a **Feature-table** element, it is assumed that the metadata applies to all **Feature** elements in that table unless they have their own tags. If no metadata tags are indicated, then it is assumed that other content may be interpreted (e.g., **dbsource="GENBANK"** would indicate that all of the sequence content is derived from GenBank if no other information is available). Strictly speaking, it is up to the rendering software to "enforce" any effects of metadata location in the document.

### 2.8.4 Versions and Date-stamps

In addition to adding authorship information, it is often useful to add other metadata. Note that a **Resoure** element may have any number of child elements and may have any number of repetitions of an element. A common use of metadata is to indicate version and date. For example:

```
<Feature …>
 <Resource>
  <Creator>
   <Software-system name="genemaker">
    <Version full-version="6.8">
     <Date role="issued" datetime="2002-01-01">
```

```
    </Version>
   </Software-system>
  </Creator>
 </Resource>
</Feature>
```

### 2.8.5 Literature and Patent Citations

The Resource element may be used to make reference to literature citations (books, journal articles, patents, etc.). For example:

```
<Resource>
 <Title>Advances in Biotechnology</Title>
 <Creator>
   <Person lastname="Doe" firstname="John"></Person>
 </Creator>
 <Type>book</Type>
 <Subject>cloning,biotechnology</Subject>
 <Publisher>
   <Organization name="LabBook Press">
   </Organization>
 </Publisher>
 <Date datetime="2001-02-08"/>
 <Identifier context="ISBN" refid="123456789"></Identifier>
 <Content role="abstract">
  Excellent source on new technology
 </Content>
</Resource>
```

# 3. Why Use BSML?

## 3.1 Advantages of XML in General

In its short lifetime XML has proven to be very useful for representing data, exchanging data between environments and applications, and transporting data over distributed networks. XML is particularly useful for hierarchically structured data, and XML linking mechanisms support network data representations.

## 3.2 Advantages of BSML in Particular

BSML takes advantage of XML features for encoding hierarchically organized information (e.g., sequence - feature table - feature - qualifier hierarchy). By providing an explicit representation of knowledge in a particular domain – biological sequences – BSML capitalizes on the general strengths of XML and the domain knowledge that BSML represents.

## 3.3 BSML in Relation to XML and Non-XML Alternatives

There are various ways to encode hierarchically organized content, including object-oriented databases, ASN.1, SGML, etc. There is probably no single solution that is optimal for all purposes (e.g., rich representation of semantics, storage ease, queryability, bandwidth conservation, etc.). BSML is a reasonable choice under many circumstances.

## 3.4 Need for Additional Standards

BSML is not intended to be the answer to every issue of knowledge representation in the life sciences. Additional public standards are needed to represent cellular processes, gene expression, etc.

## 3.5 How BSML Interacts with Other Data Representations

To move from one data representation to another requires a **mapping** between the representations. Certain considerations apply when mapping from one type of data representation to BSML.

### 3.5.1 BSML and Relational Databases

It is generally possible to decompose a hierarchical representation of data into a relational representation. For example, there have been relational database representations of

GenBank for many years (e.g., NCGR). Mapping from a relational database to BSML is generally a matter of establishing the correspondence between various tables and their joins and the hierarchical sequence - feature table - feature - qualifier structure of BSML.

### 3.5.2 BSML and Flatfile Data

Flatfile representations of sequence information may be transformed into BSML using appropriate parsing technology. The first step is to establish a mapping between elements in one format and elements in the other. This subject is discussed in greater detail in another LabBook manual on mapping public database formats (GenBank, EMBL and Swiss-Prot) to BSML (see ***Mapping to BSML 2.2 from Public Sequence Formats***).

### 3.5.3 BSML and ASN.1

BSML and ASN.1 (Abstract Scientific Notation) are very compatible and map directly, because both are hierarchical and object oriented.

### 3.5.4 BSML and Other Sequence XML Applications

Other XML-based representations of sequence information may usually be mapped directly to BSML using XSLT transformations.

## 3.6 BSML and Software

BSML documents are generally produced by software applications that extract information from other sources.

### 3.6.1 Query Software

Query software generally uses SQL statements to access information from a relational database. The LabBook OSU Human Genome (OHGD) Query, described elsewhere, provides an example of how relational databases may be queried to produce BSML content.

### 3.6.2 Creation Software

Creation software is often tailored to particular functions. LabBook, Inc. provides a BSML Java API that includes utility classes for creating BSML documents. Additional software modules (e.g., Perl modules) are available in the public domain for converting from generic formats to BSML (see ***BSML 3.1 Java API and ToolKit***).

### 3.6.3 Editing Software

Interactive editing software is useful for fine-grain editing of BSML documents. The BSML Java API includes a simple interactive editor, and more extensive editing software is available from LabBook, Inc.

### 3.6.4 Visualization Software

Visualization software must be capable of rendering BSML content. LabBook, Inc. provides the Genomic XML Viewer and Genomic XML Browser for this purpose (see ***Rendering BSML in the Genomic XML Viewer***).

### 3.6.5 XML Generic Software

BSML is an XML application. This means that any software capable of reading and writing XML can be used to work with BSML documents (DOM parsers, SAX parsers, etc.).