Issue / Document #:

2.0

Title:

Values and Encoding of Version IDs

Problem:

Construct a methodology which will allow consistent, reliable reference to specific versions of XML schemas within the JXDDS framework.

Solution:

Numbering of Versions

A version number will consist of three numbers, in the form `1st.2nd.3rd`. For example, `2.1.4`, `3.1.2`, and `14.11.22` would be valid version numbers.

The first number is the *major version number*. It will be updated coincident with large-scale revisions, such as a large number of additional participants joining the development effort, or a major technology overhaul.

The second number is the *minor version number*, and will indicate an incompatibility from a previous schema version. If we modify version `4.7.8`, and the modified schema is incompatible with `4.7.8`, then we should release it as `4.8.0`, instead.

The third number is the *compatibility number*, and indicates how the specific version fits in with other versions with respect to backwards compatibility, as described below.

At each release, the version number of a schema will be incrememented. At no time will a modified schema be released under a preexisting version number.

The first, second, and third numbers can have any integral value. Version `4.3.9` is followed by `4.3.10`.

Version numbers may skip. Should a released version have an error, or break the backwards-compatibility track, it may be removed. That version number may then be added to a list of invalid versions. The specification of such a list is outside the bounds of this document.

Compatibility between versions

We wish to be able to support backwards compatibility between schema versions. We would like to be able to issue revisions to previous versions which add elements and attributes, and which do not invalidate instances based on earlier versions of the schema.

For example, assume we have issued rapsheet schema version `7.12.3`. We wish to add an attribute to that schema, but to do nothing which would invalidate an instance of `7.12.3`. We update the schema, which we issue as version `7.12.4`. An instance which specifies version `7.12` could use version `7.12.3` or `7.12.4` interchangably. An instance which specifies `7.12.4` would not be able to use `7.12.3`.

- An instance written using schema version `A.B.C` would be verifiable with schema version `A.B.D` if `D` is greater than or equal to `C`.
- It is acceptable for software to make the substitution of schema version `A.B.D` where version `A.B.C` is specified, if and only if `D` is greater than or equal to `C`.

Representation in Schemas

Every schema is given a version number. i.e. "rapsheet 12.4".

The version number of a schema may be used to specify the location of the schema on a well-known server, but such specification is outside the bounds of this document. That should be addressed as we approach design of a reference architecture.

The version number will not appear as part of the namespace specification of the schema. Namespaces should be free of version information. This allows minimal interdependence between documents.

A schema will state its version in the usually-optional `version` attribute of the `schema` element. We will consider the `version` attribute to be mandatory for schemas defined within the JXDDS framework.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="3.7.6">
```

When a schema (the *defined* schema) imports another schema (the *imported* schema), it will specify the valid version number of the imported schema under which the defined schema was constructed. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    ...
```

```
    xmlns:person="http://bogus.gov/person"
    xmlns:dd="http://bogus.gov/dd"
    version="3.1.7">
  <xs:import
      namespace="http://bogus.gov/person"
      schemaLocation="person.xsd"
      dd:schemaVersion="3.2.1"/>
  <xs:complexType name="OfficerType">
    ...
  </xs:complexType>
</xs:schema>
```

This specifies that the schema being defined is expecting version `3.2.1` of the schema `person.xsd`.

The `dd:schemaVersion` attribute is defined in a global data dictionary schema, which must be imported, directly or indirectly, by all schemas and instances which utilize the specification. In this case, the value of the `dd:schemaVersion` attribute is simple the expected value of the version of the schema being imported.

Under backwards-compatibility rules, the software may include version `3.2.2`, `3.2.3`, etc., in place of version `3.2.1`.

Representation in Instances

Each schema instance shall specify the version of the schemas they use by setting the `dd:schemaVersion` attribute within the same element that `xsi:schemaLocation` attribute is defined. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Person
    xmlns="http://bogus.gov/person"
    xmlns:dd="http://bogus.gov/dd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://bogus.gov/person person.xsd"
    dd:schemaVersion="http://bogus.gov/person 1.3.6">
  <Name>Daniel Johnston</Name>
  <Height>5' 9"</Height>
  <Weight>252 lbs.</Weight>
  <SSN>939-23-3442</SSN>
</Person>
```

`dd:schemaVersion` is used the same way that `xsi:schemaLocation` is used, but instead of setting the filename or URI for a schema, it sets the expected version of the schema being referenced.

The `dd:schemaVersion` attribute should only be used along with the `schemaLocation` attribute. Some schemas may be imported by `person.xsd`, for example, and it would be a bad thing to specify the versions two different ways in two different places. In this case, the version of the `"http://bogus.gov/dd"` namespace, would be completely specified by `person.xsd`.

Rationale:

Requirements:

1. A schema may import one or more schemas. For example, the "rap sheet" schema may refer to the "court filing" schema and the "incident" schemas.
2. Every schema must have its own version number.
3. An instance document will refer to one or more schema namespaces within which its elements are defined.
4. An instance document will need to specify under which version of which schemas it is valid. For example, an instance document may state that it is valid under rapsheet schema version 3.2.4 and incident report schema 2.6.2.
5. A schema will need to specify under which versions of which schemas it is valid.
6. The version-number methodology will be used to support backwards compatibility between schema versions.

Comments and Discussion:

References

Inspired by `xfront.com`'s XML Schemas: Best Practice Homepage, specifically the XML Schema Versioning paper.

Last modified on $Date: 2002/08/23 22:39:53 $ by $Author: webb $ $State: Rel $
Please direct public discussion to gtri-xstf@lists.gatech.edu
Direct comments regarding the webpage and administration to gtri-xstf-editors@lists.gatech.edu