# HR XML Schema Extension 1.0

# Recommendation 2002-April-30

**This version:**
HRXMLExtension-1_0

**Previous version:**
ExtendingSchemas-1_0 (draft)

**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

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 Extending Schemas

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.

## *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.

**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 it is used in the **XML Schemas: Best Practices** discussion, where “extending” means adding functionality to schema that does not currently exist. The latter is for such functionality 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.

### *1.3 Extension 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 not endorsed.

# 2 Wrapper Extension Method

This method consists of a schema that uses an HR-XML schema as an <import>. 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
UserArea2.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 ANY Element Extension Method

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. 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" 98
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="UserArea" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="UserArea" type="UserAreaType"/>
    <xsd:complexType name="UserAreaType">
        <xsd:sequence minOccurs="0">
            <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

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
UserArea1.xsd">
    <JobPositionSeekerId/>
    <PersonalData>
        <PersonName/>
        <VoiceNumber>
            <TelNumber>567-5678</TelNumber>
        </VoiceNumber>
    </PersonalData>
    <UserArea>
        <JunkElement contextXPath="//VoiceNumber/TelNumber" xmlns="BrilliantHRSoftware.com">this is an emergency
telephone number only</JunkElement>
    </UserArea>
</JobPositionSeeker>
```

## *3.1 ANY element extension schema defined and declared*

The HR-XML “ANY” element and complexType MUST be scoped globally and MUST be explicitly defined as:

```
    <xsd:element name="UserArea" type="UserAreaType"/>
    <xsd:complexType name="UserAreaType">
        <xsd:sequence minOccurs="0">
            <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
```

# 4 Extension Implementation Considerations

## 4.1 Wrapper Constraints

1) If a pointer to a contextual node is used in an extension element, it MAY 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 default.

3) Extension nodes defined separately MAY conform to a file naming convention of:

   UserArea1.xsd – first extension file name (i.e. with trading partner A)

   UserArea2.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 Constraints

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

```
<xsd:element name="UserArea" type="UserAreaType"/>
<xsd:complexType name="UserAreaType">
    <xsd:sequence minOccurs="0">
        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

2) It must be named “UserArea”.

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="UserArea" minOccurs="0" maxOccurs="1"/>
```

5) Its declaration MAY have a “processContents” value of “strict” indicated.

   From the Schema spec:

   **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."

This functionality varies in implementation and is sometimes deceptively not implemented at all. It is recommended NOT to use "lax" as a processContents value.

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"/>
```

5) Extension elements MUST be in a namespace outside the default.

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

   UserArea1.xsd – first extension file name (i.e. with trading partner A)

   UserArea2.xsd – second extension file name (i.e. trading partner B)

   And so on….

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

# 5 Appendix A - Namespace Extension Method

## *5.1 Position Statement*

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.

## *5.2 Namespace Extension Method 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. 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.

# 6 Appendix B – Document Version History

| Version | Date | Description |
|---|---|---|
| 1.0 | 2002-April-30 | Approved Recommendation |