

Contents		Page
Foreword		iv
Introduction		v
1	Scope	1
2	Normative references	1
3	Terms and definitions	2
4	The role of declarative document architectures	2
5	Defining a document fragment template	2
6	Reusing document architectures	3
6.1	Reassigning element and attribute names	3
6.2	Remapping entity references	4
6.3	Renaming Parameter Entity Targets	4
7	Removing elements and attributes from specific locations within a document model	5
Annex A (normative) Validation of declarative document architectures		6
A.1	RELAX NG XML Schema for Validating Declarative Document Architectures	6
A.2	RELAX NG Compact Schema for Validating Declarative Document Architectures	6
A.3	Schematron Rules for Validating Declarative Document Architectures	6
Annex B (informative) Using XSLT to generate XForms		7
Bibliography		8

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 19757-8 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 34, Document Description and Processing Languages.

ISO/IEC 19757 consists of the following parts, under the general title *Document Schema Definition Languages (DSDL)*:

- *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: Datatypes*
- *Part 6: Path-based integrity constraints*
- *Part 7: Character repertoire validation*
- *Part 8: Declarative document architectures*
- *Part 9: Datatype- and namespace-aware DTDs*
- *Part 10: Validation management*

Introduction

This International Standard defines a set of Document Schema Definition Languages (DSDL) that can be used to specify one or more validation processes performed against Extensible Stylesheet Language (XML) or Standard Generalized Markup Language (SGML) documents. (XML is an application profile SGML ISO 8879:1986.)

A document model is an expression of the constraints to be placed on the structure and content of documents to be validated with the model. A number of technologies have been developed through various formal and informal consortia since the development of Document Type Definitions (DTDs) as part of ISO 8879, notably by the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS). A number of validation technologies are standardized in DSDL to complement those already available as standards or from industry.

To validate that a structured document conforms to specified constraints in structure and content relieves the potentially many applications acting on the document from having to duplicate the task of confirming that such requirements have been met. Historically, such tasks and expressions have been developed and utilized in isolation, without consideration for how the features and functionality available in other technologies might enhance validation objectives.

The main objective of this International Standard is to bring together different validation-related tasks and expressions to form a single extensible framework that allows technologies to work in series or in parallel to produce a single or a set of validation results. The extensibility of DSDL accommodates validation technologies not yet designed or specified.

In the past, different design and use criteria have led users to choose different validation technologies for different portions of their information. Bringing together information within a single XML document sometimes prevents existing document models from being used to validate sections of data. By providing an integrated suite of constraint description languages that can be applied to different subsets of a single XML document, this International Standard allows different validation technologies to be integrated under a well-defined validation policy.

This multi-part International Standard integrates constraint description technologies into a suite that:

- provides user control of names, order and repeatability of information objects (elements)
- allows users to identify restrictions on the co-concurrence of elements and element contents
- allows specific subsets of structured documents to be validated
- allows restrictions to be placed on the contents of specific elements, including restrictions based on the content of other elements in the same document
- allows the character set that can be used within specific elements to be managed, based on the application of the ISO/IEC 10646 Universal Multiple-Octet Coded Character Set (UCS)
- allows default values to be assigned to element contents and attribute values, and provides facilities for the incorporation of predefined fragments of structured data to be incorporated within documents
- allows SGML to be used to declare document structure constraints that extend DTDs to include functions such as namespace-controlled validation and datatypes.

Document Schema Definition Languages (DSDL) — Part 8: Declarative document architectures

1 Scope

Declarative document architectures provide templates that can be used to define the structure and/or content of predefined parts of document streams.

Note 1: Templates created using declarative document architectures are similar in purpose to abstract classes.

Declarative document architectures also allow default values to be assigned to specific parts of a data stream. This includes mechanisms for defining standard sequences of data that can be incorporated into document instances by reference to an identifying name, the provision of default content for elements and attributes for which no value is provided, and the matching of local element and attribute names to those used in a specific schema.

