DTD for MBean Configuration Files

This appendix shows the full text of the DTD for the MBean configuration files used by ConfigAgent. This DTD name is xmlconfig.dtd.

xmlconfig.dtd

```
<!-- Copyright (c) 2001 by Cisco Systems, Inc. All rights reserved.
This is the document type descriptor for the com.cisco.aggbu.jmx.XmlConfig
class, which was copied with permission from the
com.mortbay.Util.XmlConfiguration class.  It allows a MBean object to be
configured by with a sequence of Set, Put and Call elements.

The XML file contains a single <XmlConfig> element containing one or more
<Configure> elements describing the configuration for a single object or
class of object.

Each object or class to be configured is defined in a <Configure> element
section. A Configure element must have either a class or a jmxname attribute
defined. MBeans to be configured are matched by both class and jmxname,
so that two sets of configuration may be applied to an object.
If a Configure element has an init element and a class attribute, then an MBean instance
of that class is initialized and registered by the #newIntances(MBeanServer)
method. If a jmxname attribute is also provided, that is used for registration
with the MBean server.

Configure elements may contain Set, Put and Call elements which are used
in order by the #configure(MBeanServer,ObjectInstance) method.
Examples of these tags and their java equivalents are:

  <Set  name="Test">value</Set>              ~  obj.setTest("value");
  <Put  name="Test">value</Put>              ~  obj.put("Test","value");
  <Call name="test"><Arg>value</Arg></Call>  ~  obj.test("value");
  <Call name="test">
    <Arg>value</Arg>
    <Call name="other"/>
  </Call>                                    ~  obj.test("value").other();

Values may be literals or objects that are created with the
New element or returned from a Call element:

  <Set  name="Test1">
    <New class="com.acme.MyClass"/>
  </Set>

  <Set  name="Test2">
```

```
    <New class="com.acme.MyClass"/>
      <Arg type="int">42</Arg>
      <Set name="something"/>
    </New>
  </Set>

Note that Call and New elements may contain Set, Put and Call elements
after any Arg elements. These nested elements are applied to the
created or returned object.

Untyped balues are matched to arguments on a best effort approach. Primative
types may be specified as element attributes and the value is treated
as a String and converted to that type.
-->


<!ENTITY % CONFIG "Set|Put|Call">
<!ENTITY % TYPE "String|char|short|byte|int|long|boolean|float|double|URL">
<!ENTITY % VALUE "#PCDATA|Call|New|SystemProperty|Array">

<!ENTITY % IDATTR "id ID #IMPLIED" >
<!ENTITY % TYPEATTR "type (%TYPE;) #IMPLIED " >
<!ENTITY % ORDERATTR "order NMTOKEN #REQUIRED" >
<!ENTITY % CLASSATTR "class NMTOKEN #IMPLIED" >
<!ENTITY % NAMEATTR "name NMTOKEN #REQUIRED" >
<!ENTITY % JMXNAMEATTR "jmxname CDATA #IMPLIED" >

<!--
XmlConfig Element.
This is the root element of the configuration file:

    <XmlConfig> <Configure>...</Configure> ... </XmlConfig>

An XmlConfig element can contain Configure elements.
-->
<!ELEMENT XmlConfig ((Instantiate|Configure)*) >
<!ATTLIST XmlConfig %IDATTR; %CLASSATTR;>


<!--
Configure Element.
This is the root element that specifies the class of object that
can be configured:

    <Configure name="domain:n=v"> ... </Configure>

A Configure element can contain an optional Init element followed
by any number of Set, Put or Call elements.
-->
<!ELEMENT Configure (%CONFIG;)* >
<!ATTLIST Configure %IDATTR; %JMXNAMEATTR; %CLASSATTR;>


<!--
Instantiate Element.
