

Annex A. G5 Defect Report

DEFECT REPORT

The submitter of a defect report shall complete the items in Part 2 and shall send the form to the Convener or the Secretariat of the WG with which the relevant editor's group is associated.

The WG Convener or Secretariat shall complete the items in Part 1 and circulate the defect report for review and response by the appropriate defect editing group.

The defect editor shall complete Part 4 and submit the completed report to the Convener or the Secretariat of the WG.

PART 1 - TO BE COMPLETED BY WG SECRETARIAT
DEFECT REPORT NUMBER:
WG SECRETARIAT:
DATE CIRCULATED BY WG SECRETARIAT:
DEADLINE ON RESPONSE FROM EDITOR:

PART 2 - TO BE COMPLETED BY SUBMITTER
SUBMITTER: Rick Jelliffe, Editor IS19757-3
FOR REVIEW BY: SC34 WG1
DEFECT REPORT CONCERNING (number and title of International Standard or DIS final text): ISO/IEC IS 19757 Part 3 Schematron
QUALIFIER (e.g. error, omission, clarification required): Omission (additional features)
REFERENCES IN DOCUMENT (e.g. page, clause, figure, and/or table numbers): Multiple

NATURE OF DEFECT

(complete, concise explanation of the perceived problem):

Several improvements are required to reflect actual adoption and requirements that have emerged since the initial adoption.

- 1) Query language bindings for XSLT2 (normative), XPath2 (informative), EXSLT (informative), STX (informative)
- 2) Property lists for assertions and SVRL: to support type annotations, metadata annotation, tabular results, and richer reports improving integration and usefulness
- 3) Compound document validation: newer formats such as the XML-in-ZIP formats IS26300 ODF and IS29500 OOXML as well as websites and linkbases require easier traversal.
- 4) The current inclusion mechanism is too rudimentary. Often, perhaps typically, users wish to include multiple elements
- 5) Miscellaneous improvements: predicate logic expression may have unnecessary head, SVRL may be missing some attributes available in Schematron and may need to support richer text better

SOLUTION PROPOSED BY THE SUBMITTER

(optional):

- 1) New annexes for the four query language bindings
- 2) Additional clause in body allowing properties with mechanism mirroring existing diagnostics, and corresponding changes for the schemas for Schematron and SVRL. Eg. sch:assert/@properties
- 3) Additional clause in body allowing patterns to be scoped by an XPath to a document or sequence of documents. E.g. sch:pattern/@document Corresponding additions to Schematron and SVRL schemas
- 4) Additional clause in body allowing an import element which brings in patterns, diagnostics, properties, phases and ns declarations. E.g. sch:schema/sch:import Corresponding additions to Schematron schema
- 5) Editor to note and make changes including accumulated non-technical corrigenda

I have distributed drafts of the texts for 1) for at least six months, so that text seems uncontroversial and stable. (Informally, see

http://www.oreillynet.com/blog/2008/02/drafts_for_schematron_support.html)

PART 3 - EDITOR'S RESPONSE

ANY MATERIAL PROPOSED FOR PROCESSING AS A TECHNICAL CORRIGENDUM TO, AN AMENDMENT TO, OR A COMMENTARY ON THE INTERNATIONAL STANDARD OR DIS FINAL TEXT IS ATTACHED TO THIS COMPLETED REPORT:

The editor proposes a new edition. While the proposed new text is backwards compatible, many pages are affected and there are many new pages. The standard is maintained in XML. Furthermore, as this is a standard available for free on the PAS site of ISO, there should be no existing printed stock.

1) QLB Annexes.

Summary: Additional Query Language Binding for XSLT2. Additionally, suggested QLB bindings for XPath2, EXSLT and STX. No change to XSLT1 as the default QLB is involved.

Editor's note: WG1 should decide on each QLB separately. The XSLT2 QLB is implemented, has no reference issues, is popular, and should be normative. The STX QLB has not been implemented at this time, so it may be premature to add this. The EXSLT QLB references a specification that is not standard, so should not be normative. The XPath2 QLB is intended to cope with XQuery and less-than-XSLT systems, with no type awareness, has no reference issues but may be kept informative.

