

Schema Binding Proposal

Sandy Gao
Valentina Popescu

1

2

3

4

5 1 Terminology 1

6 2 Problem definition 2

7 3 Requirements..... 3

8 3.1 Support schema composition..... 3

9 3.2 Support schema versioning 3

10 3.3 Deterministic 3

11 3.4 Full schema support..... 3

12 3.5 Schema document exchange 3

13 4 Constraints 4

14 4.1 Support access to schema documents outside of SML-IF 4

15 4.2 Ignorable schema locations 4

16 4.3 Include definition and instance documents as-is 4

17 4.4 Lazy schema assembly..... 4

18 4.5 Support reference constraints..... 4

19 5 Interoperability Approach..... 5

20 6 Schema binding proposal 5

21 6.1 An Example..... 5

22 6.2 Solution to the Example 6

23 6.3 Default Schema..... 6

24 6.4 Formal Proposal..... 7

25 6.5 Proposal Analysis..... 8

26 7 Implementation Cost 9

27 8 Acknowledgement..... 9

1 Terminology

- 29 **Schema document:** an <xs:schema> element; can be an XML fragment
- 30 **Schema:** a set of schema components; a schema is normally (but not required to be)
- 31 constructed from one or more schema documents
- 32 **Schema component:** an element declaration or a type definition or a particle or ...
- 33 **Include:** A schema document can include another schema document using <xs:include>.
- 34 Both schema documents contribute to the same schema; and both correspond to
- 35 schema components from the same target namespace (or no namespace). If the
- 36 included schema document does not have a target namespace, namespace of
- 37 the including schema document is used.

- 38 **Redefine:** Similar to include, but use <xs:redefine>, and the redefining schema document
39 can replace certain included components with new components.
- 40 **Import:** Allows the importing schema document to refer to components from the imported
41 namespace (or no namespace), which must be different from the importing
42 schema document's target namespace. If the combination of the "namespace"
43 attribute and the "schemaLocation" attribute on <xs:import> resolves to a schema
44 document, then the resulting schema also includes components from the
45 imported schema document.
- 46 **Schema composition:** (In this document) construct a single schema from multiple schema documents,
47 using the above include, redefine and/or import mechanisms.
- 48 **Note:** "a schema" is not equal to "a schema document"!

49 2 Problem definition

50 In performing SML model validation over the SML model packaged in an SML-IF instance, associations
51 between XML Schema definition documents and instance documents need to be drawn, both to
52 completely validate XML Schema documents themselves (to make sure they produce valid schemas) and
53 to establish schema-validity of the instance documents.

54 Schema documents can be connected with other schema documents using composition features
55 provided by XML Schema. This includes <xs:include>, <xs:redefine>, and <xs:import>. A schema
56 document's validity may depend on other schema documents it includes/redefines/imports, or even other
57 schema documents that include/redefine/import it.

58 When validating a model instance document, a precise list of schema documents need to be associated
59 with it for a "schema" and the instance document is schema-assessed using this schema.

60 The XML Schema 1.0 specification provides more flexibility in constructing the schema used for
61 assessment than is appropriate for the semantics defined by SML and SML-IF validation:

- 62 • It allows processor latitude in terms of locating schema documents (resolving namespace and
63 schema location attributes) and composing schema documents together to form a single schema.
- 64 • Schema location attributes can be ignored in some cases ("xsi:schemaLocation" in instance
65 documents and "schemaLocation" on <xs:import>); and allowed to "fail to resolve" in others
66 ("schemaLocation" attribute on <xs:include> and <import>). Known schema and SML
67 implementations behave differently with respect to how/whether they process schema location
68 attributes.
- 69 • Multiple imports of the same namespace allow all but the first one to be ignored.

70 So it is clear that we have no hope of guaranteeing general case interoperability using anything based
71 only on XML Schema given the constraints above, and SML-IF needs to specify how to determine such
72 associations.

73 NOTE: this proposal is only about SML model validation, and not SML-IF validation (against the IF
74 schema). Unless otherwise indicated, "validation/validity" in the following sections is always about SML
75 model validation.

76 **3 Requirements**

77 **3.1 Support schema composition**

78 There are many real-life schemas that are constructed from multiple schema documents. Such schemas
79 may span multiple namespaces (hence the need for import); components from each namespace may be
80 further divided into multiple schema documents (hence the need for include).

