Extending Schemas 1.0

Working Draft 2001-09-11

This version:
ExtendingSchemas-1_0

Previous version:
N/A

Editor:
Paul Kiel, HR-XML, paul@hr-xml.org

Authors:
Paul Kiel, HR-XML, paul@hr-xml.org

Contributors:
Members of Technical Steering Committee

Abstract

HR-XML Consortium specifications are meant to model specific business practices. Recognizing that it cannot satisfy the needs of all implementers all the time, the need for a standard way to extend schemas becomes clear. This document is aimed to provide guidance regarding the extension of XML Schemas so that trading partners can exchange information in the real world as well as experiment with new data that could be incorporated into a future specification.

Status of this Document

This document is a draft Technical Note and is not to be referenced as a formal recommendation by any party.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Table of Contents

1 Overview

1.1 Objective

HR-XML Schemas cannot be all things to all people. In the real world of data interchange, there arises the need to include the content models that support real business transactions. In addition, how can implementers experiment with new data structures that lie outside the existing approved specifications? The recognition of the first statement and the need to answer the question give rise to the background for extending schemas.

Given that extension is a reality, how can we accommodate extensions without undermining the principle of open standards? This document is meant to provide guidance on the best practice for extending schemas. Its goal is to show:

1) Official endorsement of different methods for implementation.

2) Conventions for creating extensions to encourage consistency.

1.2 Design Requirements

All extensions to approved HR-XML standards SHOULD be done consistently across the Consortium. They MUST retain the basic values of XML, such as validation, especially in the context of any Certification process. Extension methods MUST not lead to an undermining of the open standards mission.

1.3 Scope

This document focuses on XML Schema extension methods. Where possible, references to DTD equivalent issues are included. Addressing all possible extension methods for DTDs (i.e. internal subsets) is not in scope. Not endorsed extension methods are discussed in Appendix C to indicate other methodology that was explored.

2 Supported Business Processes

2.1 Vocabulary Requirements

Extension

As discussed here, “extending” is meant to “add additional elements and attributes to an existing schema”. This is not to be confused with how Roger Costello uses it in the **XML Schemas: Best Practices** discussion; he refers to “extending” meaning adding functionality to

schema that does not currently exist. Such as being able to verify that the value of element <A> is greater than the value of element <B>. This document refers to “extending” in the former usage only.

3 DTD/Schema Design

The Technical Steering Committee has approved two methods that enable extension of HR-XML schemas without undermining an open standards mission. The “wrapper” and “ANY” techniques are explained herein. Additionally, a “Namespace” extension method was examined and rejected, details in Appendix C.

3.1 Wrapper extension method

3.1.1 Schema/DTD/Instance Elements Explained

This method consists of a schema that uses an HR-XML schema as an <include>. (A similar way to do this with DTD uses external entities.) This enables the intact use of HR-XML schemas and can be implemented without any accommodation required by the work group during the design phase. A drawback is the potential for many different root elements in the real world of multiple trading partners.

Example schema:

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
xmlns:hr="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1"
targetNamespace="BrilliantHRSoftware.com"
xmlns="BrilliantHRSoftware.com"
>
<xsd:import schemaLocation="JobPositionSeeker-1_1.xsd" namespace="http://ns.hr-xml.org/RandS/JobPositionSeeker-
1_1"/>
    <xsd:element name="MyInternalDoc">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="hr:JobPositionSeeker" />
                <xsd:element ref="SomeSpecialInfo"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="MyOpinionOfThisPerson" type="xsd:string"/>
    <xsd:element name="SomeSpecialInfo">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="MyOpinionOfThisPerson"/>
                <xsd:element ref="WhatIThinkTheyAreWorth"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="WhatIThinkTheyAreWorth" type="xsd:string"/>
</xsd:schema>
```

Example instance:

