

Guía de Desarrollo y Uso de Esquemas de Gobierno

Ministerio de Economía Fomento y Reconstrucción

Santiago, Junio de 2009

Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Organización Responsable:

Departamento de Ciencias de la Computación (DCC), FCFM, Universidad de Chile.

Av. Blanco Encalada 2120, 3er Piso, C.P. 837-0459, Santiago, Chile.

Teléfono: +56 2 9784365.

www.dcc.uchile.cl

Miembros del Equipo de Trabajo:

Renzo Angles, DCC, Universidad de Chile.

Alex Bórquez, DCC, Universidad de Chile.

Claudio Gutiérrez, DCC, Universidad de Chile (+).

Andrés Neyem, DCC, Universidad de Chile.

Sergio F. Ochoa, DCC, Universidad de Chile (+).

Andrés Pereira, DCC, Universidad de Chile.

Edgard Pineda, DCC, Universidad de Chile.

(+) Líderes del Proyecto



La estrategia digital
del Gobierno de Chile

Lista de Acrónimos

AEM	Administrador de Esquemas y Metadatos
EIF	European Interoperability Framework (Marco de Referencia para la Interoperabilidad en Europa)
GIF	Government Interoperability Framework (Marco de Referencia para la Interoperabilidad en el Gobierno)
SAGA	Standards and Architectures for e-Government Applications (Estándares y Arquitecturas para Aplicaciones de Gobierno Electrónico)
URI	Uniform Resource Identifier (Identificador Uniforme de Recurso)
URL	Uniform Resource Locator (Localizador Uniforme de Recurso)
XML	Extensible Markup Language (Lenguaje de Etiquetado Extensible)
W3C	World Wide Web Consortium (Consortio de la World Wide Web)

Índice de Contenidos

1	Introducción	8
2	Terminología	9
3	Estructura de esta guía	10
4	Recomendaciones generales para diseñar esquemas	11
4.1	Modelar datos y no formatos de documentos	11
4.2	Comprensibilidad de los esquemas XML	11
4.3	Simplicidad en el diseño de los esquemas XML.....	12
4.4	Complejidad semántica de los esquemas XML	12
4.5	Restricciones de los esquemas XML	13
4.6	Reusabilidad y redundancia de los esquemas XML	13
4.7	Extensibilidad de los esquemas XML	14
5	Guía sobre diseño de esquemas XML	14
5.1	Definiciones generales	14
5.1.1	Idioma oficial.....	14
5.1.2	Codificación de caracteres	15
5.1.3	Nombramiento de elementos y atributos	15
5.1.4	Nombramiento de tipos.....	17
5.2	Espacio de nombres (namespaces).....	17
5.2.1	Definición de namespaces	17
5.2.2	Namespace de XML Schema	18
5.2.3	Uso del default namespace y el target namespace	18
5.2.4	Uso de atributos elementFormDefault y attributeFormDefault	19

5.2.5	Definición de componentes genéricos.....	20
5.2.6	Definición de componentes particulares	21
5.3	Estructura del esquema	23
5.3.1	Tipos de datos y declaración de elementos.....	23
5.3.2	Definición de atributos.....	24
5.3.3	Definiciones globales	24
5.3.4	Atributos globales versus atributos locales	25
5.3.5	Uso de los atributos <code>default</code> y <code>fixed</code>	26
5.3.6	Definición de componentes obligatorios.....	27
5.3.7	Definición de elementos opcionales.....	27
5.3.8	Representación de componentes alternativos.....	28
5.3.9	Texto y códigos	29
5.3.10	Uso de elementos con contenido mixto.....	30
5.3.11	Redefinición de componentes de esquemas XML.....	30
5.3.12	Importación de documentos de esquemas XML	31
5.3.13	Uso de derivación de tipos.....	32
5.3.14	Referencias absolutas y relativas.....	33
5.4	Metadatos y documentación de esquemas.....	34
5.4.1	Metadatos estándar para documentos de esquemas XML.....	34
5.4.2	Versionamiento de documentos de esquemas XML	35
5.4.3	Indicación de la versión de los documentos de esquemas XML en los documentos instancia XML.....	36
5.4.4	Uso del atributo ID en el elemento <code>xsd:schema</code>	37
5.4.5	Uso de referencia para namespaces.....	37
5.4.6	Asignación de nombres a archivos de documentos de esquemas XML	39

5.4.7	Incorporación de comentarios en documentos de esquemas XML	40
6	Bibliografía	41

1 Introducción

En el año 2004, a partir de la aprobación de la Norma técnica para los órganos de la administración del Estado de Chile sobre interoperabilidad de documentos electrónicos, surgen una serie de desafíos con el fin de cumplir las características mínimas obligatorias por las cuales deben regirse los documentos electrónicos, desde su generación hasta su almacenamiento en todo ámbito. A través del Decreto Supremo (D.S.) 81/2004 (y su modificación D.S. 158/2006) se establece que *el documento electrónico deberá ser codificado en formato XML v1.0 y utilizar XML Schema para definir los esquemas de los distintos tipos de documentos. Cada esquema definido debe ser público, de libre disponibilidad y persistente.*

El diseño de esquemas XML es una tarea que exige precisión, debido al impacto que los esquemas resultantes tendrán sobre los diseños futuros. Debido a que el lenguaje XML Schema es una red de construcciones que se superponen mutuamente, crear varios esquemas entrelazados que pueden ser extendidos y versionados, aumenta el nivel de precisión requerido.