81 Schema has a feature often referred to as “chameleon include”. This means that a schema document
82 with a target namespace includes or redefines another schema document without a target namespace,
83 and the result is as if the included/redefined document had a target namespace that’s the same as the
84 including/redefining document. SML-IF needs to support this usage scenario.

85 **3.2 Support schema versioning**

86 Schema authors can’t anticipate how their schemas will be used, hence the need to evolve schemas.
87 There are different versioning scenarios. There are cases where minor modifications of older versions
88 suffice, and redefine can be used. Some schemas need to be rewritten to accommodate new
89 requirements, and new namespace may or may not be introduced (compatibility is often a good reason
90 for not changing namespaces). There are also cases where there are generic and specific versions (as
91 opposed to previous and next versions), which often co-exist and share the same namespace.

92 To support this, SML-IF needs to be able to package in the same SML-IF instance different versions of
93 the same schema in the same namespace.

94 **3.3 Deterministic**

95 For a given SML-IF instance, there MUST be no ambiguity in determining how schema documents (that
96 are included in this instance) are connected using <xs:include>, <xs:redefine>, and <xs:import>, and
97 therefore MUST be no ambiguity in determining which schema documents are used to form a schema
98 against which a given instance document is validated.

99 **3.4 Full schema support**

100 Being a generic validation language, SML supports all schema features. Being a mechanism to transmit
101 SML models, SML-IF also needs to support full schema features, especially <xs:include>, <xs:redefine>,
102 and <xs:import>. For example, in an SML model, if an instance document I is validated against a schema
103 formed from a schema document A, which redefines schema document B, then it MUST be possible to
104 transmit I, A, and B in an SML-IF instance and maintain their relationship.

105 **3.5 Schema document exchange**

106 An SML-IF document can contain XML Schema documents within its definition documents that are
107 attached for exchange purposes only. These documents are not intended to be used for XML Schema
108 validity assessment of the model instance documents. SML-IF needs to support this use case and
109 ensure that documents of this purpose do **not** participate in model instance document validation.

110 This is analogous to the case we already have for rule documents, except rule documents do not have a
111 “bind to all” default as we are contemplating for XML Schema documents. Any new types of definition
112 documents added in the future will have to address similar concerns, whose syntax will be influenced by
113 the default binding (all or none).

114 Note that for both schema documents and Schematron rule documents that are not bound to any
115 instances, their validity should still be checked when assessing SML model validity, as required by SML,
116 which has:

- 117 • Each XML Schema document in the model's definition documents MUST satisfy the conditions
118 expressed in Errors in Schema Construction and Structure (§5.1). [XML Schema Structures]
- 119 • Each Schematron document in the model's definition documents MUST be a valid Schematron
120 document [ISO/IEC 19757-3]

121 **4 Constraints**

122 **4.1 Support access to schema documents outside of SML-IF**

123 We do not want to force all schemas necessary to validate the model instance documents packaged by a
124 single SML-IF instance to be included by value in every SML-IF instance. It is not clear this would even
125 be sensible in a repository interchange scenario, let alone the more general case of usage scenarios
126 some have mentioned for SML-IF like web services message exchanges.

127 **4.2 Ignorable schema locations**

128 We cannot require honoring of `xsi:schemaLocation` and `xsi:noNamespaceSchemaLocation` in instance
129 documents or `schemaLocation` on `<xs:import>`, because

- 130 • Some existing implementations ignore them
- 131 • Honoring schema location in instance documents may have security consequences

132 Schema specification does require that processors attempt to resolve schema locations specified on
133 `<xs:include>` and `<xs:redefine>`. It is not an error for such attempt to fail for `<xs:include>`. It is an error
134 when `<xs:redefine>` contains non-annotation content.

135 It's more flexible for `<xs:import>`. Schema allows any strategy for processors to locate components to
136 import, based on either or both of the namespace and the schema location.

137 **4.3 Include definition and instance documents as-is**

138 SML-IF instance producers may not have control over the content of the schemas necessary for
139 validation of model instance documents, where "control" means what is coded in the files. I.e. there will
140 be cases where `xs:import` and `xs:include` are coded, with and without `schemaLocation`, and multiple files
141 containing schema components for the same namespace will be observed.

142 **4.4 Lazy schema assembly**

143 Schema specification allows schemas to be assembled lazily. A partial schema can be used to validate
144 an instance document, and more components can be added to the schema during the validation, as long
145 as the new components don't change the validation result of information items that are already validated.

