

# **CSML 1.9 Parser Tutorial**

**7/Nov/2006**

**This document was edited by Masao Nagasaki**

**CSML Project Team:**

**Masao Nagasaki, Euna Jeong, Atsushi Doi, Ayumu Saito, Satoru Miyano**

**Copyright©2006 CSML Project Team. All rights reserved under the Creative Commons License**

**(<http://creativecommons.org/licenses/by/2.5/>)**

# 1. Introduction

*Cell System Markup Language* (CSML) is an XML format for modeling, visualizing and simulating biopathways. CSML supports to represent several pathway types including metabolic, signaling, and genetic regulatory pathways.

*CSML Parser* is a Relaxer-based software component, useful for reading data from XML file, manipulating them (getting and setting attributes, adding and removing elements) and writing new data into XML file. The *CSML Parser* is available on the CSML WWW site at [www.csml.org](http://www.csml.org) in the form of a jar file.

*Relaxer* is an XML data binding framework. *Relaxer XML* mapping is a way to simplify the binding of Java classes to XML document. It allows transforming the data contained in XML document into/from a Java object model. The mapping information is specified by an XML RNG schema. The hierarchical structure of the XML file is represented by tree like structure of Java objects. Each XML element is represented by one Java object, each element's attribute by Java object attribute with set of getters and setters. For more information see [www.relaxer.org](http://www.relaxer.org).

This tutorial describes how to use the *CSML Parser* to read, manipulate and write CSML 1.9 files.

## 2. Setting up

To run Java programs that access CSML 1.9 parser:

- Download the **csml1.9\_parser.jar** from [www.csml.org](http://www.csml.org).
- Set up CLASSPATH variable. CLASSPATH must include the **csml1.9\_parser.jar**.

For example:

```
CLASSPATH=.: /home/usr/lib/csml1.9_parser.jar
```

## 3. Reading files

To read CSML 1.9 data file do one of the following:

- Include the following instruction in your java code:

```
CSML19StartModel model = CSML19IO.loadCSMLModel(filename);
```

The signature for loadCSMLModel method is:

```
public static CSML19StartModel loadCSMLModel(String filename)
```

- Or add the following lines to your code:

```
CSML19SAXReader reader = new CSML19SAXReader(file);  
CSML19StartModel model = reader.getModel();
```

The signatures for CSML19SAXReader constructors and getModel method are

```
public CSML19SAXReader(String filename)  
public CSML19SAXReader(File filename)  
public CSML19StartModel getModel()
```

## 4. Manipulating data structure

The CSML1.9\_parser.jar is a component that allows for reading the CSML files and storing the CSML model in an internal data structure: Java objects. The parser code is automatically generated from the RNG schema using the Relaxer software (for more information see [www.relaxer.org](http://www.relaxer.org)).

### ***Mapping of XML Elements to Java Objects***

Each XML element is mapped to one Java class, while each XML attribute is mapped to a class attribute. The mapping of XML elements to Java objects is illustrated in the table below:

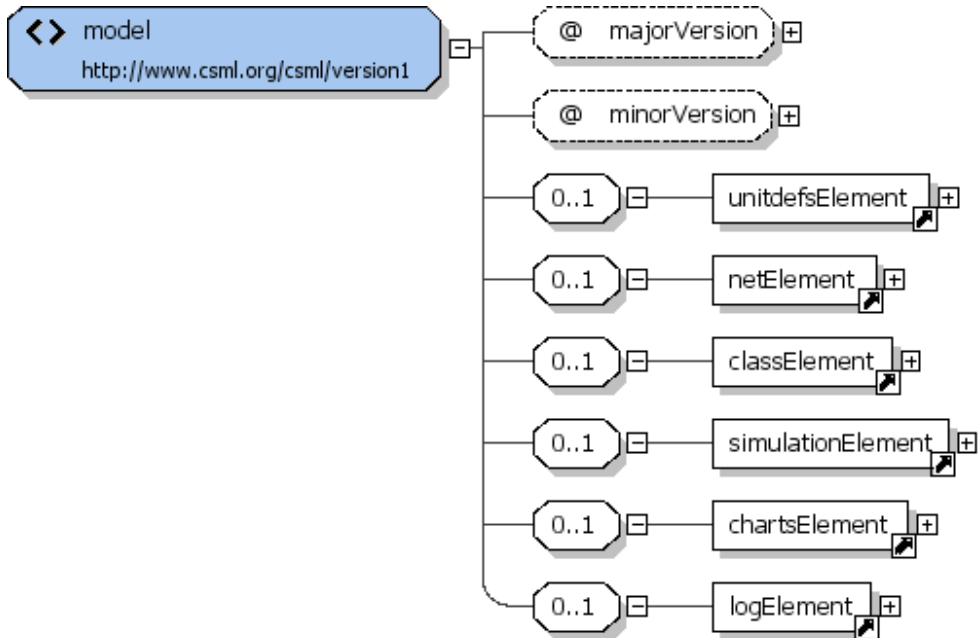
<b>CSML</b>	<b>Java Code</b>
XML Element (tag)	Class
Example:	
model	CSML19StartModel
net	CSML19NetElement
entity	CSML19EntityElement
process	CSML19ProcessElement
class	CSML19ClassElementClass