```
<MyInternalDoc
xmlns:hr="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="BrilliantHRSoftware.com"
xsi:schemaLocation="BrilliantHRSoftware.com
HRXMLExtension2.xsd">
    <hr:JobPositionSeeker>
        <hr:JobPositionSeekerId></hr:JobPositionSeekerId>
        <hr:PersonalData>
            <hr:PersonName></hr:PersonName>
            <hr:VoiceNumber>
                <hr:TelNumber></hr:TelNumber>
            </hr:VoiceNumber>
        </hr:PersonalData>
    </hr:JobPositionSeeker>
    <SomeSpecialInfo>
        <MyOpinionOfThisPerson>Completely unqualified!</MyOpinionOfThisPerson>
        <WhatIThinkTheyAreWorth>He should pay us to work here!</WhatIThinkTheyAreWorth>
    </SomeSpecialInfo>
</MyInternalDoc>
```

3.2 ANY element extension method

3.2.1 Schema/DTD/Instance Elements Explained

This method consists of an HR-XML schema that contains a single occurrence of a single element as a last child of the root element that is a data type of ANY. This element is used to house all extensions to the schema. Definitions of those extensions occur in a separate schema. (A similar way to do this with DTD is possible.) This method overcomes a drawback of the wrapper technique in that it maintains a consistent root element, which is a standard HR-XML defined element. It does, however, require work groups to accommodate by including an extension element in their schema design.

Example HR-XML schema:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
xmlns="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1"
targetNamespace="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1">
    <xsd:element name="JobPositionSeeker">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="JobPositionSeekerId" type="xsd:string"/>
                <xsd:element name="PersonalData">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="PersonName" type="xsd:string"/>
                            <xsd:element name="VoiceNumber">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="TelNumber" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
```

```
            <xsd:element ref="HRXMLExtension" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="HRXMLExtension">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any processContents="skip"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

[Note to TSC: the "processContents" attribute raises an issue – here are values for that enumeration from the schema spec:

strict

There must be a top-level declaration for the item available, or the item must have an xsi:type, and the item must be ·valid· as appropriate.

skip

No constraints at all: the item must simply be well-formed XML.

lax

If the item, or any items among its [children] if it's an element information item, has a uniquely determined declaration available, it must be ·valid· with respect to that definition, that is, ·validate· where you can, don't worry when you can't.

We should clarify which of these should be used.]

Example extension definition:

```
<xsd:schema
targetNamespace="BrilliantHRSoftware.com"
xmlns="BrilliantHRSoftware.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xsd:element name="JunkElement">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:restriction base="xsd:string">
                    <xsd:attribute name="contextXPath" type="xsd:string"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Example instance:

```
<JobPositionSeeker
xmlns="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1
JobPositionSeeker-1_1.xsd
```

```
BrilliantHRSoftware.com
HRXMLExtension1.xsd">
    <JobPositionSeekerId/>
    <PersonalData>
        <PersonName/>
        <VoiceNumber>
            <TelNumber>567-5678</TelNumber>
        </VoiceNumber>
    </PersonalData>
    <HRXMLExtension>
        <JunkElement contextXPath="//VoiceNumber/TelNumber" xmlns="BrilliantHRSoftware.com">this is an emergency
telephone number only</JunkElement>
    </HRXMLExtension>
</JobPositionSeeker>
```

4 Implementation Considerations

4.1 Wrapper

4.1.1 Constraints

1) If a pointer to a contextual node is used in an extension element, it SHOULD have the form of an XPath attribute or element with the name “contextXPath.” (Alternately, a DOM referencing node MAY be used with the name “contextDOMPath.”)

```
<xsd:attribute name="contextXPath" type="xsd:string"/>
<xsd:attribute name="contextDOMPath" type="xsd:string"/>
```

2) Extension elements MUST be in a namespace outside the Consortium’s realm, meaning it MUST NOT contain “hr-xml.org.” (It is not recommended to use namespaces with DTDs.)

3) Extension nodes defined separately (whether as an XSD include or as a DTD external entity), MAY conform to a file naming convention of:

   HRXMLExtension1.xsd – first extension file name (i.e. with trading partner A)

   HRXMLExtension2.xsd – second extension file name (i.e. trading partner B)

   And so on….

4) Extensions MUST NOT be used in any certification process.

