

JDF Schema

1 Overview

This XML schema provides a formal definition of the XML syntax specified in the JDF Specification Version 1.0. It includes both JDF and JMF syntax, and defines the XML namespace "http://www.CIP4.org/JDFSchema_1".

It is not possible to express in an XML schema all of the syntactic constraints found in the specification. Nor is it possible to express the semantics associated with the syntactic constructs. In case of inconsistencies or incompleteness, the specification should be considered normative.

This schema can be used with schema validating XML parsers to enforce the validity of JDF and JMF documents. Additional code must be written for complete validation, since the schema does not express all of the syntactic constraints in the JDF Specification.

The schema is being used to generate the base classes in the CIP4 JDF SDK. The SDK enforces additional syntactic constraints, beyond what is represented in the XML schema.

It is possible to extend the JDF schema in other namespaces, defining new resources, messages, elements, and attributes, as described in the JDF Specification.

The schema is divided into n files:

- JDFCore.xsd: datatype definitions for abstract types and types that are used for multiple elements and group definitions

- JDFProcesses.xsd: definitions of the processes included in the JDF specification

- JDFMessages.xsd: definitions of the concrete messages included in the JDF specification

- JDF.xsd: container file that references all the others

JDF and JMF instance documents should declare the JDF namespace as follows:

```
<?xml version='1.0' encoding='utf-8' ?>
<jdf:JDF xmlns:jdf="http://www.CIP4.org/JDFSchema_1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.CIP4.org/JDFSchema_1 JDF.xsd" ...>
```

```
<?xml version='1.0' encoding='utf-8' ?>
<jdf:JMF xmlns:jdf="http://www.CIP4.org/JDFSchema_1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.CIP4.org/JDFSchema_1 JDF.xsd" ...>
```

2 Release for JDF Specification 1.0

This release for version 1.0 of the specification cannot make use of the Process and Message definitions because to the requirement for a type attribute (xsi:type). It is anticipated that this will be modified in the next point release of the specification.

3 Schema Design Patterns

The schema declares only two global elements: JDF and JMF. This ensures that the root element of a document must be either JDF or JMF. All other elements are declared as locals.

One goal of the schema was to allow elements to appear in any order. To achieve this, the following structure is used throughout the schema:

```
<sequence minOccurs="0" maxOccurs="unbounded">
  <element name="child1" type="jdf:xxx" minOccurs="0"/>
  <element name="child2" type="jdf:yyy" minOccurs="0"/>
  . . .
</sequence>
```

One side effect of this pattern is that any child element may appear any number of times. The SDK will enforce correct cardinality of the child elements (maybe).

Every complexType includes wildcards, to allow extension elements and attributes, defined in other namespaces, to be used. Extension elements and attributes can appear anywhere in a JDF or JMF document.

AttributeGroups are widely used. They allow related attributes to be grouped together and referenced from multiple locations. This technique is especially useful in defining resources.

4 Naming conventions

Elements and attributes are named exactly as in the JDF Specification. ComplexTypes, SimpleTypes, AttributeGroups, and Groups follow these naming conventions:

AttributeGroups are named <someName>Attribs. Base attribute groups, which are used only as parts of larger attribute groups, are named <someName>Attribs__.

For each resource, there is a base type <resourceName>__, which gets used to define a resource type named <resourceName>_r, a resource element type named <resourceName>_re, and a partitioned resource type <resourceName>_rp.

All message-related types are named <someName>_m.

Enumerated types are named e<someName>__.

Process types are named exactly as for the process being defined ie <processName>, in addition there is an additional process, JDFGenericProcess which is to be used for processes and combined processes which do not have a specific definition.