CABINET OFFICE

Office of the *e-Envoy*

Leading the drive to get the UK online

delivering UK online

e-Government Schema Guidelines for XML

3.0

December 2002

Document Control

Abstract
This document contains rules and guidelines for developing XML schemas for e-GIF compliant systems. These guidelines include mandatory requirements for XML Schema structure and content, as well as best practice recommendations for schema design.

Current Version

Date	Version	Status	Editor	Comment
2 December 2002	3.0	Release	Paul Spencer paul.spencer@boynings.co.uk	Review comments added.

Change History

Date	Version	Status	Editor/ Author	Comment
16 August 2002	3.0a	draft	Paul Spencer paul.spencer@boynings.co.uk	Document restyled, Many recommendations reviewed and updated. Appendix on future considerations added.
27 March 2002	2.0	Release	Paul Spencer paul.spencer@boynings.co.uk	Section describing intended audience added. Approved by government Schema group 19 Feb 2002. Next due for review October 2002
25 October 2001	2.0a	Draft	Ann Wrightson	Updated to reflect schema development experience and developments in GSG technical consensus since version 0.2 Does not reflect work still in progress where XML representation is still undefined - High Level Architecture, Government Data Standards Catalogue, e-Government Metadata Standard.
8 March 2001	1.0	Agreed	P Spencer	Approved by Interoperability Group
29 December 2000	0.2	Draft	Paul Spencer	DRAFT WITH SUBSTANTIAL CHANGES IN THE LIGHT OF CONSULTATION AND EXPERIENCE
6 Sept 2000	0.1b	Draft	P Spencer	Second draft for public consultation
28 July 2000	0.1A	Draft	Paul Spencer	First draft

Contents

Introduction

Joined-up government needs joined-up information systems. The e-Government Schema Guidelines for XML help with this by providing a framework for a consistent approach to schema design. This will help understanding of schemas, promote re-use of schema components and aid system interoperability.

The e-Government Schema Guidelines for XML form part of the e-Government Interoperability Framework[1]. Adherence to the mandatory aspects of these guidelines is required for a schema to be registered with UK GovTalk™.

Background

This is the third issue of this document. Since the publication of version 2, the Government Schema Group, and Government organizations generally, have extended their experience of schema development. Other Governments have also started similar initiatives, and their experience has been taken into account in this version of the guidelines. There are therefore a few new guidelines and changes to many of the existing guidelines.

The main changes in this version are:

a change of document title to allow use of schema languages other than W3C XML Schema (see the appendix "Representing Value Sets

There are two issues here. One is how should value sets be referenced and codes used in instance documents dereferenced (possibly differently for different platforms and different languages)?

The other is how should long-term archives reference value sets when it is possible that access to the value set could be required decades or centuries after the document has been produced?

- Locale- and Application-Specific Schema Extensions" on page 29);
- the incorporation of appendices to introduce subjects for further discussion; and
- additional explanations and examples.

Intended Audience

These guidelines are intended for those developing XML schemas for the public sector.

System Context for Schema Development

The guidelines below are in two categories, mandatory *requirements* for schema development within an e-GIF driven development environment and *recommendations* of best practice.

- Some of the mandatory requirements constitute system-wide design decisions regarding integrity relationships between XML schemas. This includes specific support for architectural schemas – schemas containing reusable structures and datatypes providing reusable resources for developing schemas.

Practical Context for Schema Development

Schemas can be generated by hand using suitable tools or generated directly by tools from suitable data models. The majority of schemas will continue to be developed by hand, at least in the short term. However, their development will be supported by an increasing body of information architecture resources: the High Level Information Architecture, the Government Data Standards Catalogue, the e-Government Metadata Standard, and local information models relating to specific applications.

In addition, tool support for XML messaging and XML schema development is improving, and the management of the UK GovTalk™ schema collection is evolving to meet the expected needs of a wide range of users, and increasing numbers of schemas.

Notational Basis of the Schemas

The UK Government e-Government Interoperability Framework (e-GIF[1 & 2]), specifies XML as the primary means for data integration. This is driving the ongoing development of a repertoire of XML schemas. These XML schemas adhere to the XML Schema Recommendation of the World Wide Web Consortium (W3C)[4, 5 & 6].

To avoid excessive amounts of description and explanation, it is assumed that the reader is familiar with, at least, Part 0[4] of the W3C specification, and the nature and use of Namespaces in XML[7].

The W3C Recommendation allows many options for how schemas can be designed. This document provides specific recommendations and guidance for the development of XML Schemas for e-GIF compliant applications and systems.

In particular, the W3C XML Schema Recommendation provides several ways to reuse schema components. These need to be used selectively, and carefully managed, in the context of the e-GIF. In particular, it is essential that schema reuse is easy to understand for application developers who are neither experienced abstract data modellers nor experienced XML designers. Therefore, the requirements and recommendations below emphasize simplicity and ease of use rather than technical elegance.

Requirement Classification

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document apply to the client application and are to be interpreted as described in RFC 2119[8].