<b>CSML</b>	<b>Java Code</b>
XML Attribute	Attribute, get and set methods
Example:	
majorVersion	majorVersion_ getMajorVersion() setMajorVersion()
label	label_ getLabel() setLabel()
type	type_ getType() setType()

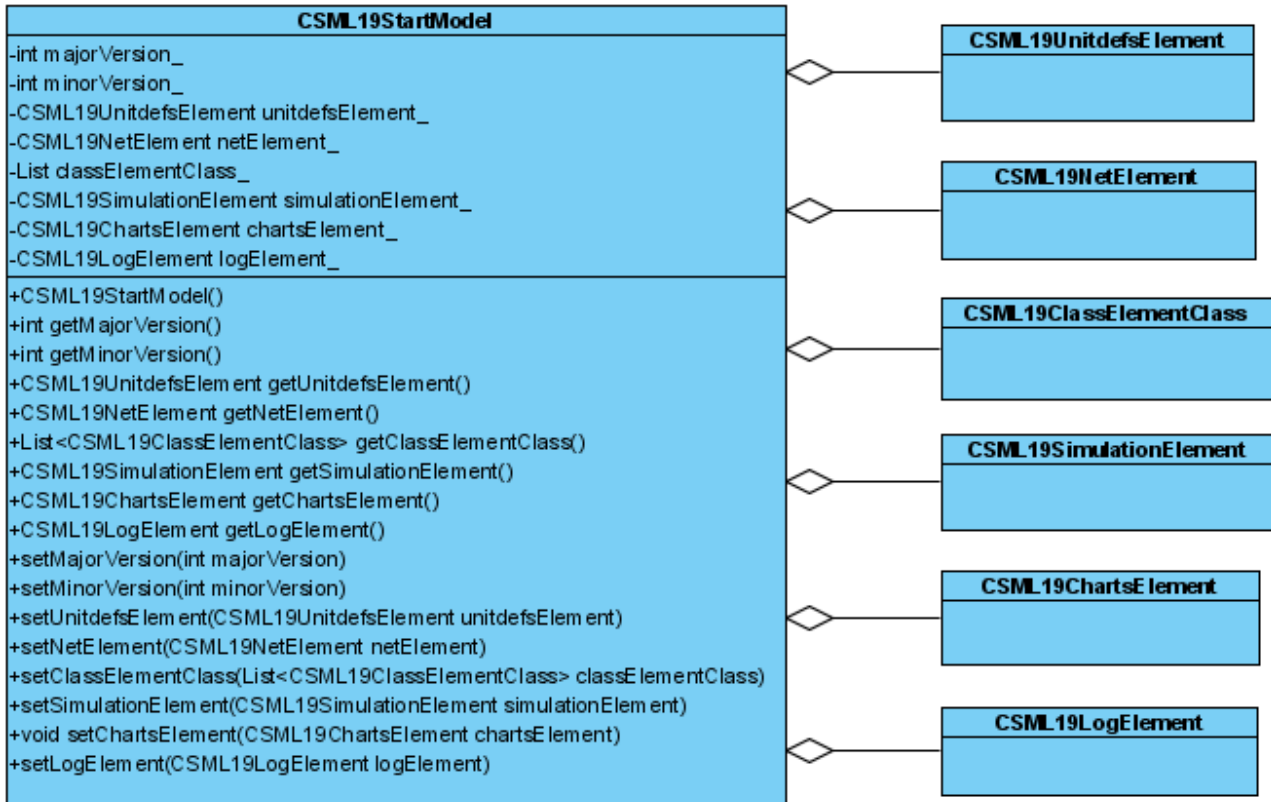
The detailed mapping of class names is specified in the CSML 1.9 RNG file at [www.csml.org](http://www.csml.org)

## XML elements hierarchy representation

The hierarchy of XML elements is illustrated on the picture below:



The internal representation of the CSML model is a tree like structure of Java objects. Each parent object includes its child objects and methods for manipulating them. The example shows the class diagram created for the <model> XML element.



