Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

ISO/IEC 19757-9 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 34, Document Description and Processing Languages.

— *Part 1: Overview*

— *Part 2: Regular grammar-based validation — RELAX NG*

— *Part 3: Rule-based validation — Schematron*

— *Part 4: Namespace-based validation dispatching language — NVDL*

— *Part 5: Datatype Library Language — DTLL*

— *Part 6: Path-based integrity constraints*

— *Part 7: Character Reportoire Description Language — CRDL*

— *Part 8: Document Schema Renaming Language — DSRL*

— *Part 9: Datatype- and namespace-aware DTDs*

— *Part 10: Validation Management*

Introduction

The language of Document Type Definitions (DTDs) was the original schema language defined by W3C XML and was closely based upon the DTD language defined by SGML. For a variety of reasons, both technical and economic, many users of XML for document-centric applications, especially among those who were previously (and in some cases continue to be) users of SGML, still favour the use of DTD language for grammar-based schema definition in such applications.

It is important to provide users that have made a significant investment in DTDs with a migration path that will enable them to adopt DSDL without having to translate all their existing DTDs to a different schema language, especially as this would oblige them to replace all systems that only work with DTDs, with all the expense and organizational upheavals thereby entailed. A sensible migration path should enable such users to continue to use DTDs for as much of the document validation process as can reasonably be managed, but also enable them to reap the benefits of using those parts of DSDL that most obviously complement and extend the use of DTDs for validation purposes.

It is equally important that the migration path should enable users to continue to use legacy systems that are incapable of using any kind of extension to the DTD language, while at the same time introducing new systems that are equipped to use such extensions. The method of extension must therefore be such that DTDs with extended functionality are valid XML DTDs in accordance with W3C XML. It should remain possible to validate an instance *that does not make any use of the extended functionality* against a DTD that contains the extended functionality, using legacy system tools, and achieve the same result as would be achieved if the DTD did not contain the extended functionality.

The most significant validation tasks that cannot be performed using a DTD alone are:

— validation of names with respect to namespaces

— validation of data content and attribute values with respect to datatypes

— rules-based validation

Of these three, the last is made possible by implementation of Part 3 of DSDL. The first two tasks are currently only possible if Part 2 of DSDL is implemented, but existing DTD users are unlikely to wish to maintain parallel RELAX NG and DTD schemas for each application simply as a means of supporting the use of namespaces and datatypes.

For these reasons this International Standard addresses the extension of DTDs to support the use of namespaces and datatypes.

Document Schema Definition Languages (DSDL) — Part 9: Namespace- and datatype-aware DTDs

1 Scope

This International Standard defines a language that is designed to extend the functionality of an XML DTD to include:

— specifying one or more namespaces to which some or all of the element and attribute names in a DTD belong

— constraining elements with content model 'ANY' to contain elements whose names belong to one or more specified namespaces

— specifying datatypes for elements that contain data content and for attribute values.

Two alternative syntax bindings for this language are defined. The first syntax binding uses XML processing instructions and is designed to enable declarations in this language to be embedded within an XML DTD without invalidating the DTD or altering its interpretation so far as legacy DTD parsers are concerned. This first syntax also provides a means of associating a DTD with an external subset containing declarations in either syntax. This syntax is defined in Clause 4 using the modified BNF syntax notation used in W3C XML.

The second syntax binding uses an XML document syntax and is defined in Clause 6. The syntax rules are defined by a schema that conforms to the RELAX NG Compact Syntax defined in Part 2 of this standard. This syntax is designed to enable declarations in this language to be expressed almost entirely in XML (in this case one XML processing instruction needs to be inserted in the DTD), to facilitate implementation using existing XML tools, either as a namespace-qualified fragment embedded within an XML instance or as a separate XML document.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

SGML, *ISO 8879:1986 Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML)*, http://www.iso.org/

W3C XML, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, 30 October 2003, http://www.w3.org/TR/2003/PER-xml-20031030/

W3C XML-Names, *Namespaces in XML*, W3C Recommendation, 14 January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

RELAX NG, *Grammar-based schema language (RELAX NG)*, ISO/IEC 19757 Part 2, http://www.dsdl.org/

DTLL, *Datatype Library Language*, ISO/IEC 19757 Part 5, http://www.dsdl.org/

IRI, *Internationalized Resource Identifiers (IRI)*, IETF RFC 3987, http://www.ietf.org/rfc/rfc3987

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply:

3.1 qualified datatype name

A datatype name containing a qualifying prefix and a local part separated by a colon ':', the prefix of which is bound to the IRI of a datatype library by a datatype library prefix binding declaration as described in this standard.