Layout and Terminology

These guidelines are broken up into several sections relating to different aspects of schema design:

- "XML Schema Guidelines" provide guidance on data design and the requirements and conventions relating to the schemas as a whole;
- "XML Schema Component Guidelines" provide requirements and conventions relating to modelling individual schema components; and
- "Metadata and Schemas" provides requirements and conventions relating to the use of metadata within schemas.

Appendices are then used to highlight those areas under discussion for future inclusion in this document. Comments are invited on the content of these appendices.

Each guideline is broken up into two or three sections:

- "Guidance" provides a summary of the requirement or recommendation;
- "Explanation" provides background information on why the guidance has been adopted; and
- "Examples" may be present to provide non-normative examples of use of the guideline

The terms **(XML) schema** and **(XML) schema document** are often used interchangeably to refer to XML documents containing schema elements expressed in XML as described in the W3C Recommendation. There is also a more precise technical meaning for **schema**, as the exact abstract data structure required to schema-validate an element of an XML document (this is described in detail in the W3C XML Schema Recommendation Part 1[5]). For the purposes of this document, **schema** is normally used loosely, to mean a schema element within an XML document. The term **schema document** is used to mean an XML document containing one or more schema elements.

An **XML document** is a well-formed and complete piece of XML as defined by the XML Recommendation[9]. Since in the case of interoperability requirements, most documents are being sent as messages between computer systems, these are also referred to here as **XML messages**.

An **architectural schema** is a schema providing resources for reuse in message schemas (for example, the definition of a `NationalInsuranceNumber` simple data type).

In due course, architectural schemas are expected to be developed to support use of the High Level Information Architecture and the Government Data Standards Catalogue in schema development. Government organizations may develop their own architectural schemas.

A **message schema document** is a schema document defining the structure and content of an XML document or message payload. The term "message schema" (rather than "document schema") is used here so as not to cause confusion by having "schema documents" and "document schemas" meaning two different things.

An **instance** is an element within an XML document, which is schema-valid with respect to some message schema. If this document is the document element, the document is often referred to as an **instance document**. Note that there will not be direct instance documents of architectural schemas - they are purely for re-use within message schemas.

XML Schema Guidelines

Primary Schema Language

Guidance

W3C XML Schema MUST be used as the main schema language for describing XML documents.

Explanation

It is important for interoperability that all Government systems use the same schema language. Currently, the only two that meet the e-GIF requirements of standardization and wide market support are the Document Type Definition (DTD) and XML Schema. Of these, XML Schema is preferred because of its support for namespaces, data typing and modular schema design.

See page 28 for a discussion on the use of DTDs.

There will be examples where schemas need to be tailored for different uses. See the appendix "Representing Value Sets

There are two issues here. One is how should value sets be referenced and codes used in instance documents dereferenced (possibly differently for different platforms and different languages)?

The other is how should long-term archives reference value sets when it is possible that access to the value set could be required decades or centuries after the document has been produced?

Locale- and Application-Specific Schema Extensions" on page 29 for further information.

Schema Complexity

Guidance

Remember your audience! The less common facilities available with XML Schema SHOULD NOT be used where there are simpler alternatives. Schema developers SHOULD look at examples of other GovTalk schemas, particularly those developed centrally, to help determine appropriate style. Schema developers SHOULD take into account the testability of their schemas.

Explanation

This is perhaps the most important rule. XML Schema allows enormous power and flexibility in the way schemas are defined. In most cases, schemas can be made simple or complex while achieving the same aim. Since these are new technologies, many people who will be looking at your schemas will have little experience, so try to keep them simple.

Schema development and testing tools have errors, mainly in the less frequently used aspects of XML Schema. Simple schemas are not only intrinsically simpler to test, but are also less likely to cause confusion by exposing the weaknesses of commonly-used tools.

Model Data not Forms

Guidance

XML schemas SHOULD model the underlying data needed for an application, rather than existing forms – the forms are often a good starting point, but SHOULD NOT dominate the eventual message design.

Explanation

There are two reasons for this: first, a well designed form is designed for use on paper, not on a computer screen (a significantly different medium in many ways). Second, a schema design should follow from a information models and activity models of the whole e-service - and that should be designed in its own right, rather than blindly following legacy paper-based processes.

Use of Namespaces and Qualifiers

Guidance

If your schema document has a target namespace, any default namespace for the document MUST be the same as the target namespace.

The W3C XML Schema namespace MUST be qualified with a prefix of either `xsd` or `xs`.

A suitable qualifier MUST be used for other namespaces.

Explanation

There is never a disadvantage of making the default namespace of a schema document the same as the target namespace. However, any other approach can cause problems if another schema document with no target namespace is `<include>`d in the document being developed.

Since this means that neither the XML Schema namespace nor any other can be the default, they require a prefix. Schema development tools invariably default to using to using either `xs` or `xsd` as the prefix for the XML Schema namespace, so these are provided as options.

This makes the usage of namespaces more explicit, and allows schema designers more flexibility in using namespaces within the schema.

