Schema Binding Proposal

Sandy Gao
Valentina Popescu

1 Terminology

Schema document:	an <xs:schema> element; can be an XML fragment
Schema:	a set of schema components; a schema is normally (but not required to be) constructed from one or more schema documents
Schema component:	an element declaration or a type definition or a particle or …
Include:	A schema document can include another schema document using <xs:include>. Both schema documents contribute to the same schema; and both correspond to schema components from the same target namespace (or no namespace). If the included schema document does not have a target namespace, namespace of the including schema document is used.
Redefine:	Similar to include, but use <xs:redefine>, and the redefining schema document can replace certain included components with new components.
Import:	Allows the importing schema document to refer to components from the imported namespace (or no namespace), which must be different from the importing schema document's target namespace. If the combination of the "namespace" attribute and the "schemaLocation" attribute on <xs:import> resolves to a schema document, then the resulting schema also includes components from the imported schema document.
Schema composition:	(In this document) construct a single schema from multiple schema documents, using the above include, redefine and/or import mechanisms.

Note: "a schema" is ***not*** equal to "a schema document"!

2 Problem definition

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

Deleted: validating

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

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

Deleted: an

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

- It allows processor latitude in terms of locating schema documents (resolving namespace and schema location attributes) and composing schema documents together to form a single schema.

- Schema location attributes can be ignored in some cases ("xsi:schemaLocation" in instance documents and "schemaLocation" on <xs:import>); and allowed to "fail to resolve" in others ("schemaLocation" attribute on <xs:include> and <import>). Known schema and SML implementations behave differently with respect to how/whether they process schema location attributes.

- Multiple imports of the same namespace allow all but the first one to be ignored.

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

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

3 Requirements

3.1 Support schema composition

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

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

3.2 Support schema versioning

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

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

3.3 Deterministic

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

3.4 Full schema support

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

3.5 Schema document exchange

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

Deleted: ***Definition***

Deleted: documents

Deleted: purpose

Deleted: have no relationship with the attached instances and they

Deleted: validation purposes within the SMLIF document.

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

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

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

4 Constraints

4.1 Support access to schema documents outside of SML-IF

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

4.2 Ignorable schema locations

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

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

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

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

4.3 Include definition and instance documents as-is

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

4.4 *Lazy schema assembly*

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

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

4.5 *Support reference constraints*

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

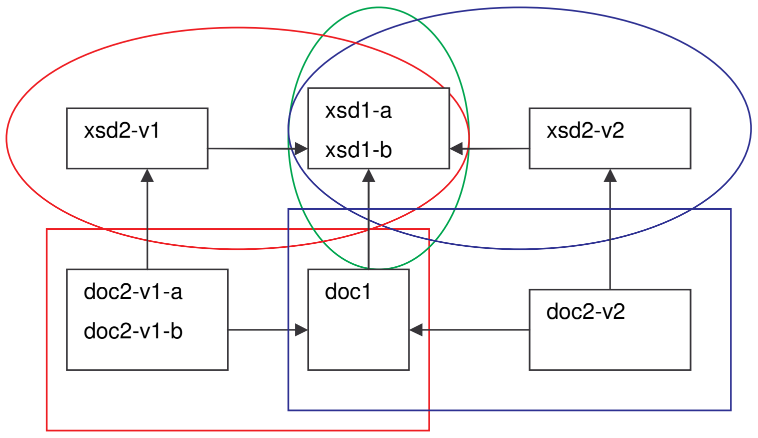
Deleted: More precisely, if during validation of schema or instance document in the IF instance, no attempt is made to retrieve schema documents outside of the set of schema documents included in the IF instance, or to retrieve schema components outside of the set of components constructed from schema documents included in the IF instance, then the IF instance is said to be within the "complete set".¶
The only exception to the above rule is for the "built-in" components. These components are always available in SML-IF processors. They can be referred to freely, without the need to explicitly load them.

Deleted: <#>does sml:refType belong here (depends on how we resolve bug 4673)?¶

Deleted: There are several ways to specify how to locate schema documents that are used to construct a schema. Each approach needs to answer the following questions:¶
<#>How to handle <xs:include>¶
<#>How to handle <xs:redefine>¶
<#>How to handle <xs:import>¶
<#>How to locate the correct version of each schema document used to validate a given instance¶
In XML Schema, schema composition is achieved using one or more of <xs:include>, <xs:redefine>, and <xs:import>. Schema doesn't provide details about how to assemble a schema from unrelated schema documents. To achieve interoperability, the following subsections always describe various approaches as having a synthetic "root" schema document whose transitive closure over include/redefine/import covers all schema documents that are supposed to be used. This is similar to how XSLT 2.0 determines how schemas should be assembled. Implementations are not required to implement schema assembly using synthetic schema documents.¶
<#>Namespace matching¶
In the SML-IF instance, there is a namespace to schema document mapping. Such a mapping can be established by, for example, traversing all schema documents and gathering their target namespace attributes.¶
If for a given namespace, there is more than one schema document, then a choice must be made about which to use. This text assumes that all documents for the namespace are logically merged, as if the consumer synthesized another schema document with this namespace as its target namespace and includes ... [1]

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.