146 This is sometimes not easy to enforce, but a consequence of "supporting full schema" implies that SML-IF
147 validation cannot violate this constraint.

148 **4.5 Support reference constraints**

149 Reference-related constraints (`targetElement`, `targetType`, acyclic, SML identity constraints) need to be
150 properly supported. When 2 documents **A** and **B** are connected by an SML reference, these constraints
151 require the ability to determine whether a component from the schema used to assess **A** is identical to a
152 component from the schema used to assess **B**. The schema spec doesn't define identity of components
153 across multiple schemas. The same source declaration may produce totally different components in
154 different schemas. So to check those reference-related constraints, related instance documents MUST
155 be validated using the same schema.

5 Interoperability Approach

We divide the universe of SML-IF documents into two disjoint subsets:

- A set that have all schema documents included, by value (smlif:data) and/or by reference (smlif:locator), in the SML-IF instance; the “schema-complete set”
- All other SML-IF documents; the “schema-incomplete set”

It is necessary for a producer to declaratively distinguish between these two cases, since it is not always possible to distinguish based on the content alone. For example, XML Schema allows xs:include’s schema location attribute’s value to not resolve, although the value is required. This can be done by introducing a “schemaComplete” attribute on the <smlif:definitions> element to indicate whether this SML-IF instance includes all necessary definition documents.

When this attribute is specified with an actual value “true”, then for the instance to be valid, its schema definition documents and instance documents can only refer to either built-in components or components from definition documents included in the instance. “Built-in” components include:

- 4 xsi: attributes (defined by XML Schema)
- all schema built-in types (xs:anyType and simple types defined in XML Schema Part 2)
- sml:ref attribute declaration
- sml:uri element declaration

Remember, this is not trying to say that SML-IF document instances in the schema-incomplete set are now invalid. It does say that SML-IF cannot guarantee interoperability for the schema-incomplete set.

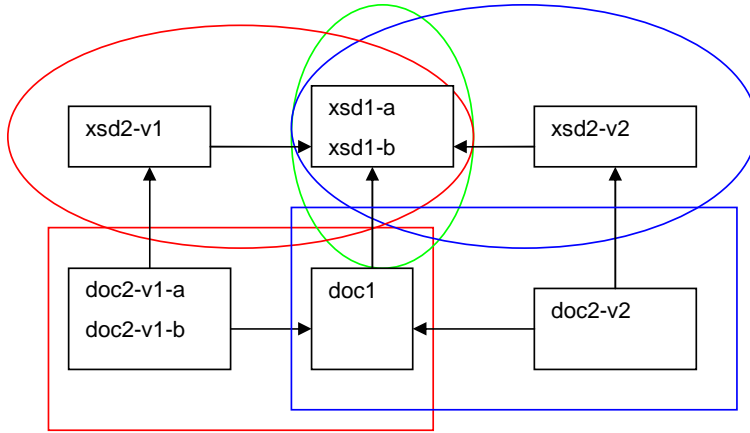
6 Schema binding proposal

6.1 An Example

(See the picture next page) Assume an SML model packaged in an IF document has 4 schema documents: **xsd1-a** and **xsd1-b** have target namespace **ns1**, and **xsd2-v1** and **xsd2-v2** have target namespace **ns2**, where **xsd2-v1** and **xsd2-v2** are conflicting versions of the same schema (same target namespace). There are 4 instances: **doc1** uses **xsd1-a** and **xsd1-b**; **doc2-v1-a** and **doc2-v1-b** uses **xsd2-v1**, and **doc2-v2** uses **xsd2-v2**. All **doc2-*** instances have SML references to **doc1**, and their references have targetType constraints, pointing to a component in **ns1**.

To check targetType, **doc2-v1-a**, **doc2-v1-b** and **doc1** must be validated using the same schema (**xsd1-a + xsd1-b + xsd2-v1**); similarly, **doc2-v2** and **doc1** must be validated using the schema from **xsd1-a + xsd1-b + xsd2-v2**. More concretely, in the following picture, instances in the red rectangle are validated using the schema built from schema documents in the red oval; and instances in the blue rectangle are validated using the schema built from the blue oval.

Note that **doc1** is validated twice using 2 different schemas. **doc1** may also be validated against only xsd1; this is up to the model author to specify.