This Part provides a syntax for:

- using XPath to address elements and attributes to be modified
- assigning a default value to the contents of a specific type of element or attribute
- defining named fragments of predefined data elements that can be included within a document instance
- renaming elements, attributes, entities and processing instructions in specific locations within the document model, including the assignment of element or attribute names to different namespaces
- the definition of replacement contents for specific elements or attributes
- removing elements or attributes from specific locations within the document model.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 19757. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 19757 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

Each of the following documents has a unique identifier that is used to cite the document in the text. The unique identifier consists of the part of the reference up to the first comma.

URI, IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, Internet Standards Track Specification, August 1998, <http://www.ietf.org/rfc/rfc2396.txt>

XML, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 6 October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>

XML-Infoset, *XML Information Set*, W3C Recommendation, 24 October 2001, <http://www.w3.org/TR/2001/REC-xml-infoset-20011024/>

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

XPath, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>

XML Schema, *XML Schema*, W3C Recommendation, 24 October 2001, <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>

XSLT, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, <http://www.w3.org/TR/1999/REC-xslt-19991116>

XInclude, *XML Inclusions (1.0)*, W3C Recommendation, ???, <http://www.w3.org/TR/200?/REC-???/>

XForms, *XML Forms (1.0)*, W3C Recommendation, ?? October 2003, <http://www.w3.org/TR/2003/REC-????/>

3 Terms and definitions

document architecture

set of rules that are used to map a document instance to a document model defined by one or more schemas

4 The role of declarative document architectures

Declarative document architectures provide templates that can be used to define the structure and/or content of specific fragments of a document instance within a schema or DTDs. If they are complete, such fragments can be incorporated into document instances using XInclude. If they are only partially complete, and require further input from users to form a section that can be validly included in the document, they can be presented to users as an XML form (e.g. XForms compliant data requests) to ensure the capture of missing data.

In addition to forming an XML-based mechanism that mimics the functionality of SGML and XML entities, document architectures can form the basis of abstract design patterns. An element in a document instance can, either by being assigned a fixed attribute value in a schema or by inclusion of an attribute in the instance, be mapped to a named element in a specified schema, so that it can be validated using the declarations in that schema. If subelements have been renamed within the identified element these can be mapped to new names either by use of attributes associated with individual elements or by the use of a "name map" associated with any of their parent elements.

Declarative document architectures can also be used to remap entity references to alternative names, and to reassign entity maps as parts of different processes. For example, at certain stages in processing it may be important to retain entity references without expansion. In such instances the entity replacement mechanism can be replaced by one that automatically maps each entity reference to an entity reference, without breaking the rules about recursive entity references.

Note 2: This functionality mimics that of SDATA entity definitions in SGML.

Elements and attributes that conform to this Part shall have a namespace whose associated URI is:

`url://dsdl.org/Part8`

In this Part the prefix `dsdl8` is used to identify points at which this URI defines the namespace.

5 Defining a document fragment template

A document fragment template defines a well-formed set of related elements, with or without predefined contents for their elements and attribute values. Document fragment templates can only be defined in as direct children of schemas.¹ The `dsdl8:predefined-fragment` element may be used to identify document fragment templates within a schema. This element can be used in two forms:

- As an element whose contents contain the required fragment

¹ Document fragment templates cannot be defined as part of an element or attribute definition or within any rule or pattern defined by other parts of the International Standard.

- As an empty element whose `source` attribute contains a URI that identifies where a copy of the template can be obtained from.

The namespace of the root element of a document fragment must be specifically declared using an attribute defined according to the specification for XML-Names. The optional `dsd18:schema-source` attribute can be used to enter a URI that where a copy of a schema that can be used to validate the fragment can be obtained.

Note 3: Strictly speaking the namespace definition should be sufficient to identify the schema required. The `dsd18:schema-source` attribute can be used to provide a locally significant mapping of the namespace name to a specific copy of the validation schema.

Elements that require further input before the fragment can be embedded in a document instance must be flagged with a `dsd18:request-content` attribute whose value is `true`. Where attribute values need to be specified before the fragment is complete the `dsd18:request-attributes` attribute may be added. The value of this attribute is a set of nametokens identifying the qualified name of all attributes whose value needs to be captured.

Note 4: A document fragment template must be a well-formed XML document that can be included in an XML document once missing content and attribute values have been defined.