4.2 ANY

4.2.1 XML 1.0 Validity Constraint on ANY

A validity constraint in the XML 1.0 spec states that while an element may have ANY as a content model, the elements within it must be defined. In short, it isn't simply an escape hatch for anything. A specific "anything" must be defined somewhere, such as with external entities or includes.

4.2.2 Constraints

1) The XSD declaration for this element MUST conform to accepted schema design guidelines and scoped globally as such:

```
<xsd:element name="HRXMLExtension">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any processContents="skip"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

2) It must be named "HRXMLExtension".

3) It MUST be the last child of the root element.

4) It MUST have a minimum occurrence of zero and a maximum occurrence of one.

```
<xsd:element ref="HRXMLExtension" minOccurs="0" maxOccurs="1"/>
```

5) Its declaration MUST have a "processContents" value of <any> indicated.

   The valid values are (from the Schema spec);

   strict

   There must be a top-level declaration for the item available, or the item must have an xsi:type, and the item must be valid as appropriate.

   skip

   No constraints at all: the item must simply be well-formed XML.

   lax

   If the item, or any items among its [children] if it's an element information item, has a uniquely determined declaration available, it must be valid with respect to that definition, that is, validate where you can, don't worry when you can't.

6) If a pointer to a contextual node is used in an extension element, it SHOULD have the form of an XPath attribute or element with the name “contextXPath.” (Alternately, a DOM referencing node MAY be used with the name “contextDOMPath.”)

```
<xsd:attribute name="contextXPath" type="xsd:string"/>
<xsd:attribute name="contextDOMPath" type="xsd:string"/>
```

7) Extension elements MUST be in a namespace outside the Consortium’s realm, meaning it MUST NOT contain “hr-xml.org.” (It is not recommended to use namespaces with DTDs.)

8) Extension nodes defined separately (whether as an XSD include or as a DTD external entity), MAY conform to a file naming convention of:

   HRXMLExtension1.xsd – first extension file name (i.e. with trading partner A)

   HRXMLExtension2.xsd – second extension file name (i.e. trading partner B)

   And so on….

9) Extensions MUST NOT be used in any certification process.

5 Issues List

Issue	Resolution	Rationale
Do we want to make a convention regarding full path or relative paths to HRXMLExtension1.xsd? Validity of extension – do we require well-formed or valid extensions (processContents attribute of ANY)?		

6 Appendix A - Document Version History

Version	Date	Description
1_0	2001-08-08	First draft
1_0	2001-09-11	Added contextDOMPath attribute notes.

7 Appendix B – Related Documents

Reference	Link
XML Schemas: Best Practices	http://www.xfront.com/BestPracticesHomepage.html
XML Namespaces	http://www.w3.org/TR/REC-xml-names/

Schema Design Guidelines	http://schemas.hr-xml.org/xc/canon/TSC/SchemaDesignGuidelines-1_0-20010712.pdf

8 Appendix C – Namespace extension method

This method was examined by the Technical Steering Committee and was not endorsed. The central reason was due to the fact that "in context" extensions prevent a document from being properly valid. In essence, an XML parser cannot validate the data because extensions violate the content models of elements as defined in the schema. Since redefining elements was not an alternative, this method cannot be endorsed for use.

8.1 Schema/DTD/Instance Elements Explained

This method uses **XML Namespaces** to differentiate elements of the HR-XML schema from extension elements. The elements are inserted in context within the HR-XML schema. (A similar way to do this with DTD is possible using **XML Namespaces**.)

Sample invalid instance:

```
<JobPositionSeeker
xmlns="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ns.hr-xml.org/RandS/JobPositionSeeker-1_1
JobPositionSeeker-1_1.xsd
BrilliantHRSoftware.com
BrilliantHRSoftware.xsd">
    <JobPositionSeekerId/>
    <PersonalData>
        <PersonName/>
        <VoiceNumber>
            <TelNumber>567-5678</TelNumber>
            <JunkElement xmlns ="BrilliantHRSoftware.com">this is an emergency number only</JunkElement>
        </VoiceNumber>
    </PersonalData>
</JobPositionSeeker>
```

<JunkElement> is not in the content model of <JobPositionSeeker> even though it is part of a different namespace.