Examples

```
<xsd:schema
  targetNamespace="http://www.govtalk.gov.uk/taxation/VAT100"
  xmlns="http://www.govtalk.gov.uk/taxation/VAT100"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.0"
  id="HMCE-VAT100">
```

Guidance

A namespace URI MUST NOT contain version information.

Explanation

There are many ways of applying versions to schemas, one of which is to indicate the version number (or an issue date) in the URI of the target namespace. This method of versioning can cause problems with documents developed according to old versions on the schema, and so is discouraged. An alternative versioning system is indicated in the section "Metadata and Schemas" on page 21.

Use of `elementFormDefault` and `attributeFormDefault`

Guidance

`elementFormDefault` MUST be set to `qualified` and `attributeFormDefault` SHOULD be set to `unqualified`.

The exception to this is if you are defining attributes that will be attached to elements from other namespaces. XLink[8] is a good example of this - the linking information is provided in attributes from the XLink namespace that are attached to elements from the namespace of the source documents.

Explanation

This ensures that a developer reading or reusing a schema can rely on the visible prefixes and namespaces, instead of having to trace the detailed internal structure of a schema.

Examples

In this case, the attributes will not be attached to elements from other namespaces, and so must be qualified:

```
<xsd:schema
  targetNamespace="http://www.govtalk.gov.uk/taxation/VAT100"
  xmlns="http://www.govtalk.gov.uk/taxation/VAT100"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.0"
  id="HMCE-VAT100">
```

In this case, the attributes might be attached to elements from a different namespace:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.govtalk.gov.uk/gms"
  elementFormDefault="qualified"
  attributeFormDefault="qualified"
    ...
```

Messages and Schemas

Guidance

A message schema SHOULD describe a single kind of XML message.

Explanation

The key aim here is to enable reuse of common message parts without having over-complex message schemas.

Although it is tempting to use the flexibility of XML Schema to provide sophisticated schema definitions covering groups of related messages, this temptation should be resisted for the sake of simplicity and ease of use. In particular, avoid designing a schema where making a change affecting just one message in one e-service involves re-issuing a schema document used to validate messages in other e-services.

Where a group of messages uses very similar content, a design choice needs to be made between creating one message schema for the group, and creating a local architectural schema to contain the common parts.

Data Type .v. Element Declarations

Guidance

In many cases, there is a choice of defining a re-usable component as either a data type or as an element. A component MUST be defined as a data type if either:

- it is to be used with different element names in different contexts; or
- it is expected that further data types will be derived from it.

A component MUST be defined as an element if

- there is no intention to derive new components from it; and
- the element is to be used with its name unchanged

Explanation

There are many circumstances in which an element should be used with its name unchanged. For example, if a Unique Tax Reference (UTR) always has the name `UniqueTaxReference`, its semantics will be known and two systems using the same element will be known to be using the same definition. It is therefore possible to build a dictionary of element names with known interoperable semantics.

However, there are other circumstances where it is not appropriate to allocate a name to an element at the time an architectural schema is developed. For example, an address could have several meanings and so be used with different names, such as `CorrespondenceAddress`, `HomeAddress`, `BusinessAddress`, `ElectoralAddress` etc. An address should therefore be defined as a global data type (as it is in the GovTalk Address and Personal Details schema).

The other circumstance for choosing between an element and a data type to define a component is if there is an intention to derive other components from it. By only using data types in this instance, we simplify understanding of schemas by only having a single inheritance mechanism and avoiding use of `xsd:redefine` for this purpose.

In some cases in an architectural schema, it is appropriate to define both a data type and an element. The element is then available with known fixed semantics for re-use and the data type available for appropriate modification.

Examples

The declaration of a component that will always be used with the same name and will not have other components derived from it:

```
<xsd:element name="UniqueTaxReference>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]{1,10}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