3.2 unqualified datatype name

A datatype name does not contain a colon ':' and therefore contains only a local part.

4 DTD extension declarations using XML processing instruction syntax

The following set of syntax productions define how namespace and datatype declarations expressed in accordance with this standard may be represented as XML processing instructions. Syntax productions for *name-token-group*, *NCName*, *URI*, *lit*, *lita* and *s* are given in W3C XML.

DTD-extension-processing-instruction ::= *pio* "DSDL-9" *s* (*namespace-prefix-binding-declaration* | *namespace-name-binding-declaration* | *wildcard-namespace-qualifier-declaration* | *default-datatype-library-declaration* | *datatype-library-prefix-binding-declaration* | *datatype-declaration* | *external-declarations-subset-locator*) *pic*

namespace-prefix-binding-declaration ::= "namespace-prefix-binding" *s* "namespace-IRI=" *IRI-literal s* "prefix=" *prefix-literal*

namespace-name-binding-declaration ::= "namespace-name-binding" *s* "namespace-IRI=" *IRI-literal s* "applies-to-element=" *name-locator-literal*

wildcard-namespace-qualifier-declaration ::= "wildcard-namespace-qualifier" *s* "wildcard-namespace-list=" *wildcard-namespace-literal s* "applies-to-element=" *name-locator-literal*

wildcard-namespace-literal ::= ((*lit wildcard-namespace-list lit*) | (*lita wildcard-namespace-list lita*))

wildcard-namespace-list ::= *IRI* (*s IRI*)*

name-locator-literal ::= ((*lit name-locator-sequence lit*) | (*lita name-locator-sequence lita*))

name-locator-sequence ::= (*s*)? (*name-token-group* | *any-name*) (*s*)?

any-name ::= "#any"

default-datatype-library-declaration ::= "default-datatype-library=" *IRI-literal*

datatype-library-prefix-binding-declaration ::= "datatype-library-prefix-binding" *s* "datatype-library-IRI=" *IRI-literal s* "prefix=" *prefix-literal*

datatype-declaration ::= "datatype=" (*datatype-name* | *datatype-qname*) *s* (("applies-to-element=" *name-locator-literal*) | ("applies-to-attribute=" *name-locator-literal* "of-element" *name-locator-literal*))

datatype-name ::= *NCName*

datatype-qname ::= ((*lit prefix* ":" *datatype-name lit*) | (*lita prefix* ":" *datatype-name lita*))

prefix-literal ::= ((*lit prefix lit*) | (*lita prefix lita*))

prefix ::= *NCName*

external-declarations-subset-locator ::= "external-declarations-subset" *s* "location=" *IRI-literal s* "syntax=" ((*lit* ("PI" | "XML") *lit*) | (*lita* ("PI" | "XML") *lita*))

IRI-literal ::= ((*lit IRI lit*) | (*lita IRI lita*))

IRI ::= *URI*

NOTE - An *IRI* must be a *URI* that conforms to IRI.

5 Location of DTD extension declarations in a DTD

DTD extension declarations in XML processing instruction syntax may be located within a DTD's external subset or within the local subset of a DTD declaration.

6 DTD extension declarations using XML document syntax

The following schema in RELAX NG Compact Syntax specifies how a set of namespace and datatype declarations expressed in accordance with this standard may be represented as an XML document.

```
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"
            default namespace = "http://dsdl.org/dsdl-9"

            start = document

            document = element dtd-extension { head, body }

            head = (element applies-to-dtd {public | system} )*

            public = element public {text}, system?

            system = element system {xsd:anyURI}

            body = ( namespace-prefix-binding |
                     namespace-name-binding |
                     wildcard-namespace-qualifier |
                     default-datatype-library |
                     datatype-library-prefix-binding |
                     datatype )+

            namespace-prefix-binding = element namespace-prefix-binding {
                                    namespace-iri, prefix
                                    }

            namespace-name-binding = element namespace-name-binding {
                                    namespace-iri, applies-to-element
                                    }

            applies-to-element     = element applies-to-element {
                                    (name+ | any-name)
                                    }

            applies-to-attribute   = element applies-to-attribute {
                                    (name+ | any-name), of-element-names
                                    }

            of-element-names       = element of-element-names {
                                    name+ | any-name
                                    }

            name                   = element name {xsd:NMTOKEN}

            any-name               = element any {empty}

            wildcard-namespace-qualifier = element wildcard-namespace-qualifier {
                                    (namespace-iri, applies-to-element)
                                    }

            default-datatype-library = element default-datatype-library {
```

```
                        library-iri
                        }

datatype-library-prefix-binding = element datatype-library-prefix-binding {
                        (library-iri, prefix)
                        }

library-iri             = element library-iri {xsd:anyURI}

datatype                = element datatype {
                        name, (applies-to-element | applies-to-attribute)
                        }

namespace-iri           = element namespace-iri {xsd:anyURI}

prefix                  = element prefix {xsd:NMTOKEN}
```

7 Inclusion of namespace information in DTDs

A namespace IRI may be associated with the names of any elements that are declared in a DTD, including both qualified and unqualified names. A namespace IRI may also be declared as a constraint on the content of an element whose declared content model is "ANY".