Note 5: This standard does not specify how attribute value and element content should be supplied to complete the elements and attributes identified as requiring completion before inclusion, only that they do require completion. Missing values can be supplied by parameters passed to XSLT transformations, by use of XForms that request the relevant information, or any other mechanism deemed suitable by the application for capturing the required information.

Validators must report an error if one or more elements or attributes are still to be provided with values when the document fragment template is included into a document instance.

6 Reusing document architectures

In many cases the names assigned to elements in existing schemas and attributes are specified in a language that is not understood by user communities. The facilities in this clause allow locally-significant names to be mapped to those used to name elements, attributes, entities and processing instruction types declared in a referenced schema.

6.1 Reassigning element and attribute names

To map an element to a differently named element in a schema definition users of this Part of the standard can:

- create a mapping schema that declares the mappings required, or
- assign a `dsd18:map-names` attribute to an element in a document instance.

A `dsd18:name-map` element is used to identify reusable mappings between names used in schemas and those used in document instances. The element can be included as a foreign element in the definition of the appropriate element or attribute in the schema. The contents of the element consists of a list of nametokens that identify alternative names that can be assigned to the element or attribute. Names may be qualified providing the relevant namespaces have been declared within the schema.

More than one `dsd18:name-map` can be associated with a particular element or attribute definition. Nametokens from multiple declarations are concatenated. Maps stored in an external resource may be included using the schema's normal inclusion mechanism.

Mappings that are specific to a given instance can be specified using a `dsd18:map-names`. The contents of the attribute consists of matched pairs of nametokens, where the first name is the name of the element or attribute in the document instance and the second is the equivalent name in the schema.

Note 6: It is assumed that such attributes will be defined as fixed attributes in the local document declaration so that they only need to be defined once in the document, not on each instance of the element.

Where a `map-names` attribute is assigned to an element which also has a `name-map` element within its model definition local definitions extend those currently defined.

Note 7: Multiple names can be assigned to any element. Declarations that duplicate an existing map entry can be ignored.

Should we allow names to be paths so that we can define different rules for specific attributes associated with specific elements? (Part 1 says that XPath should be used here, and there is not reason why a well-controlled subset should not be used, but who defines this set remains an open question.) In theory adopting this option would allow users to change one occurrence of an attribute/element to be mapped to one point in the schema, defined using the `position()` function, and another to another, but this may be too powerful to be implemented efficiently. Would the gain be worth the pain?

Name maps are inherited by descendants. Where all attributes of a specific name-map refer to a single name in the relevant schema this means that definitions placed on the parent element will suffice for the children. But where mapping of an attribute name depends on its parent then the mapping must be associated with the definition of the child element to affected. Should this definition be specific to that element, but other inherited mappings need to be assigned to descendants of the element a `dSDL8:inherit-map-of-parent` must be assigned a value of `true`.

Note 8: The default value of the `dSDL8:inherit-map-of-parent` attribute is `false`.

Ideally you would add an extra field to the XML Infoset to record the name used in the instance and then use the existing fields to record the name and namespace of the declaration the element has been validated against. (You could change the definition of `name()` so that it returned the actual name and reserve `namespace()` and `local-name()` for returning the names of the element in the schema.) But if we are not allowed to extend the Infoset spec, will `name()` and `local-name()` refer to the name as entered in the document instance, or that in the schema the document is validated against? (In other words, is reconstructing the document instance more important than passing on information about the valid information along the pipeline, or is there some way of doing both without extending the Infoset?)

6.2 Remapping entity references

Often the names assigned to entity references are difficult for users to understand, especially when they are specified using a language which is not the native language of a particular user community. The facilities in this clause allow locally-significant names to be mapped to those used to define entities in a referenced entity set.

To map an entity reference to a differently named entity in a entity definition users of this Part of the standard can:

- create a mapping schema that declares the mappings required
- assign a `dSDL8:map-entities` attribute to an element in a document instance.

A `dSDL8:entity-name-map` element is used to identify reusable mappings between names used in entity definitions and those used in entity references. Such mappings may be defined using a foreign element in a schema that can be defined at any level in the model at which entities may be defined. The contents of the element consists of matched pairs of entity names where the first name is that assigned to the entity in its definition and the second is an alternative name for the entity. The same defined entity name can be used as the first member of more than one entry. More than one `dSDL8:entity-name-map` element can occur in the same schema. Entries in such maps shall be concatenated.