This element specifies a set of arguments to an object constructor
and an order attribute specifying when the object is to be constructed
wrt all of the other objects scheduled to be created by the ConfigAgent:

    <Instantiate order="20"> ... </Init>
-->
<!ELEMENT Instantiate (Arg*,(%CONFIG;)*)>
<!ATTLIST Instantiate %IDATTR; %ORDERATTR; %JMXNAMEATTR; class NMTOKEN #REQUIRED>
```

```
<!--
Set Element.
This element maps to a call to a set method on the current object.
The name and optional type attributes are used to select the set
method.
A Set element can contain value text and/or the value elements Call,
New and SystemProperty. If no value type is specified, then white
space is trimmed out of the value. If it contains multiple value
elements they are added as strings before being converted to any
specified type.
-->
<!ELEMENT Set ( %VALUE; )* >
<!ATTLIST Set %IDATTR; %NAMEATTR; %TYPEATTR; >


<!--
Put Element.
This element maps to a call to a put method on the current object,
which must implement the Map interface. The name attribute is used
as the put key and the optional type attribute can force the type
of the value.

A Put element can contain value text and/or the value elements Call,
New and SystemProperty. If no value type is specified, then white
space is trimmed out of the value. If it contains multiple value
elements they are added as strings before being converted to any
specified type.
-->
<!ELEMENT Put ( %VALUE; )* >
<!ATTLIST Put %IDATTR; %NAMEATTR; %TYPEATTR;>


<!--
Call Element.
This element maps to an arbitrary call to amethod on the current object,
The name attribute and Arg elements are used to select the method.

A Call element can contain a sequence of Arg elements followed by
a sequence of Set, Put and/or Call elements which act on any object
returned by the original call:

 <Call name="test"><Arg>value1</Arg><Set name="Test">Value2</Set></Call>

This is equivalent to:

 Object o2 = o1.test("value1");
 o2.setTest("value2");

-->
<!ELEMENT Call (Arg*,(%CONFIG;)*)>
<!ATTLIST Call %IDATTR; %NAMEATTR;>


<!--
Arg Element.
This element defines a positional argument for the Call element.
The optional type attribute can force the type of the value.

An Arg element can contain value text and/or the value elements Call,
New and SystemProperty. If no value type is specified, then white
space is trimmed out of the value. If it contains multiple value
elements they are added as strings before being converted to any
specified type.
-->
```

```
<!ELEMENT Arg ( %VALUE; )* >
<!ATTLIST Arg %IDATTR; %TYPEATTR; >



<!--
New Element.
This element allows the creation of a new object as part of a
value of a Set, Put or Arg element. The class attribute determines
the type of the new object and the contained Arg elements
are used to select the constructor for the new object.

A New element can contain a sequence of Arg elements followed by
a sequence of Set, Put and/or Call elements which act on the new object:

 <New class="com.acme.MyClass">
   <Arg>value1</Arg><Set name="Test">Value2</Set>
 </New>

This is equivalent to:

 Object o = new com.acme.MyClass("value1");
 o.setTest("value2");

-->
<!ELEMENT New (Arg*,(%CONFIG;)*)>
<!ATTLIST New %IDATTR; %CLASSATTR; >



<!--
System Property Element.
This element allows JVM System properties to be retrieved as
part of the value of a Set, Put or Arg element.
The name attribute specifies the property name and the optional
default argument provides a default value.

 <SystemProperty name="Test" default="value"/>

This is equivalent to:

 System.getProperty("Test","value");

-->
<!ELEMENT SystemProperty EMPTY>
<!ATTLIST SystemProperty %IDATTR; %NAMEATTR; default CDATA #IMPLIED>


<!--
Array element
Can have a class attribute to specify the base type of each object
in the array.
-->
<!ELEMENT Array (Item)* >
<!ATTLIST Array %IDATTR; %CLASSATTR;>


<!--
Item element
Only occurs inside Arrays and is identical to Arg in every way except its name
-->
<!ELEMENT Item ( %VALUE; )* >
<!ATTLIST Item %IDATTR; %TYPEATTR; >
```