7.1 Associating namespaces with qualified names

A namespace IRI may be associated with specified element names or attribute names that include qualifying prefixes, by declaring a corresponding namespace prefix binding.

Each qualifying prefix may only be bound once to a namespace IRI. If two or more namespace prefix binding declarations specify the same qualifying prefix, the second and subsequent declarations are ignored.

7.2 Associating namespaces with unqualified names

A namespace IRI may be associated with specified element names that do not include qualifying prefixes, by declaring a corresponding namespace name binding.

An unqualified name has no namespace unless a namespace name binding is declared that applies to this name.

A namespace IRI may be associated with any element name for which a namespace name binding has not previously been declared, by declaring a namespace name binding in which the "any name" indicator is included instead of a list of names. Any subsequent namespace name binding declarations are ignored.

Each unqualified name may only be bound once to a namespace IRI. If two or more namespace name binding declarations explicitly specify the same name, the second and subsequent declarations are ignored.

7.3 Associating namespace constraints with elements with content model "ANY"

A namespace IRI may be associated with the content of an element whose content model is declared to be "ANY", by defining a wildcard namespace qualifier declaration for that element. A namespace IRI may only be specified in a wildcard namespace qualifier declaration if it occurs either in a namespace prefix binding declaration or in a namespace name binding declaration for the same DTD.

A wildcard namespace qualifier declaration may only associate a namespace IRI with the content of elements whose content model is "ANY". The name of any element whose content model is not "ANY" will be ignored.

8 Inclusion of datatype information in DTDs

A datatype may be associated with a specified element or attribute, by including a corresponding datatype declaration in the DTD.

Every datatype name must be associated with a declared datatype library. Unqualified datatype names are associated with a default datatype library, while qualified datatype names are associated with declared datatype libraries by declaring a binding between the qualifying prefix and the associated datatype library.

If two or more datatype declarations apply to the same element content or attribute value, the second and subsequent declarations are ignored.

8.1 Associating datatype libraries with qualified datatype names

If a specified datatype name includes a qualifying prefix, a corresponding datatype library prefix binding declaration must be included in the DTD.

Each qualifying prefix may only be bound once to a datatype library. If two or more datatype library prefix binding declarations specify the same prefix, the second and subsequent declarations are ignored.

8.2 Associating datatype libraries with unqualified datatype names

If a specified datatype name has no qualifying prefix, a default datatype library declaration must be included in the DTD.

8.3 Datatype libraries

The IRI specified in either a datatype library prefix binding declaration or a default datatype library declaration must identify a datatype library, for example a library that conforms to Part 5 of this International Standard (DTLL).

If a datatype name is specified without a qualifying prefix, the DTD must include a default datatype library declaration. A DTD may only have one default datatype library. The second and subsequent default datatype library declarations are ignored.

9 Validity of DTDs and instances that include namespace or datatype information in accordance with this Part of DSDL

9.1 Validity of DTDs

If a DTD is valid according to a legacy parser, it must be valid according to a DSDL-9-aware parser, regardless of whether or not it contains namespace or datatype information in accordance with this Part of DSDL.

Similarly, if a DTD is invalid according to a legacy parser, it must be invalid according to a DSDL-9-aware parser, which must report the same errors as a legacy parser.

If a DTD contains namespace or datatype information that is intended to be expressed in accordance with this Part of DSDL, but contains errors of any kind, a DSDL-9-aware parser may report these errors in warning messages but shall not report the DTD as being invalid.

9.2 Validity of instances

If an instance is valid according to a legacy parser and its DTD does not contain namespace or datatype information in accordance with this Part of DSDL, it must be valid according to a DSDL-9-aware parser.

Similarly, if an instance is invalid according to a legacy parser and its DTD does not contain namespace or datatype information in accordance with this Part of DSDL, it must be invalid according to a DSDL-9-aware parser, which must report the same errors as a legacy parser.

Annex A
(informative)

Example declarations

A.1 Examples in XML processing instruction syntax

-- EXAMPLES TO BE ADDED HERE --

A.2 Examples in XML document syntax

-- EXAMPLES TO BE ADDED HERE --