Contents

- [A Query Language Binding for XSLT 2](#)
- [B Query Language Binding for XPath 2](#)
- [C Query Language Binding for EXSLT](#)
- [D Query Language Binding for STX](#)

Annex A

(normative)

Query Language Binding for XSLT 2

The `xslt2` query language binding allows schemas implemented using . All implementations that support the `xslt2` query language binding should also support `xpath2` query language binding.

A Schematron schema with a `queryBinding` attribute with the value `xslt2`, in any mix of upper and lower case letters, shall use the following binding:

- The query language used is the extended version of specified in with backwards compatibility mode as false. Consequently, the data model used is the data model of constructed from an infoset or a PSVI. All namespaces, prefixes, functions and operators defined by shall be available. An implementation may allow user-written functions and extensions, in the appropriate namespace.
- The rule context is interpreted according to the Production Production 1 of . The rule context may be elements, attributes, comments and processing instructions, including sequences of these.
- The assertion test is interpreted according to Production 1 of , using `fn:boolean()` on the result of evaluating the expression and to find the effective boolean value.
- The name query is interpreted according to Production 1 of , using `fn:node-name()` on the result of evaluating the expression which should be an element or attribute node and returning a string value.
- The value-of query is interpreted according to Production Production 1 of , using `fn:string()` on the result of evaluating the expression and returning a string value.
- The let value is interpreted according to Production Production 1 of .
- The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the `$` character. This is a character not found as a delimiter in URLs or XPath. The `$` character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.
- A Schematron let expression is treated as an variable. The `$` delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The `$` character not followed by the name of an in-scope variable shall be treated as a literal character.
- All namespaces and prefixes defined by for modules are reserved with their usage. All functions defined by these modules shall be available in addition to the the functions defined by .

The `key` element may be used, in the namespace, before the pattern elements.

The `function` element may be used, in the namespace, before the pattern elements.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

The `fn:error()` function or other dynamic errors should not be used to provide assertion text or to substitute for assertions and diagnostics.

Annex B

(informative)

Query Language Binding for XPath 2

The `xpath2` query language binding allows schemas implemented using the simplest

A Schematron schema with a `queryBinding` attribute with the value `xpath2`, in any mix of upper and lower case letters, shall use the following binding:

- By default the query language used is that of used with no schema and with backwards compatibility mode as false. Consequently, the data model used is the data model of constructed from an infoset not a post schema validation infoset. All namespaces, prefixes, functions and operators defined by shall be available. No user-written functions or extensions shall be available.
- The rule context is interpreted according to the Production Production 1 of . The rule context may be elements, attributes, comments and processing instructions, including sequences of these.
- The assertion test is interpreted according to Production 1 of , using `fn:boolean()` on the result of evaluating the expression and to find the effective boolean value.
- The name query is interpreted according to Production 1 of , using `fn:node-name()` on the result of evaluating the expression which should be an element or attribute node and returning a string value.
- The value-of query is interpreted according to Production Production 1 of , using `fn:string()` on the result of evaluating the expression and returning a string value.
- The `let` element should not be used.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

The `fn:error()` function or other dynamic errors should not be used to provide assertion text or to substitute for assertions and diagnostics.

Annex C

(informative)

Query Language Binding for EXSLT

The `exslt` query language binding is the default `xslt` binding augmented by the functions.

A Schematron schema with a `queryBinding` attribute with the value `exslt`, in any mix of upper and lower case letters, shall use the following binding:

- The query language used is the extended version of specified in . Consequently, the data model used is the data model of those specifications.
- The rule context is interpreted according to the Production 1 of . The rule context may be elements, attributes, comments and processing instructions.
- The assertion test is interpreted according to Production 14 of , as returning a Boolean value.
- The name query is interpreted according to Production 14 of , as returning a string value. Typically, the `select` attribute contains an expression returning an element node: the name query takes the local or prefixed name of the node, not its value.
- The value-of query is interpreted according to Production 14 of , as returning a string value.
- The let value is interpreted according to Production 14 of , as returning a string value.
- The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the `$` character. This is a character not found as a delimiter in URLs or XPath. The `$` character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.
- A Schematron let expression is treated as an variable. The `$` delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The `$` character not followed by the name of an in-scope variable shall be treated as a literal character.
- All namespaces and prefixes defined by for modules are reserved with their usage. All functions defined by these modules shall be available in addition to the the functions defined by Production 14 of and Production 1 of

NOTE

As does not require all implementations to provide all modules, some schemas using this query language binding will fail if they use functions which the implementation does not provide.