## Manipulating the CSML 1.9 Model

To manipulate on CSML 1.9 model, you can use the methods starting with `get*`, `set*` and `add*` prefix. Methods starting with `get*` prefix allows you to get attributes or sub elements of the current element. Methods starting with `set*` prefix allows you to set attributes or sub elements (only singular). Methods starting with `add*` prefix allows you to add sub elements of the current element (multiple sub elements).

Sample methods in `CSML19StartModel` class:

### 1. getting attributes:

```

public int getMajorVersion()
public Integer getMajorVersionAsInteger()
public String getMajorVersionAsString()
  
```

### 2. setting attributes:

```

public void setMinorVersion(int minorVersion)
public void setMinorVersion(Integer minorVersion)
public void setMinorVersionByString(String string)
  
```

### 3. getting singular sub element:

```

public CSML19NetElement getNetElement()
public CSML19LogElement getLogElement()
public CSML19UnitdefsElement getUnitdefsElement()

```

4. setting singular sub element:

```

public void setNetElement(CSML19NetElement netElement)
public void setChartsElement(CSML19ChartsElement chartsElement)
public void setClassElementClass(CSML19ClassElementClass classElementClass)

```

5. adding multiple sub element:

```

public void addClassElementClass(CSML19ClassElementClass classElementClass)
public void addClassElementClass(CSML19ClassElementClass[] classElementClass)

```

6. getting multiple sub element:

```

public CSML19ClassElementClass[] getClassElementClass()
public CSML19ClassElementClass getClassElementClass(int index)

```

## 5. Writing files

To write CSML 1.9 data into file do one of the following:

- include the following instructions in your java code:

```
CSML19IO.saveCSMLModel(data, file)
```

The signature for save CSMLModel method is:

```
public static boolean saveCSMLModel(CSML19StartModel data, String file)
```

- Or add the following lines to your code:

```

FileOutputStream file = new FileOutputStream(filename);
Document document = data.makeDocument();
JaxpMarshaller.marshall(document, file);

```

The signatures for makeDocument and getModel methods are

```

public Document makeDocument()
public static void marshal(Document document, OutputStream ostream)

```

## 6. Sample code

```
/*
 * Created on Oct 23, 2006
 * CSML Project Team
 */
package org.csml.parser.csml19.test;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;

import org.csml.parser.csml19.CSML19SAXReader;
import org.csml.parser.csml19.JaxpMarshaller;
import org.csml.parser.csml19.base.CSML19NetElement;
import org.csml.parser.csml19.base.CSML19StartModel;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

/**
 * The sample class showing how to read, write and manipulate
 * on CSML 1.9 files.
 */
public final class ParserTest {

    /**
     * Default constructor.
     */
    private ParserTest() {
        super();
    }

    /**
     * @param args commandline arguments, expected name of input file
     * @throws SAXException thrown exception
     * @throws IOException thrown exception
     * @throws FileNotFoundException thrown exception
     * @throws ParserConfigurationException thrown exception
     * @throws TransformerException
     * @throws TransformerConfigurationException
     */
    public static void main(String[] args)
        throws FileNotFoundException, IOException, SAXException,
        ParserConfigurationException, TransformerConfigurationException,
        TransformerException {

        if (args.length < 1) {

            System.err.println(
                "usage: java org.csml.parser.csml19.test.ParserTest " +
                "filename");
            System.exit(0);
        }
    }
}
```

```

    }

    // get the input file name
    String filename = args[0];

    // read the file to the Relaxer data structure
    CSML19SAXReader reader = new CSML19SAXReader(filename);

    // get the Relaxer data structure root
    CSML19StartModel model = reader.getModel();

    // get base information about the model
    String majorVersion = model.getMajorVersionAsString();
    String minorVersion = model.getMinorVersionAsString();

    // get the net sub element (and its properties) of the model
    CSML19NetElement net = model.getNetElement();
    String label = net.getLabel();
    int nmb = net.sizeInnerNetElement();

    // print the information on the console
    System.out.println("File name: " + filename + "\n" +
        " csml major version: " + majorVersion + "\n" +
        " csml minor version: " + minorVersion + "\n" +
        " net label: " + label + "\n" +
        " size of net inner elements: " + nmb + "\n");

    // change net label

    String newLabel = "newNetName";
    net.setLabel(newLabel);

    // save of the modified data structure to the new file
    FileOutputStream ostream = new FileOutputStream(filename + ".new_.gon");
    Document document = model.makeDocument();

    JaxpMarshaller.marshal(document, ostream);

}
}

```