Por otra parte, el artículo 29 del D.S. 81 establece una serie de disposiciones que asegurarán a las autoridades superiores de los órganos de la Administración, el cumplimiento de ciertas prácticas que se regirán por esta norma. Estas disposiciones son:

- *Que todo esquema definido será público, de libre disponibilidad y persistente, salvo que el contenido del documento tenga el carácter de confidencial o secreto, situación en la cual a éste se le podrá dar el mismo tratamiento.*
- *Que se cree un registro de esquemas XML que potencie la reutilización de esquemas que sean de aplicación transversal. Para los documentos existentes se crearán los esquemas respectivos, por ejemplo memo, circular, etc.*
- *Se establezcan reglas que regulen las directivas para la creación de esquemas, estableciendo lineamientos y buenas prácticas para el desarrollo de esquemas XML.*
- *Existan metadatos y definición de diccionarios semánticos.*
- *Se creen diccionarios de definiciones y sus requerimientos.*

En vista de lo anterior, esta *guía de buenas prácticas* provee un marco de referencia consistente para el desarrollo y uso de esquemas de documentos electrónicos en el gobierno. Esta guía ayudará a diseñar esquemas XML, y a implementar interoperabilidad de datos entre sistemas.

2 Terminología

Las palabras clave **DEBE** (MUST), **NO DEBE** (MUST NOT), **OBLIGATORIO** (REQUIRED), **DEBERÁ** (SHALL), **NO DEBERÁ** (SHALL NOT), **DEBERÍA** (SHOULD), **NO DEBERÍA** (SHOULD NOT), **RECOMENDADO** (RECOMMENDED), **PUEDE** (MAY) y **OPCIONAL** (OPTIONAL) en este documento serán interpretadas como está descrito en el RFC 2119-es [1].

A continuación se presenta el significado de algunos términos importantes usados en este documento:

- **Esquema.** Un esquema define la estructura abstracta de un dato.
- **Esquema XML.** Es la definición del esquema de un componente, expresada en el lenguaje XML Schema.
- **Esquema XML Arquitectural.** Es un esquema XML diseñado para ser reutilizado por otros esquemas XML.
- **Documento de Esquemas XML.** Es un documento que contiene uno o más esquemas XML. La extensión del archivo que contiene el documento de esquemas XML es *xsd*.
- **Documento XML.** Es un documento expresado en el lenguaje XML.
- **Documento Instancia XML.** Es un documento XML que respeta las reglas de estructuras y tipos definidas por un documento de esquemas XML. En este caso se dice que el documento XML es válido.
- **Documento de Esquemas XML Arquitectural.** Es un documento que contiene uno o más esquemas XML arquitecturales. Un documento de esquemas XML arquitectural no tiene documentos instancia XML asociados.
- **Componente.** Es un elemento, atributo o tipo de dato definido en un esquema.

- **Componente XML.** Es un elemento, atributo o tipo de dato definido en un esquema XML.
- **Instancia XML.** Es un elemento o atributo dentro de un documento instancia XML. Una instancia XML es válida, si respeta las reglas de estructuras y tipos definidas por un esquema XML.
- **Documento Electrónico.** Es un documento instancia XML perteneciente a una organización. Nótese que se habla de un documento instancia XML, ya que todo documento electrónico debe ser validado por un documento de esquemas XML.

3 Estructura de esta guía

Esta *Guía de Desarrollo y Uso de Esquemas de Gobierno* consta de las siguientes secciones:

- *Recomendaciones Generales para Diseñar Esquemas.* Esta sección provee un panorama general de los requisitos comunes que son importantes para los esquemas de documentos.
- *Guía sobre Diseño de Esquemas XML.* Esta sección provee una lista de *buenas prácticas* para diseñar, escribir y codificar esquemas XML.

Cada directriz está dividida en dos o tres secciones, conteniendo los siguientes ítems:

- *Recomendación*, que provee un resumen de la recomendación.
- *Explicación*, la cual provee información de orientación a cerca de por qué la recomendación debe ser adoptada.
- *Ejemplo*, que usualmente está presente para proporcionar uno o más ejemplos generales de uso de la recomendación.