The `key` element may be used, in the namespace, before the pattern elements.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

Annex D

(informative)

Query Language Binding for STX

The `stx` query language binding provides a schema language suited to memory-efficient streaming implementation.

A Schematron schema with a `queryBinding` attribute with the value `stx`, in any mix of upper and lower case letters, shall use the following binding:

- The query language used is the streaming dialect of specified in . Consequently, the data model used is the data model of which is a variant on noting that all elements and attributes are treated as untyped in .
- The rule context is interpreted according to the Production 2 of . The rule context may be elements, attributes, comments and processing instructions.
- The assertion test is interpreted according to Production 1 of , as returning a Boolean value.
- The name query is interpreted according to Production 1 of , as returning a string value. Typically, the `select` attribute contains an expression returning an element node: the name query takes the local or prefixed name of the node, not its value.
- The value-of query is interpreted according to Production 1 of , as returning a string value.
- The let value is interpreted according to Production 1 of , as returning a string value.
- The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the `$` character. This is a character not found as a delimiter in URLs or XPath. The `$` character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.
- A Schematron let expression is treated as an variable. The `$` delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The `$` character not followed by the name of an in-scope variable shall be treated as a literal character.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

The `message` element shall not be used to provide text for assertions and diagnostics.

2) Properties and SVRL.

Summary: Add `assert/@properties`, `report/@properties`, `schema/properties`, `properties/property/property/@scheme`, `property/xslt:copy` to Schematron. This allows the linking to static and dynamic properties: text or foreign elements. Add `failed-assert/property-reference` and `successful-report/property-reference` to SVRL, to transfer this to output. Also enhance SVRL by allowing titles and `p` elements in Schematron rules and SVRL, to increase regularity.

Insert new definition for property to definition section

3.17 property

named data giving additional metadata on an assertion or report

Insert new clauses to 5.5 Ancillary elements and attributes

5.5.n properties element

A section containing individual property elements. An implementation is not required to make use of this element.

NOTE: The properties element was not part of IS19757-3:2006.

5.5.n property element

An element to declare additional arbitrary properties for the subjects of failed assertions and successful reports.

NOTE: The property element is suitable for linking assertions or reports to actions, to additional metadata, to datatyping, and to dynamically extracted text related to the subject.

The optional scheme element should be an IRI or other public identifier which specifies the notation used for the metadata value.

NOTE: Where the property value contains elements in a well-known namespace or where the scheme used is otherwise obvious or unnecessary, the scheme element may be omitted. For example, if the property element contains an IS19757-7 Character Repertoire Description Language schema, no scheme attribute is appropriate.

NOTE: Properties are defined as assertions in order to associate assertion text with the property. A property of an `assert` element typically should be information for validation. A property of a `report` element typically should be information for document augmentation (post schema validation information set.)

An implementation is not required to make use of this element.

NOTE: The property element was not part of IS19757-3:2006.

Insert new non-normative note on role s 5.5.9

NOTE: Typical values for the role attribute on a diagnostic element might be "warning", "caution" and "note".

5.5.3 title element change "schema or pattern" to "schema, pattern or rule"

To Annex C Default Query Language Binding add second last paragraph
(and to the XSLT2 query language binding if accepted)

The XSLT(2) deep copy copy-of element may be used, in the XSLT(2) namespace and without child nodes, inside a property element.