This clause also specifies attributes that can be used to indicate when entity mapping is to be disabled within a document instance, or within a specific element within a document instance. When an element includes a `dSDL8:disable-entity-mapping` attribute whose value is `true` entity reference replacement within the contents of the specified element and any children that do not have a value of `false` for the same attribute are inhibited.

6.3 Renaming Parameter Entity Targets

Where the names of and properties of parameter entities have not been defined in terms understandable to user-communities, users of this Part of the standard can create a mapping schema that declares the alternative names

to be used by adding an empty `dsdl8:map-pi-target` element as a foreign element under the root node of a schema. The element must be assigned values for two compulsory attribute:

- `dsdl8:target-name`, which contains the name to be used as the `PITarget` value for mapped processing instructions
- `dsdl8:alternative-names`, which contains one or more nametokens that can be used as alternative names for the target name.

Where properties of the target namespace are also to be assigned locally significant names a `dsdl8:property-names` attribute can be assigned to the element. The contents of this attribute is a set of matched pairs of nametokens, the first member of the pair being a property name applicable to processing instructions whose target specified for the element, and the second of which is an alternative name for that property.

Note 9: Multiple assignments of alternative names to the same target property shall not be considered to be an error.

7 Removing elements and attributes from specific locations within a document model

In some situations it is important to be able to restrict the occurrence of certain elements or attributes within specific fragments of documents, perhaps because values in contents make it obvious that such elements and attributes are no longer relevant.

Note 10: While Part 3 of this standard allows errors to be reported when such situations occur it does not provide facilities for inhibiting such elements within fragments.

Note 11: This clause provides facilities that mimic the SGML `exclusions` option. (Part 2 contains a mechanism for extending schema definitions within local declarations that is similar in effect to SGML inclusions.)

Should we be duplicating the exclusions function, or should we be content with reporting validation errors if excluded elements are identified without providing any further control of models?

To be completed

Annex A (normative)

Validation of declarative document architectures

The normative schemas defined in this annex provide formal definitions for the elements and attributes used to declare document architectures. The elements defined by these schemas will normally be used as foreign elements within schemas or as foreign attributes within document instances.

A.1 RELAX NG XML Schema for Validating Declarative Document Architectures

To be completed

A.2 RELAX NG Compact Schema for Validating Declarative Document Architectures

To be completed

A.3 Schematron Rules for Validating Declarative Document Architectures

To be completed

Annex B (informative)

Using XSLT to generate XForms

This annex contains an XSL Transformation that will convert any incomplete document fragment into an XForm whose result, after completion of all fields, will be a complete document fragment that can be incorporated into a document instance using XInclude.

To be completed

Bibliography

- [1] *XSL Transformations (XSLT) Version 1.0*, <http://www.w3.org/TR/xslt>

Summary of editorial comments:

[6.1] Reassigning element and attribute names

Should we allow names to be paths so that we can define different rules for specific attributes associated with specific elements? (Part 1 says that XPath should be used here, and there is not reason why a well-controlled subset should not be used, but who defines this set remains an open question.) In theory adopting this option would allow users to change one occurrence of an attribute/element to be mapped to one point in the schema, defined using the position() function, and another to another, but this may be too powerful to be implemented efficiently. Would the gain be worth the pain?

[6.1] Reassigning element and attribute names

Ideally you would add an extra field to the XML Infoset to record the name used in the instance and then use the existing fields to record the name and namespace of the declaration the element has been validated against. (You could change the definition of name() so that it returned the actual name and reserve namespace() and local-name() for returning the names of the element in the schema.) But if we are not allowed to extend the Infoset spec, will name() and local-name() refer to the name as entered in the document instance, or that in the schema the document is validated against? (In other words, is reconstructing the document instance more important than passing on information about the valid information along the pipeline, or is there some way of doing both without extending the Infoset?)

[7] Removing elements and attributes from specific locations within a document model

Should we be duplicating the exclusions function, or should we be content with reporting validation errors if excluded elements are identified without providing any further control of models?