6.2 Solution to the Example

```
    <schemaBindings>
      <!-- Each "schemaBinding" element corresponds to a schema and model
           instance documents that are assessed against this schema -->
      <schemaBinding>
        <!-- all "namespaceBinding" children together build the schema -->
        <namespaceBinding namespace="ns1" aliases="xsd1-a xsd1-b"/>
        <namespaceBinding namespace="ns2" aliases="xsd2-v1"/>
        <!-- list all applicable instances; same as for rule bindings -->
        <documentAlias>doc1</documentAlias>
        <documentAlias>doc2-v1-a</documentAlias>
        <documentAlias>doc2-v1-b</documentAlias>
      </schemaBinding>
      <schemaBinding>
        <namespaceBinding namespace="ns1" aliases="xsd1-a xsd1-b"/>
        <namespaceBinding namespace="ns2" aliases="xsd2-v2"/>
        <documentAlias>doc1</documentAlias>
        <documentAlias>doc2-v2</documentAlias>
      </schemaBinding>
    </schemaBindings>
    <definitions schemaComplete="true">
      <!-- schema documents for xsd1-a, xsd1-b, xsd2-v1, xsd2-v2 -->
    </definitions>
```

Deleted: *<#>Explicit schema binding¶*
With this approach, the SMLIF producer provides an explicit mapping of namespace to schema document alias(s). The following bindings should be supported:¶
<#>6.2.1 Global schema binding¶
The mapping for each namespace to a set of schema documents can be defined at the SML-IF instance level by specifying explicit mappings for namespaces, to override the default namespace "match all" approach described in section 6.1.¶
This explicit binding of a namespace with a set of schema documents can be used to fulfill requirement 3.5. For the situation when a definition document is attached to the SMLIF with the intention of just exchanging the data, an explicit binding of its targetNamespace with the correct schema document(s) can solve the match-all namespace matching approach issue identified in section 7, since the global binding will override the match-all namespace matching approach.¶
<#>6.2.2 Definition-level schema binding¶
Each schema document can provide its own namespace to schema document alias mapping, which overrides any global level binding for these namespaces. That is, for a given namespace, if it's specified in the definition-level binding then use it; otherwise if it's specified in the global binding then use this one; otherwise use the default namespace to schema document mapping as defined in section 6.1¶

6.3 Formal Proposal

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

```
<model>
   ...
   <ruleBindings> ?
     <ruleBinding> *
       <documentAlias="xs:anyURI"/> ?
       <ruleAlias="xs:anyURI"/>
     </ruleBinding>
   </ruleBindings>
   <schemaBindings> ?
     <schemaBinding> *
       <namespaceBinding/> *  <!--
                                    a single namespace name and
                                    and list of schema document aliases -->
       <documentAlias/> *     <!--
                                    a list of instance document aliases -->
     </schemaBinding>
   </schemaBindings>
   ...
   <definitions schemaComplete="xs:boolean"> ?
   ...
</model>
```

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.

2.2.2. Otherwise if there are schema documents in the IF whose targetNamespace is **N**, then components from all those schema documents are used

2.2.3. Otherwise

2.2.3.1. If a schema-complete document (/model/definitions/@schemaComplete=true) is being processed, then no component from N (other than built-ins) is included in the schema being composed

2.2.3.2. Otherwise, it is implementation-defined whether the processor tries to retrieve components for N from outside the SML-IF instance

2.3. Whenever there is an <include> or <redefine>, the schemaLocation is used to match aliases of schema documents, as with base SML-IF.

2.3.1. If there is a schema document in the IF matching that alias, then that document is used

Deleted: Global schema binding works only when a namespace is always defined by the same definition document(s), for all schema instances. For the case when two instances refer to the same namespace but the namespace is defined by two different definition documents, an instance-level binding will be applied (instance A refers to namespace ns, defined in definition document A1; instance B refers to the same namespace ns defined in definition document B1.)¶
<#>6.2.3 Instance-level schema binding¶
Each instance document can provide its own namespace to schema document alias mapping, which will override any global or definition-level bindings for these namespaces.. That is, for a given namespace, if it's specified in the instance-level explicit binding, then use it; otherwise if it's specified in a definition-level binding at the schema document level (for some schema documents that's already used by this instance), then use it; otherwise if it's specified as a global schema binding, then use it; otherwise use the default namespace to schema document alias mapping.¶
If the listed schema documents include or redefine other schema documents, then they are resolved using aliases; all imports are ignored, because the explicit list should have included all of them.¶
¶
<#>6.2.4 Reusable document -level binding¶
If there are many documents in the same SML-IF that share the same/similar list of schema documents, then repeating that for each document will become a performance burden. We can improve efficiency of document-level binding by making it possible to name the bindings so that they can be used by more than one document. ¶
¶
<#>Brute Force schema binding¶
Use "instance-level schema binding" as the only source to locate schema documents for a given namespace. That is, every instance document in the IF document MUST specify namespaces used by that document and schema documents used for that namespace.¶
¶
"Reusable instance-level binding" can also be used with this approach.¶
<#>7. Analysis¶
<#>7.1 Namespace matching
¶
"Namespace matching" is easier to specify, understand, and implement. But it doesn't have proper support for:¶
<#>Chameleon include/redefin