Update Annex E Design Requirements. Add to numbered list

n. Support the declaration of simple static and dynamic properties

Update Annex A RELAX NG Schema for Schematron with the following

```
# Element declarations
schema = element schema {
  attribute id { xsd:ID }?,
  rich,
  attribute schemaVersion { non-empty-string }?,
  attribute defaultPhase { xsd:IDREF }?,
  attribute queryBinding { non-empty-string }?,
  (foreign
  & inclusion*
  & (title?,
  ns*,
  p*,
  let*,
  phase*,
  pattern+,
  p*,
  diagnostics?,
  properties?))
}
```

```
assert = element assert {
  attribute test { exprValue },
  attribute flag { flagValue }?,
  attribute id { xsd:ID }?,
  attribute diagnostics { xsd:IDREFS }?,
  attribute properties { xsd:IDREFS }?,
  rich,
  linkable,
  (foreign & (text | name | value-of | emph | dir | span)*)
}
```

```
report = element report {
  attribute test { exprValue },
  attribute flag { flagValue }?,
  attribute id { xsd:ID }?,
  attribute diagnostics { xsd:IDREFS }?,
  attribute properties { xsd:IDREFS }?,
  rich,
  linkable,
  (foreign & (text | name | value-of | emph | dir | span)*)
}
```

```
rule = element rule {
  attribute flag { flagValue }?,
  rich,
  linkable,
  (foreign & inclusion*
  & ((attribute abstract { "true" }, title?,
  attribute id { xsd:ID }, let*, (assert | report | extends|p)+)
  | (attribute context { pathValue },
  attribute id { xsd:ID }?,
  attribute abstract { "false" }?, title?,
  let*, title?, (assert | report | extends | p)+)))
}
```

properties

```
properties = element properties { property* }
```

```
property = element property {  
    attribute id { xsd:ID },  
    attribute role { roleValue }?,  
    attribute scheme { text }?,  
    { foreign & (text | name | value-of | emph | dir | span)* }
```

Update Annex D D.2 RELAX NG schema for SVRL

```
# only rules that are fired are reported,
```

```
fired-rule =
```

```
element fired-rule {  
    attribute id { xsd:ID }?,  
    attribute name { text }?,  
    attribute context { text },  
    attribute role { xsd:NMTOKEN }?,  
    attribute flag { xsd:NMTOKEN }?,  
    empty  
}
```

```
# only failed assertions are reported
```

```
failed-assert = element failed-assert {  
    attlist.assert-and-report,  
    diagnostic-reference*,  
    property-reference*,  
    human-text  
}
```

```
# only successful asserts are reported
```

```
successful-report = element successful-report {  
    attlist.assert-and-report,  
    diagnostic-reference*,  
    property-reference*,  
    human-text
```

```
}
```

```
# property-reference
```

```
property-reference = element property-reference {  
    attribute property { xsd:NMTOKEN },  
    attribute role { text }?,  
    attribute scheme { text }?,  
    human-text  
}
```

```
# human-text
```

```
human-text = element text {  
    attribute xml:space {text}?,  
    attribute xml:lang {text}?,  
    attribute see {text}?,  
    attribute icon {text}?,  
    attribute fpi { text }?,  
    rich-text }
```

```
# rich-text
```

```
rich-text = {(foreign | dir | span | emph | text)* }
```

```
# directionality
```

```
dir = element dir {  
    attribute class {text}?,  
    attribute dir {text}?,  
    text }
```

```
# emphasis
```

```
emph = element emph {  
    attribute class {text}?;  
    text }
```

```
# arbitrary markup
```

```
span = element span {  
    attribute class {text},  
    text }
```

```
# foreign
foreign =
    foreign-attributes, foreign-element*
```

```
foreign-attributes =
attribute * { text }*
```

```
foreign-element = element * - svrl:* {
(attribute * { text }
| foreign-element
| text)*
}
```

Add new Annex

Annex X (Informative)

Use of Schematron Properties

In the following examples, the property elements given should be placed inside the properties element in the document below. It gives examples of properties usable by a validation application on an assert element and properties usable for information set augmentation on a report element.

```
<sch:schema ...>
```

```
  <sch:title>Example schema of annotation with</sch:title>
```

```
  <sch:pattern>
```

```
    <sch:rule context="assetValue">
```

```
      <sch:assert test="true()" properties="precision3 non-negative-decimal iso8859-1">
```