The declaration of a component which will be used with different names:

```
<xsd:complexType name="InternationalAddressStructure">
  <xsd:sequence>
    <xsd:element
      name="IntAddressLine"
      type="AddressLineType"
      minOccurs="2"
      maxOccurs="5"/>
    <xsd:choice>
      <xsd:sequence>
        <xsd:element name="Country" type="AddressLineType"/>
        <xsd:element
          name="InternationalPostCode"
          type="InternationalPostCodeType"
          minOccurs="0"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element
          name="InternationalPostCode"
          type="InternationalPostCodeType"/>
        <xsd:element
          name="Country"
          type="AddressLineType"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Global Definitions

Guidance

Schema documents SHOULD only make available globally those component definitions that are either

- re-used within the schema;
- to be made available for re-use in other schemas; or
- are intended to be used as the document element of instance documents.

In general, this means that architectural schemas will use a salami slice and/or Venetian blind style, while message schemas will use a Russian Doll style. (See "Appendix A Styles of Schema Design" on page 25 for a detailed explanation of the named styles.)

Explanation

The main reason for this approach is to limit the effect of change. By keeping component definitions local, it is easy to control who else uses these definitions and so limit the impact of change.

Also, a key attribute of message schemas is that they are easily readable and maintainable. (If a schema is generated from a metadata repository or some other database, the requirements for change. However, the schema still needs to be readable.)

Experience has shown that the Russian Doll model provides good readability since the format of the schema mirrors the format of the instance messages. It also has the benefit that definitions that are only used once are defined locally, so that the same name can be re-used within the schema, leading to simpler element and attribute names.

Common Definitions and Namespaces

Guidance

An architectural schema that contains a collection of schema components (elements and/or datatypes) that will be reused locally within a number of schemas, SHOULD be defined without a target namespace. The resulting architectural schema is then accessed from other architectural schemas or message schemas using the `xsd:include` mechanism. This results in all the definitions of the included schema being in the target namespace of the including schema.

A set of definitions specific to Government MUST be defined within a suitable namespace. The resulting architectural schema is then accessed from other architectural schemas or message schemas using the `xsd:import` mechanism. All references to the architectural schema will then use the namespace for that schema. Architectural schemas which are maintained using a distinct business process SHOULD have their own target namespace.

By *non-specific* we mean terms that are not specific to Government or a single application. Thus the definition of a National Insurance Number is specific, but that of an email address or ISBN is not, and so does not belong in a Government namespace. In many cases, the division is not obvious. For example, Government might use a specific format from a choice of several, in which case that definition (effectively a restriction based on a non-Government definition) belongs in a Government namespace.

Explanation
The use of architectural schemas without a target namespace (chameleon schemas) simplifies the use of namespaces in instance documents. However, when components have defined semantics specific to Government, they should reside in a Government namespace.

This keeps Government namespaces for Government data, without making excessive use of difference namespaces in instance documents.

Example
See the UK GovTalk^{TM} Address and Personal Details schemas for examples of both cases.

Element .v. Attributes

Guidance
Schemas MUST be designed so that elements are the main holders of information content in the XML instances. Attributes are more suited to holding ancillary metadata – simple items providing more information about the element content. Note that, if the element containing the attribute has element content, any attributes SHOULD apply to all the descendant elements.

Attributes SHOULD NOT be used to qualify other attributes.

Explanation
Unlike elements, attributes cannot hold structured data. For this reason, elements are preferred as the principal holders of information content. However, allowing the use of attributes to hold metadata about an element's content (for example, the format of a date, a unit of measure or the identification of a value set) can make an instance document simpler and easier to understand.

If an attribute were allowed to qualify other attributes, there could arise cases with multiple attributes where it would not be clear whether an attribute was qualifying the element or one of several attributes. Therefore it is not allowed for attributes to qualify other attributes.

Examples
A date of birth might be represented in a message as:

```
<DateOfBirth>1975-06-03</DateOfBirth>
```

However, more information might be required, such as how that date of birth has been verified. This could be defined as an attribute, making the element in a message look like:

```
<DateOfBirth
  VerifiedBy="View of Birth Certificate">1975-06-03</DateOfBirth>
```

The following would be inappropriate:

```
<DateOfBirth
  VerifiedBy="View of Birth Certificate"
  ValueSet="ISO 8601"
  Code="2">1975-06-03</DateOfBirth>
```

It is not clear here whether the `Code` is qualifying the `VerifiedBy` or the `ValueSet` attribute. A more appropriate rendition would be:

```
<DateOfBirth>
  <VerifiedBy Code="2">View of Birth Certificate</VerifiedBy>
  <Value ValueSet="ISO 8601">1975-06-03</Value>
</DateOfBirth>
```

We have just seen the incorporation of a value set. This is another example:

```
<Currency ValueSet="ISO 4217">GBP</Currency>
```

This can be extended by providing a link to further information on the value set used for `Currency`. Note that, in this case, the XLink attributes are considered to be qualifying the `Currency` element, not the `ValueSet` attribute.

```
<Currency
  ValueSet="ISO 4217"
  xlink:type="simple"
  xlink:href="
http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=3
4749">GBP</Currency>
```

Indicating Value Sets

Guidance

Value sets SHOULD be indicated by an attribute on the element whose text content holds the value belonging to the value set. Further information MAY be provided using an XLink simple link to the location of the information. The destination of this link SHOULD be to a web site managed by either the UK Government or a recognised specification authority (i.e. a standards body such as ISO or a well-respected organization such as OASIS).

Explanation

Value sets, both internationally accepted (such as the ISO 4217 set for currency codes) and Government defined (such as the DfES codes for ethnicity) are frequently used to aid interoperability. It is important to understanding to indicate the value set in use, and useful to the reader if further information is available and the location of that information is indicated.

Note: The mechanism for dereferencing values in value sets is currently under consideration (see page 29).

Note: Long-term archives may require a different mechanism for indicating the value set in use (see page 29).

Examples

Without a link to further information:

```
<Currency ValueSet="ISO 4217">GBP</Currency>
```

With a link to further information:

```
<Currency
  ValueSet="ISO 4217"
  xlink:type="simple"
  xlink:href="
http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=3
4749">GBP</Currency>
```

Representing Alternative Conditions

Guidance

Alternative conditions SHOULD be represented using attribute values rather than by the presence or the absence of an element.

Explanation

As a matter of XML style, some implementers use the presence or absence of an (empty) optional element to indicate the presence of absence of a condition; some prefer to always have the element present, and use an attribute (typically with values such as "yes" and "no") to indicate the presence of absence of a condition. This makes understanding an instance document (and any code processing it) easier for the human reader.

(Technical note: In XML Schema, there is a facility to specify that an element may be empty using nillable="true". However, this means that the instance document must reference the XML Schema Instance namespace and explicitly set the attribute xsi:null to "true" for the element concerned. This is a good example of a more obscure XML Schema feature which it is better not to use, for the sake of simplicity and readability.)

Commenting Schemas

Guidance

In documenting a W3C XML schema, the `documentation` element MUST be used rather than XML comments.

Explanation

The `documentation` sub-element of the `annotation` element exists to help you document your schemas. The advantage of using this element rather than putting text into XML comments is that, being part of the content of the schema document, the text can be processed easily with a stylesheet, for example to prepare user documentation. Information in comments does not have to be passed from the XML processor to an application such as an XSLT processor, and so can be lost.

However, annotation content does add processing overhead in the XML processor. Because of this, schemas that will be widely used can exist in documented and undocumented form – provided that the two are kept in step automatically.

Examples

An example of good practice is the GovTalk envelope schema (http://www.govtalk.gov.uk/interoperability/agreedschema_schema.asp?schemaid=45), which is rich in annotation elements with documentation sub-elements. For machine use, it is processed using an XSLT stylesheet to produce a schema document with only the top level annotation element remaining. It is processed through other stylesheets to generate human-readable documentation.

Use of Schema Reuse Features

Guidance

Use of `xsd:redefine` SHOULD be avoided.

Explanation

This is to avoid pervasive side-effects in reused components, and to increase clarity and readability.

However, there are occasions where `xsd:redefine` can be used with care. For example, if the only change between a schema for one year and the equivalent schema for the next is a restriction of a code list that is marked up as a simple data type, a solution might be to `redefine` the code list and avoid other schema changes. See the appendix "Representing Periodic (e.g. Annual) Variations to Schemas" on page 29.

Guidance

`xsd:import` MUST NOT be used without a `namespace` attribute.

Explanation

This feature allows unqualified reference to foreign components with no target namespace. This would lead to schemas which are difficult to debug and to update - and for which the reuse dependencies were invisible.

XML Schema Component Guidelines

Naming Conventions

Guidance

The names of complex data types SHOULD end with the text string `Structure`. The name of simple data types SHOULD end with the text string `Type`. Because of this, avoid these endings for element names.

Explanation

This gives consistency of naming, while allowing simple differentiation between simple data type names, complex data type names and element names.

Guidance

Abbreviations SHOULD NOT be used. Extremely long names SHOULD be avoided by designing concise and informative names. Well known abbreviations, including the use of initial letters only, MAY be used. However, a well known abbreviation to one community may be incomprehensible to others who need to use the same message (and who do understand the full name).

Explanation

This is intended to make names comprehensible across Government and so aid understanding of schemas.

Guidance

All names SHOULD use upper camel case. That is, names start with an initial capital, then each new word within the name starts with an initial capital. Where an all uppercase abbreviation (such as UK) or a digit is incorporated into a name, the following word should start with a lower case letter.

Explanation

This is one of many possible naming conventions, but adopting one provides consistency. This helps when referring to names since the capitalization is known and so does not have to be remembered.

Examples

UKaddress, but UnitedKingdomAddress.

Guidance

Enumerated values SHOULD use lower case throughout. Where the value is a proper name or an abbreviation or acronym that normally is used with different capitalization, the usual capitalization should be used.

Explanation

The important thing about enumerated values is consistency in the use of case. We therefore use lower case throughout for these unless they are names that properly start with a capital letter. Thus we use "yes" and "no", but would use "Inland Revenue", for example in an enumerated type containing names of Government departments.

Use of the Government Data Standards Catalogue

Guidance

The Government Data Standards Catalogue (GDSC)[11] MUST be used as a reference document for data type and element definitions unless a domain-specific schema has been agreed for a specific use. Where there are centrally-defined schemas defining these datatypes, these MUST be used. Where these schemas do not exist, schemas SHOULD be designed so that it is easy to replace these interim local schemas with GDSC schemas when they become available. These interim schemas SHOULD be submitted to the Government Schema Group (GSG).

Explanation

This helps interoperability by ensuring that items defined with their semantics in the GDSC are used where possible.

In some instances, domain-specific schemas might be in general use. An example is the use of XBRL for business reporting to Companies House and the Inland Revenue. In these cases, these domain-specific schemas can be used within their domain, but data transferred outside this domain should use the GDSC.

Submitting schemas defining GDSC items to the GSG will help their work and ensure you are contacted when any central definition is being designed.

Use of XML Schema Inheritance

Guidance

If an existing definition does not meet your exact requirements, you MAY use the XML Schema inheritance mechanism to define a new data type based largely on an existing one.

In some cases a data type enumerates all permitted values, or defines a standardized data format such as a postcode whose importance for interoperability goes beyond XML messages. In these cases, inheritance SHOULD only be used to restrict the possible values of the data type, so the values allowed under the new definition are a subset of those allowed in the definition on which it is based. In other words, make sure the modified definition still complies with the underlying data standard.

Explanation

The inheritance mechanism allows the derivation of new types to be made obvious to the user, and allows tools to identify the dependencies between definitions. However, care is required since this introduces a binding between definitions that is not present if new definitions are produced instead.

Data Content of Elements

Guidance

Optional elements which are designed to have content SHOULD NOT be allowed to occur empty. The schema SHOULD ensure that they are either absent or populated.

Explanation

If you have optional elements, lack of data can be signified by omitting the element from an instance document. Some implementers find it easier to provide an empty element than leave the element out – for example, you can use the same code for populated and unpopulated items for legacy system information going into XML, and so make the code simpler. However, this is considered to be less important than keeping the interoperability layer (i.e. the XML documents) clean and concise; empty-but-present optional items occupy system resources, and there are many cases within Government when these resources may be hard pressed.

Guidance
Mandatory elements which are designed to have content SHOULD not be allowed to occur empty. The schema SHOULD ensure that they are populated.

Explanation
If an element is mandatory, there is a good chance that the business rules for the document also require it to contain data. If this is the case, the relevant XML Schema mechanism should reflect this.

For this reason, there is a centrally defined `PopulatedStringType` that enforces at least a single character to be present, and this should be used in preference to `xsd:string`.

Local .v. Global Attribute Definitions

Guidance
In general, attributes SHOULD be given a local scope by defining them within the context of their owning element.

Explanation
This keeps things simple and easy to understand, while avoiding possible namespace issues (since the attribute form should normally be `unqualified`). If an attribute with a similar definition is used in several places, define a data type or attribute group and reuse this. If discussions about data in attributes are suggesting solutions more complex than this, then the data in the attributes should probably be in elements instead.

Text .v. Codes

Guidance
This guidance has been removed while the handling of value sets is decided (see page 29).

Use of Mixed Content Model for Data

Guidance
XML documents can be broadly divided into two types - the text-centric, such as the report of a public enquiry, and the data-centric, such as a tax return.

In a data-centric document, the mixed content model (where an element contains both other elements and character data, SHOULD be avoided.

Explanation
In a data-centric document, it is important to be able to extract data from the document in as simple a way as possible. It is easier to extract an element's character data when this is the only element content. This does not apply to text-centric documents, where the mixed content is an inherent part of the document structure, and will be processed accordingly.

Examples
The following is an example from a text-centric document:

```
<Paragraph>This is an example of a <Emphasize>mixed</Emphasize> content
model in a text-centric document, and is acceptable.</Paragraph>
```

the following is acceptable in a data-centric document:

```
<DateOfBirth
```

```
  VerifiedBy="View of Birth Certificate">1975-06-03</DateOfBirth>
```

While this is not:

```
<DateOfBirth>1975-06-03
  <VerifiedBy>View of Birth Certificate</VerifiedBy>
</DateOfBirth>
```

Metadata and Schemas

Schema documents and e-GMS Metadata

Guidance

Schema documents MUST have e-GMS metadata. The e-GMS Local Metadata Standard for XML schemas describes the metadata required.

Explanation

UK GovTalk™ will implement some form of schema registry and repository. For schemas, and individual declarations within schemas, to be indexed and searched, consistent use of metadata is required. The e-GMS and the e-GMS local standard for schemas were developed to meet that need.

The `version` and `id` Attributes in the `schema` Element

Guidance

In accordance with current W3C practice, schemas MUST indicate a schema version number using the `version` attribute of the `schema` element. Released schemas MUST have a version number in the form n.m (e.g. 1.2), while drafts MUST have a version number of the form n.ma (e.g. 1.2b).

The major version number (n) MUST be changed when the change from the previous version of schema will cause existing documents to fail to validate. This could occur, for example, if a new mandatory element is added.

The minor version number (m) MUST be changed when the change to the schema will result in all existing documents continuing to validate. However, some new documents which validate against the new version will fail against the old version. For example, this could occur with the addition of a new optional element.

The version letter (a) MUST change every time a new draft is issued. In the example above, the version 1.2b is the second draft based on an existing release 1.1, and will lead to a new release 1.2.

The `id` attribute of the `schema` element SHOULD be used to indicate the identity of the schema and MUST be the same as that indicated in the e-GMS `Identifier` element [12].

Explanation

Indicating the version of a schema is good practice and helps prevent problems caused by people accidentally working with incorrect schema versions.

Examples