... [2]

2.3.2. Otherwise

2.3.2.1. If it's a schema-complete set, then the <include> or <redefine> is unresolved (which is allowed by XML Schema validity assessment rules)

2.3.2.2. Otherwise, it's implementation-defined whether it tries to resolve <include> or <redefine> to schema documents outside the IF

2.4. The list of <documentAlias> documents are assessed against this *same* schema. targetXXX and identity constraints can now be checked. Similar to <documentAlias> under <ruleBinding> elements, each <documentAlias> can refer to multiple documents via URI prefixing.

3. Compose a schema using *all* schema documents included in the IF, then use this schema to assess those instance documents that are not included in any <schemaBinding>.

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

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

6.4 Proposal Analysis

- Great synergy with <ruleBindings>
  - It works in a way very similar to Schematron rules. You associate a schema (built from a set of schema documents) with a set of instance documents
- Handles all the requirements
  - Supports schema composition: chameleon included documents is supported by removing them from the corresponding <namespaceBinding> (whose “namespace” attribute is absent)
  - Supports schema versioning: multiple versions can be specified in different <schemaBinding> elements
  - Deterministic: the association between instances and schemas is deterministic
  - Full schema support: <include/redefine/import> are all supported
  - Schema document exchange: similar to chameleon included documents, exchange-only documents can also be omitted from the corresponding <namespaceBinding>
- Meets all the constraints
  - Supports access to schema documents outside of SML-IF: when schemaComplete=false, processors are allowed to use external schema documents
  - Ignorable schema locations: all xsi:schemaLocation attributes can be ignored
  - Includes definition and instance documents as-is: no need to modify any included document; document aliases are used.
  - Lazy schema assembly: the schema is known up-front; no need to handle lazy assembly
  - Supports reference constraints: instances specified under the same <schemaBinding> use the same schema, so reference constraints can be checked.
- Simple to understand
- This has may Note that the “trivial case” is also handled by bullet 2. That is, there is no <schemaBinding> and all instance documents are assessed against the same schema.

7 Implementation Cost

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

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

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

8 Acknowledgement

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

There are several ways to specify how to locate schema documents that are used to construct a schema. Each approach needs to answer the following questions:

How to handle <xs:include>

How to handle <xs:redefine>

How to handle <xs:import>

How to locate the correct version of each schema document used to validate a given instance

In XML Schema, schema composition is achieved using one or more of <xs:include>, <xs:redefine>, and <xs:import>. Schema doesn't provide details about how to assemble a schema from unrelated schema documents. To achieve interoperability, the following subsections always describe various approaches as having a synthetic "root" schema document whose transitive closure over include/redefine/import covers all schema documents that are supposed to be used. This is similar to how XSLT 2.0 determines how schemas should be assembled. Implementations are not required to implement schema assembly using synthetic schema documents.

Namespace matching

In the SML-IF instance, there is a namespace to schema document mapping. Such a mapping can be established by, for example, traversing all schema documents and gathering their target namespace attributes.

If for a given namespace, there is more than one schema document, then a choice must be made about which to use. This text assumes that all documents for the namespace are logically merged, as if the consumer synthesized another schema document with this namespace as its target namespace and includes all these documents. This synthetic schema document is used in the above namespace to schema document mapping.

For each model instance document, construct a list of namespace names that are used by this instance (e.g. namespace of elements, attributes, or xsi:type values). Synthesize a schema document that imports all these namespaces, with schema locations pointing to schema documents from the namespace to schema document mapping. The schema corresponding to this synthetic schema document is used to validate the instance.

<xs:include> and <xs:redefine> are handled by matching the schemaLocation value with aliases of schema documents.

<xs:import> is handled by ignoring its schemaLocation value and always using the document from the namespace to schema document mapping.

Selective namespace matching

The above describes the "all" match alternative of the namespace matching approach. There are other possibilities. For example "first" match means that for each namespace, the first document with that namespace as its target namespace is used in the mapping. <xs:include> and/or <xs:redefine> are used if other documents from the same namespace need to be taken into account.