The element assetValue should be a non-negative decimal number, with three digits of precision, in Australian dollars.

```
    </sch:assert>
```

```
    <sch:report test="true()"
```

```
      properties="australianDollar precision3a thisElement thisValue ">
```

The element assetValue should be a non-negative decimal number, with three digits of precision, in Australian dollars.

```
    </sch:report>
```

```
  </sch:pattern>
```

```
  <sch:properties>
```

```
  ...
```

```
  </sch:properties>
```

```
</sch:schema>
```

In the following examples, the values of the role attributes are neither defined nor reserved by this standard.

X.1 Annotation with controlled vocabulary information

```
<sch:property id="australianDollar" scheme="ISO4217" role="currency">AUD</sch:property>
```

X.2 Annotation with static information

```
<sch:property id="precision3" role="precision" >3</sch:property>
```

X.3 Annotation with static structured information

```
<sch:property id="precision3a" >  
  <precision>3</precision>  
</sch:property>
```

X.4 Type annotation with XSD Simple Type

```
<sch:property id="non-negative-decimal" >  
  <xsd:simpleType type="xs:decimal">  
    <xsd:maxInclusive value="0" />  
    <xsd:fractionDigits value='2' />  
  </xsd:simpleType>  
</sch:property>
```

X.5 Type annotation with dynamic information

```
<sch:property id="thisValue" role="value">  
  <sch:value-of select="." />  
</sch:property>
```

X.6 Type annotation with dynamic structured information

This can be used to draw in elements as well as values from the instance.

```
<sch:property id="thisElement" role="contents">  
  <xslt:copy-of select="." xmlns:xslt="" />  
</sch:property>
```

X.7 Annotation with ISO19757-7 character repertoire reference

```
<sch:property id="iso8859-1" role="repertoire"  
  xmlns:cdrl="">  
  <cdrl:charref href="http://www.eg.com/cdrl/iso8859-1.cdr" />
```

</sch:property>

X.8 SVRL document

Let the instance be the single element which conforms to the properties above.

```
<assetValue>1000.00</assetValue>
```

The following is an SVRL document which transfers the properties from the report element.

```
<svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl">
```

```
<svrl:active-pattern />
```

```
<svrl:fired-rule context="assetValue" />
```

```
<svrl:successful-report test="false()" location="/assetValue" >
```

```
<svrl:property-reference
```

```
  property="australianDollar" scheme="ISO4217" role="currency">AUD</sch:property>
```

```
<svrl:property-reference property="precision3a" >
```

```
  <precision>3</precision>
```

```
</svrl:property-reference>
```

```
<svrl:property-reference property="thisValue" role="value">1000.00</property>
```

```
<svrl:property-reference property="thisElement" role="contents">
```

```
  <assetValue>1000.00</assetValue>
```

```
</svrl:property>
```

```
<svrl:text>The element assetValue should be a non-negative decimal number, with three digits of precision, in Australian dollars.</svrl:text>
```

```
</svrl:successful-report>
```

```
</svrl:schematron-output>
```

3) Compound and subordinate documents.

Summary: An attribute pattern/@documents has an XPath so that patterns can be selected on documents located from paths constructed from information in the original instance. The terms compound document is used because of familiarity, rather than HyTime's Bounded Information Set for example, and the notion of reachability has been avoided. For example

```
<sch:pattern documents="/*[@xlink:type='simple'][@xlink:behaviour='embed']/@href" >
```

- i) To Terms and Definitions add

compound document: a notional instance document which has been divided into the original instance and a number of subordinate well-formed XML documents. These subordinate documents may also contain schema-like information.

- ii) To main text To body, clause 5.4.9 pattern element Insert as second paragraph:

The optional documents attribute provides IRIs of subordinate document the rule contexts are relative to. If the expression evaluates to more than one IRI, then the pattern is sought in each of the documents. The documents attribute is evaluated in the context of the original instance document root.