4 Recomendaciones generales para diseñar esquemas

4.1 Modelar datos y no formatos de documentos

Recomendación

Los esquemas XML DEBERÍAN modelar la estructura de los datos relevantes de un documento, en forma independiente de sus posibles representaciones.

Explicación

Los esquemas XML deberían reflejar la estructura abstracta de los datos, en forma independiente de sus posibles representaciones (formularios, reportes, etc.). Un esquema XML es la representación específica de la estructura de los datos con la cual el sistema opera. La tarea de modelar los datos implica identificar los tipos de datos, su estructura y sus relaciones.

Ejemplo

Un documento XML que modele una boleta de compra-venta no tiene para qué modelar el formato, disposición espacial, y líneas del documento de papel. Sólo debería modelar los datos, por ejemplo: RUT emisor, Fecha, Ítems, Montos, Total, etc.

4.2 Comprensibilidad de los esquemas XML

Recomendación

Los esquemas XML DEBERÍAN ser diseñados en forma clara, coherente e inequívoca.

Explicación

Los esquemas XML deberían ser diseñados de forma tal que faciliten su identificación y comprensión. Estos esquemas deberían contener documentación entendible para el usuario, y donde sea apropiado, deberían contener enlaces a documentos de requerimientos o de diseño.

Ejemplos

Ver sección [5.4](#).

4.3 Simplicidad en el diseño de los esquemas XML

Recomendación

Los esquemas XML DEBERÍAN ser diseñados de manera simple, es decir evitando el uso de funcionalidades poco comunes y esquemas de gran tamaño, y usando las representaciones más intuitivas posibles para los elementos y atributos.

Explicación

El lenguaje XML Schema permite variedad y flexibilidad en la definición de esquemas para documentos XML. En la mayoría de los casos, la estructura de los datos puede ser modelada por diversos esquemas XML. Debido a que todo esquema XML será de dominio público, varios usuarios con diferentes experiencias podrían utilizar el esquema XML propuesto. Por consiguiente, dicho esquema debería ser definido de la manera más simple posible, con la finalidad de potenciar su reutilización por parte de diferentes usuarios.

4.4 Completitud semántica de los esquemas XML

Recomendación

Los esquemas XML DEBEN definir el significado de cada elemento y atributo que será usado en los documentos electrónicos de una organización.

Explicación

Los esquemas XML deben definir el significado de cada uno de los elementos y atributos que serán usados en la validación de los documentos electrónicos. Por ejemplo, si un documento electrónico

utiliza un atributo o elemento, entonces el significado de ese ítem debe estar presente en el esquema XML asociado.

Ejemplos

Ver secciones [5.3.1](#) hasta [5.3.8](#).

4.5 Restricciones de los esquemas XML

Recomendación

Los esquemas XML DEBERÍAN restringir los valores de los elementos y atributos que serán usados en los documentos electrónicos.

Explicación

Cuando se diseña un esquema XML se deberían delimitar los valores de los elementos y atributos, a un conjunto de valores válidos. Esto debe hacerse con la finalidad de que, tanto los elementos como los atributos, puedan ser manejados adecuadamente por una aplicación computacional.

Los esquemas XML establecen un contrato que permite, al creador y al receptor de un documento electrónico, verificar que la instancia del documento electrónico cumple con el contrato.

Ejemplo

Ver sección [5.3.6](#)

4.6 Reusabilidad y redundancia de los esquemas XML

Recomendación

Los documentos de esquemas XML DEBERÍAN ser diseñados de manera tal que sus esquemas XML puedan ser reutilizados por otros documentos de esquemas XML.

Explicación

Los documentos de esquemas XML deberían ser diseñados de tal forma que potencien la reutilización y eliminen la redundancia de esquemas XML. Por ejemplo, un documento de esquemas XML podría importar o incluir otros documentos de esquemas XML, con la finalidad de reutilizar los esquemas allí contenidos.

Ejemplos

Ver secciones [5.3.12](#) hasta [5.3.14](#).

4.7 Extensibilidad de los esquemas XML

Recomendación

Los esquemas XML DEBERÍAN ser diseñados para ser extensibles en el tiempo.

Explicación

Los esquemas XML deberían ser fácilmente extensibles en el tiempo, a fin de permitir la incorporación de nuevos componentes XML. Los puntos de extensión pueden ser explícitos, si el esquema XML está diseñado para evolucionar.

Ejemplos

Ver secciones [5.3.12](#) hasta [5.3.14](#).

5 Guía sobre diseño de esquemas XML

5.1 Definiciones generales

5.1.1 Idioma oficial

Recomendación

Los esquemas XML DEBEN ser diseñados con nombres de elementos, atributos y tipos de datos que deben estar compuestos de palabras en el idioma Español.