191

192 **6.2 Solution to the Example**

```

193 <schemaBindings>
194   <!-- Each "schemaBinding" element corresponds to a schema and model
195        instance documents that are assessed against this schema -->
196   <schemaBinding>
197     <!-- all "namespaceBinding" children together build the schema -->
198     <namespaceBinding namespace="ns1" aliases="xsd1-a xsd1-b"/>
199     <namespaceBinding namespace="ns2" aliases="xsd2-v1"/>
200     <!-- list all applicable instances; same as for rule bindings -->
201     <documentAlias>doc1</documentAlias>
202     <documentAlias>doc2-v1-a</documentAlias>
203     <documentAlias>doc2-v1-b</documentAlias>
204   </schemaBinding>
205   <schemaBinding>
206     <namespaceBinding namespace="ns1" aliases="xsd1-a xsd1-b"/>
207     <namespaceBinding namespace="ns2" aliases="xsd2-v2"/>
208     <documentAlias>doc1</documentAlias>
209     <documentAlias>doc2-v2</documentAlias>
210   </schemaBinding>
211 </schemaBindings>
212 <definitions schemaComplete="true">
213   <!-- schema documents for xsd1-a, xsd1-b, xsd2-v1, xsd2-v2 -->
214 </definitions>

```

215

216 **6.3 Default Schema**

217 [There are cases where most instance documents use the same schema. It's desirable to have a default](#)
218 [schema to cover this case, instead of having to have a <schemaBinding> that lists all those instances.](#)

219 [For example, if an IF document contains 3 schema documents: ns1.xsd, ns2.xsd, and ns2-exchange.xsd,](#)
220 [where the latter 2 documents share the same target namespace, but ns2-exchange.xsd is meant to be](#)
221 [exchanged only and should not be considered as part of the schema that governs instance documents.](#)
222 [This can be achieved using the following syntax:](#)

```

223 <schemaBindings>
224 <!-- The "defaultSchema" element corresponds to a schema that governs
225 all instance documents *not* included in any "schemaBinding". -->
226 <defaultSchema>
227 <!-- all "namespaceBinding" children together build the schema -->
228 <namespaceBinding namespace="ns1" aliases="ns1.xsd"/>
229 <namespaceBinding namespace="ns2" aliases="ns2.xsd"/>
230 </defaultSchema>
231 </schemaBindings>
232 "defaultSchema" can be used together with "schemaBinding" as a default to cover instances documents
233 that are not included in any "schemaBinding".

```

6.4 Formal Proposal

1. Change the IF document structure to add the following (new content highlighted):

```

236 <model>
237 ...
238 <ruleBindings> ?
239 <ruleBinding> *
240 <documentAlias="xs:anyURI"/> ?
241 <ruleAlias="xs:anyURI"/>
242 </ruleBinding>
243 </ruleBindings>
244 <schemaBindings> ?
245 <defaultSchema> ?
246 <namespaceBinding/> * <!-- a single namespace name
247 and list of schema document aliases -->
248 </defaultSchema>
249 <schemaBinding> *
250 <namespaceBinding/> * <!-- a single namespace name
251 and list of schema document aliases -->
252 <documentAlias/> * <!-- a list of instance document aliases -->
253 </schemaBinding>
254 </schemaBindings>
255 ...
256 <definitions schemaComplete="xs:boolean"> ?
257 ...
258 </model>

```

Deleted: ¶

Deleted: and

Deleted: ¶

The details of the preceding XML syntax, e.g. whether the data is contained in attributes or elements, is fully negotiable. The XML above simply captures enough to have the discussion that follows.

2. For every schema binding **SB** in the model, i.e. every "/model/schemaBindings/schemaBinding" element (using XPATH notation):
 - 2.1. Compose a schema using all documents specified under all **SB**'s <namespaceBinding> children
 - 2.2. Whenever there is an <import> for a namespace **N**
 - 2.2.1. If there is a <namespaceBinding> child of **SB** whose "namespace" matches **N**, then components from schema documents listed in the corresponding "aliases" are used. As with rule bindings, URI prefixing is used for matching schema document aliases.

Note: at most one <namespaceBinding> is allowed per namespace **N** within a given **SB**. If more than one namespace binding exists for the namespace as part of a single schema binding, the SML-IF instance is in error.

Note: if the set of aliases for namespace **N** is empty, the namespace has no schema documents defining it in the schema binding.