```
<xsd:schema
  targetNamespace="http://www.govtalk.gov.uk/taxation/VAT100"
  xmlns="http://www.govtalk.gov.uk/taxation/VAT100"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.0"
  id="HMCE-VAT100">
```

Indicating Schema Versions in Data

Guidance

Schemas MUST require the inclusion of a version number in those instance document elements where it is appropriate. This MUST include the intended document element of instance documents and MAY include other elements. This version number MUST use the attribute name `SchemaVersion`.

Explanation

Many XML instances will be persistent documents, outliving the schema version for which they were developed. Indicating the version in the document indicates which version was used as the model for development.

Similarly, although XML messages are not generally persistent, the applications that generate them are, and might use out of date schema versions. By indicating the schema version in the message, a receiving application can decide whether to accept the message as it is, process it in some special way or reject it with a suitable error message.

An instance document might use components declared in several schema documents. In this case, several elements may require version numbers.

Examples

The declaration of a VAT element might be:

```
<xsd:element name="VAT100">
  <xsd:complexType>
    <xsd:sequence>
      <!-- element content goes here -->
    </xsd:sequence>
    <xsd:attribute
      name="SchemaVersion"
      type="xsd:NMTOKEN"
      use="required"
      fixed="1.0"/>
  </xsd:complexType>
</xsd:element>
```

Compliance

Compliance to the mandatory requirements is required for all XML schemas to be approved under UK GovTalk^{TM} . When schemas are reviewed in the course of the UK GovTalk^{TM} process, then both the requirements and the recommendations will be applied in the course of the review. Recommendations are expected to be followed unless there is sufficient reason to do otherwise in a specific case.

Conformance

Conformance is verified by review during the UK GovTalk^{TM} process.

Appendix A Styles of Schema Design

XML Schema allows a variety of styles of definition and declaration, ranging from a very hierarchical style to a very flat style. There is no absolute right or wrong here, but different styles are appropriate for different situations as described in the section "Global Definitions". On the xml-dev mailing list, certain distinctive styles were given descriptive names, which are explained below, and used above in the guidelines.

At one extreme, every element is defined locally to where it is used. This is known as the Russian Doll model. In this style, a simple Name element might be defined as:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="Name">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Forename" maxOccurs="unbounded">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="35"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Surname">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="35"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Here, the element `Name` defines the two element types `Forename` and `Surname` within its own scope. The declarations include their data types as anonymous simple data types.

At the other extreme is the hierarchical approach known as the Salami Slice model. In this approach, each element is defined globally by effectively breaking down the instance document into its components.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="Name">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Forename" maxOccurs="unbounded"/>
        <xsd:element ref="Surname"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Forename">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="35"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="Surname">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="35"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
</xsd:schema>
```

The third schema style is known as the Venetian Blind model. This is similar to the Salami Slice model, but instead of creating global element definitions, we create global data types:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="Name">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Forename" type="ForenameType"
                     maxOccurs="unbounded"/>
        <xsd:element name="Surname" type="SurnameType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="ForenameType">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="35"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="SurnameType">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="35"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

It is possible to go even further on the breakdown, combining the Salami Slice and Venetian Blind Models to create both elements and data types globally:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="Name">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Forename" maxOccurs="unbounded"/>
        <xsd:element ref="Surname"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Forename" type="ForenameType"/>
  <xsd:element name="Surname" type="SurnameType"/>
  <xsd:simpleType name="ForenameType">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="35"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="SurnameType">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="35"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

In general, a schema document will combine elements of each style. Any document that is valid to the Russian Doll or Venetian Blind style schemas will be valid to any of the other styles. However, the Salami Slice and combined styles both allow the `Forename` and `Surname` elements to be the document element of conforming documents. The Russian Doll and Venetian Blind styles only allow `Name` to be the document element of a conforming document.

Appendix B Items for Further Study

The following items are currently under discussion and could affect these guidelines in the future. Comments are invited.

DTDs

Many existing XML document types, especially in the document management area, are described using DTDs. The e-GIF[2] does not include DTDs in its data integration specifications, and so it cannot be included here.

Whilst new DTDs should not be developed for Government use, the use of existing DTDs requires further consideration.

e-GMS Metadata

The desired outcome is that following adoption of a general approach to XML representation of e-GMS metadata in XML (currently a consultation draft v1.0, in restricted circulation), the Local Metadata Standard for XML schemas (currently in v1.0) will be updated, and thenceforth used for all GovTalk schemas. Existing schemas contain various prototype metadata representations, and will need to be updated once the specification is stable.

Schema metadata is not expected to change significantly with the issue of e-GMS v2.0.

The effect of the full adoption of e-GMS metadata on XML schemas will require further consideration following release.

Use of `default` and `fixed`

There are circumstances where it would be useful to either fix or default the value of an element or attribute. For example, this allows the schema to effectively declare its version in instance documents (see "Indicating Schema Versions in Data" on page 22).

However, this means that the fixed value may only be available if the document is validated against the schema. Documents that rely on the XML processor inserting the value from the schema will not make this information available if a schema language other than XML Schema is used.

My preference is to forbid anything that relies on the schema to insert document content.