NOTE: The documents attribute was not part of IS19757-3:2006.

To body, clause 6.4 Query Language Binding Add to list "a query language binding may also provide..."

The pattern document-selecting query language

- iii) To Schematron RELAX NG schema Add attribute to element pattern

documents { pathValue }?,

- iv) To SVRL RELAX NG schema Add attribute to element active-pattern

documents { text }?,

- v) To default QLB and prospective EXSLT QLB Add list item

The documents attribute of the pattern element shall be interpreted according to the Production 1 of <xref to="xslt-rec"/>, as returning one or more strings that are evaluated using the document() function.

- vi) To QLB to prospective XSLT2, XPath2 QLB add

The documents attribute of the element shall be interpreted according to the Production 1 of <xref to="xslt2-rec"/>, as returning a sequence of strings that are evaluated using the document() function.

- vii) To Annex D SVRL, D.1 Description, second paragraph, add

Elements related to the same subordinate document shall be grouped together.

viii) To s6.3 Schema semantics change para 3 from “a document is valid” to “a compound document is valid”

Update Annex E Design Requirements. To numbered list replace

2. Support progressive validation

With

2. Support progressive validation of compound documents

4) Inclusions.

Summary: The extends element is enhanced to also allow a merge functionality on external documents. This enhancement fits in with the existing semantics of the extends elements. As well, the definition of the include element is loosened to allow fragments identifiers, as used by the main implementation.

Replace **5.4.3 extends** element text with the following:

The extends element allows reference to the contents of other declarations. The extends element shall either have an href attribute or a rule attribute but not both.

An href attribute shall be an IRI reference to an external well-formed XML document or to an element in external well-formed XML document that is Schematron element of the same type as the parent element of the extends element. The contents of that referenced element shall be inserted in place of the extends element.

In such a case, the relative position of elements in the post-inclusion document may be to that extent invalid against the schema for Schematron Annex A, however other schema constraints such as containment shall still apply.

Abstract rules are named lists of assertions without a context expression. A rule attribute shall reference an abstract rule. The current rule uses all the assertions from the abstract rule it extends.

NOTE: The capability to extend with multiple elements was not part of IS19757-3:2006.

5.4.4 include element

The required href attribute shall be an IRI reference to an well-formed XML document or to an element in a well-formed XML document.

The referenced element shall be inserted in place of the include element. The referenced element shall be a type which is allowed by the grammar for Schematron at the location of the include element.

5) Misc

- 1) To Introduction, after para 6 add

This edition is backwards compatible with IS19757-3:2006, supercedes it, and augments it with the following capabilities: patterns may validate different documents, the inclusion mechanism has been supplemented by an enhanced extension mechanism, assertions may have linked properties, and SVRL may take richer text. As well, this edition provides extra query language bindings, in particular for XSLT2.

- 2) Change URI to IRI in **5.5.10 see attribute** Change **s5.2 Namespace** for text “A foreign attribute is an attribute with a name whose namespace URI” to “A foreign attribute is an attribute with a name whose namespace URI [IRI]”. Change **s5.4.7** “The required `uri` attribute is a namespace URI.” To “The required `uri` attribute is a namespace URI [IRI].” Add normative reference to IRI RFC.

- 3) Change generic references to XSLT to XSLT1.

- 4) Section 6.4 Query Language Binding. Remove “xslt1.1” from the list.

- 5) Change list item 2, list 1, Annex A to add a sentence on text nodes, resulting in:

The rule context is interpreted according to the Production 1 of XSLT. The rule context may be the root node, elements, attributes, comments and processing instructions. An implementation may allow the rule context to be text nodes at user option however implementations may reject or fail to implement schemas which specify text nodes.

- 6) Remove first line of formal definition in s6.3 Schema Semantics

Editor: Not recommended for acceptance by WG1

- 7) Revise references section to add:

- a. XSLT2, xpath2, xpath2-model, xpath2-functions
- b. STX (if QLB added)
- c. EXSLT
- d. IRI