- 273 2.2.2. Otherwise if there are schema documents in the IF whose targetNamespace is **N**, then
 274 components from all those schema documents are used
- 275 2.2.3. Otherwise
- 276 2.2.3.1. If a schema-complete document (/model/definitions/@schemaComplete=true) is
 277 being processed, then no component from N (other than built-ins) is included in the
 278 schema being composed
- 279 2.2.3.2. Otherwise, it is implementation-defined whether the processor tries to retrieve
 280 components for N from outside the SML-IF instance
- 281 2.3. Whenever there is an <include> or <redefine>, the schemaLocation is used to match aliases of
 282 schema documents, as with base SML-IF.
- 283 2.3.1. If there is a schema document in the IF matching that alias, then that document is used
- 284 2.3.2. Otherwise
- 285 2.3.2.1. If it's a schema-complete set, then the <include> or <redefine> is unresolved (which
 286 is allowed by XML Schema validity assessment rules)
- 287 2.3.2.2. Otherwise, it's implementation-defined whether it tries to resolve <include> or
 288 <redefine> to schema documents outside the IF
- 289 2.4. The list of <documentAlias> documents are assessed against this *same* schema. targetXXX
 290 and identity constraints can now be checked. Similar to <documentAlias> under <ruleBinding>
 291 elements, each <documentAlias> can refer to multiple documents via URI prefixing.

292 3. If <defaultSchema> is present, then compose a schema from it following rules 2.1 to 2.3 above;
 293 otherwise compose a schema using *all* schema documents included in the IF. Then use this
 294 schema to assess those instance documents that are not included in any <schemaBinding>.

Deleted: C
 Deleted: , t

295 Note: in the common case where match-all namespace matching is the desired result, this is achieved by
 296 omitting <schemaBindings>, i.e. without introducing any additional complexity into the SML-IF instance.

297 Note: one implication of this formulation is that the Schema document exchange requirement of section
 298 3.5 is supported. This would be done by explicitly binding /model/instances/* to a schema binding that
 299 excludes the exchange-only schemas. The model instance documents may still contain information items
 300 from namespace(s) in the exchange-only schemas, however those schema documents would not be
 301 used to assess schema validity of the model instance documents.

Deleted: 3.5

302 6.5 Proposal Analysis

- 303 • Great synergy with <ruleBindings>
 - 304 ○ It works in a way very similar to Schematron rules. You associate a schema (built from a
 305 set of schema documents) with a set of instance documents
- 306 • Handles all the requirements
 - 307 ○ Supports schema composition: chameleon included documents is supported by removing
 308 them from the corresponding <namespaceBinding> (whose "namespace" attribute is
 309 absent)
 - 310 ○ Supports schema versioning: multiple versions can be specified in different
 311 <schemaBinding> elements
 - 312 ○ Deterministic: the association between instances and schemas is deterministic
 - 313 ○ Full schema support: <include/redefine/import> are all supported
 - 314 ○ Schema document exchange: similar to chameleon included documents, exchange-only
 315 documents can also be omitted from the corresponding <namespaceBinding>
- 316 • Meets all the constraints

- 317 o Supports access to schema documents outside of SML-IF: when schemaComplete=false,
318 processors are allowed to use external schema documents
- 319 o Ignorable schema locations: all xsi:schemaLocation attributes can be ignored
- 320 o Includes definition and instance documents as-is: no need to modify any included
321 document; document aliases are used.
- 322 o Lazy schema assembly: the schema is known up-front; no need to handle lazy assembly
- 323 o Supports reference constraints: instances specified under the same <schemaBinding>
324 use the same schema, so reference constraints can be checked.
- 325 • Simple to understand
- 326 • This has may Note that the "trivial case" is also handled by bullet 2. That is, there is no
327 <schemaBinding> and all instance documents are assessed against the same schema.

328 **7 Implementation Cost**

329 We have to assume that all existing schema processors are capable of handling the "namespace
330 matching" approach. That is, they can compose a schema from a list of schema documents.

331 This approach should be straightforward to handle. All the SML processor needs to do is to compute a
332 list of schema documents based on schema documents mentioned in <schemaBinding> and give that list
333 to the schema processor.

334 The "Explicit Binding" approach from earlier iterations of this proposal had what we believe is equivalent
335 function, but was eliminated because it also had greater complexity (four levels of binding to sift through,
336 instead of the two used here, i.e. schema binding and match-all namespace matching).

337 **8 Acknowledgement**

338 John Arwe, Bassam Tabarra, Harm Sluiman, and Pratul Dubish all provided useful input into the
339 formulation of this document. This does not imply their endorsement of the proposal.