Representing Value Sets

There are two issues here. One is how should value sets be referenced and codes used in instance documents dereferenced (possibly differently for different platforms and different languages)?

The other is how should long-term archives reference value sets when it is possible that access to the value set could be required decades or centuries after the document has been produced?

Locale- and Application-Specific Schema Extensions

When a schema is developed for general use, it does not constrain instance documents as much as one developed for a specific use. In some cases, there is a need to apply additional constraints whilst ensuring that the instance documents are valid to the original schemas.

The document "Report on Alternative methods of EML Localisation"[16] discusses this in the specific case of the Election Markup Language, and could form the basis for discussion of guidelines for more general cases.

Representing Periodic (e.g. Annual) Variations to Schemas

Many Government applications change their data models periodically, typically once a year. There are various alternatives to how these variations should be captured at the schema level.

For example, the simplest is to define a new schema based on the old one. The disadvantage is that the changes are not obvious (although tools such as DeltaXML[17] might help document the changes).

An alternative is to use the `redefine` element of XML Schema. This allows the changed components to be modified in the schema without affecting others. However, this is limited in its use and is unlikely to be suitable in all cases.

We invite other suggestions from those who have practical experience of solutions to this problem.

Alternative Case Conventions

Since these guidelines first recommended using upper camel case for both elements and attributes, ebXML[18] has adopted an alternative convention, which has also been adopted in the USA by the Federal XML Developer's Guide[19].

This convention is to use upper camel case for element names and lower camel case for attribute names. Is there any reason to adopt this convention? Should it remain under consideration for the future? Or should be stick with what we have in perpetuity?

Usage of Acronyms and Abbreviations

The current UK Government usage is defined in the section "Naming Conventions" on page 17. ebXML[18] proposes stricter guidelines (acronyms SHOULD NOT be used and abbreviations MUST NOT be used in element and attribute names). In the USA, these have been adopted by the Federal XML Developer's Guide[19].

Should we follow suit?

Use of ISO 11179 for Component Names

We currently have no standard for choosing names. In the USA, the Federal XML Developer's Guide[19] specifies usage of ISO 11179 Part 5[20] as a naming convention, with rules for mapping these names into XML element and attribute names.

Should we do likewise?

Potential Impact of an XML Schema Registry / Repository

There are no specific discussion points at this stage. However, at some point these guidelines will need to be revised to cater for a UK Government schema registry/repository.

Abbreviations

Abb	Definition
GDSC	Government Data Standards Catalogue
EDI	Electronic Data Interchange
e-GIF	e-Government Interoperability Framework
e-GMS	e-Government Metadata Standard
FTP	File Transfer Protocol (an Internet protocol for managing and transferring files)
HTTP	Hypertext Transfer Protocol
OeE	Office of the e-Envoy
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XPath	XML Path Language
XSLT	Extensible Stylesheet Language Transformations

References

1. The e-Government Interoperability Framework, version 4 Part 1: Framework, currently at http://www.govtalk.gov.uk/interoperability/egif_document.asp?docnum=534

2. The e-Government Interoperability Framework, version 4 Part 2: Technical Policies and Specifications, currently at http://www.govtalk.gov.uk/interoperability/egif_document.asp?docnum=536

3. The e-Government Metadata Framework, currently at http://www.govtalk.gov.uk/interoperability/metadata_document.asp?docnum=219

4. XML Schema Part 0: Primer, W3C Recommendation 2001 http://www.w3.org/TR/xmlschema-0/

5. XML Schema Part 1: Structures, W3C Recommendation 2001 http://www.w3.org/TR/xmlschema-1/

6. XML Schema Part 2: Datatypes, W3C Recommendation 2001 http://www.w3.org/TR/xmlschema-2/

7. Namespaces in XML http://www.w3.org/TR/REC-xml-names/

8. XML Linking Language (XLink) Version 1.0 http://www.w3.org/TR/xlink/

9. Extensible Markup Language (XML) 1.0 (Second Edition) http://www.w3.org/TR/REC-xml

10. Key words for use in RFCs to Indicate Requirement Levels http://www.ietf.org/rfc/rfc2119.txt

11. Government Data Standards Catalogue, in three parts, currently available using the document list at http://www.govtalk.gov.uk/interoperability/egif.asp

12. e-GMS Local Metadata Standard: XML Schemas

13. The Schematron - An XML Structure Validation Language using Patterns in Trees http://www.ascc.net/xml/resource/schematron/schematron.html

14. XML Path Language (XPath) Version 1.0
http://www.w3.org/TR/xpath/

15. XSL Transformations (XSLT) Version 1.0
http://www.w3.org/TR/xslt/

16. Report on Alternative methods of EML Localisation

17. DeltaXML
http://www.deltaxml.com/

18. ebXML Technical Architecture Specification v1.0.4
http://www.ebxml.org/specs/ebTA.pdf

19. Draft Federal XML Developer's Guide
http://xml.gov/documents/in_progress/developersguide.pdf

20. ISO 11179: Specification and Standardization of Data Elements
http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSN
UMBER=1758