Explicación

Los nombres de los elementos, atributos y tipos de datos usados en los esquemas XML, deben utilizar palabras del idioma Español, a fin de facilitar su identificación y comprensión por parte del usuario.

Los caracteres permitidos son sólo aquellos en los rangos a-z y A-Z. También podrán usarse las vocales acentuadas, la letra ' n ' con virgulilla o tilde (ñ y Ñ), y la vocal ' u ' con diéresis (ü y Ü). Sin embargo, se recomienda no utilizar estos últimos, dentro de lo posible.

5.1.2 Codificación de caracteres

Recomendación

La codificación de caracteres a usar en los esquemas XML DEBERÍA ser UTF-8.

Explicación

El artículo 9 del Decreto Supremo 81 establece que *para la codificación de caracteres se utilizará preferentemente UTF-8*. En caso de utilizar otra codificación, se recomienda el uso de un servicio de conversión a UNICODE.

5.1.3 Nombramiento de elementos y atributos

Recomendación

Los nombres de los elementos en los esquemas XML DEBERÍAN ser escritos de acuerdo a la convención *Upper Camel Case*. Los nombres de los atributos DEBERÍAN ser escritos siguiendo la convención *Lower Camel Case*.

Explicación

Camel Case es un estilo de escritura que se aplica a frases o palabras compuestas. Esta convención es utilizada por varios lenguajes de programación y guías de especificaciones en la industria y el gobierno.

Los nombres de los elementos en los esquemas XML deberían ser escritos de acuerdo a la convención Upper Camel Case, esto es, palabras compuestas sin espacios y poniendo en mayúscula la primera letra de cada palabra (por ejemplo: UpperCamelCase).

En cuanto a los nombres de los atributos, estos deberían ser escritos siguiendo la convención Lower Camel Case, esto es, palabras compuestas sin espacios, donde la primera letra de la primera palabra debe ir con minúscula, y poniendo en mayúscula la primera letra a partir de la segunda palabra (por ejemplo: lowerCamelCase).

Tanto los elementos como los atributos deberían ser nombrados en forma singular, a no ser que el concepto sea en sí mismo plural.

Símbolos tales como: punto, guiones (alto y bajo) u otra clase de separadores (por ejemplo, espacios) no deben usarse en los elementos y atributos. Esto mismo vale para aquellos caracteres no permitidos en la especificación XML del consorcio W3C.

Los acrónimos, abreviaturas u otro tipo de truncado de palabras no deberían ser usados, a menos que estos sean parte de un registro central. En ese caso, si se utilizan al inicio de la declaración de un atributo, entonces deben aparecer en minúsculas; en el resto de los casos siempre deben ser escritos en mayúsculas. Aquellas abreviaturas o acrónimos que se usen en elementos o tipos de datos, deben permanecer en mayúsculas.

Ejemplo

El siguiente ejemplo muestra en la línea 2 la utilización de esta recomendación.

```
1 <?xml version="1.0" encoding="UTF-8" ?>  
2 <NombreElemento nombreAtributo="ValorAtributo"/>
```

5.1.4 Nombramiento de tipos

Recomendación

Los nombres de los tipos de datos (tanto simples como complejos) en los esquemas XML, DEBERÍAN ser escritos de acuerdo a la convención Upper Camel Case, anexando al final del nombre la palabra "Type".

Explicación

Los nombres de los tipos de datos en los esquemas XML deben ser escritos de acuerdo a la convención Upper Camel Case, esto es, palabras compuestas sin espacios y poniendo en mayúscula la primera letra de cada palabra. Además, estos nombres deben terminar con la cadena de texto "Type". Esta regla provee consistencia en el nombramiento, debido a que permite una forma sencilla de diferenciación entre nombres de tipo de datos (simples o complejos) y nombres de elementos.

Ejemplo

El siguiente ejemplo muestra en la línea 5 la utilización de esta recomendación.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsd:schema
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4   elementFormDefault="qualified">
5   <xsd:simpleType name="CodigoPostalType">
6     <xsd:restriction base="xs:long">
7       <xsd:maxInclusive value="0"/>
8       <xsd:maxExclusive value="9999999"/>
9     </xsd:restriction>
10  </xsd:simpleType>
11 </xsd:schema>
```

5.2 Espacio de nombres (namespaces)

5.2.1 Definición de namespaces

Recomendación

Es RECOMENDADO que cada organización defina su propio espacio de nombres, según la especificación de namespace de la W3C.

Explicación

El uso de múltiples vocabularios trae problemas de reconocimiento y colisión de nombres de elementos y atributos. Por lo tanto, el tener diferentes espacios de nombres permite desarrollar en forma modular y distribuida los esquemas. Esto da flexibilidad a la definición de esquemas.

5.2.2 Namespace de XML Schema

Recomendación

El namespace de XML Schema de la W3C DEBE ser calificado con un prefijo de `xsd` o `xs`.

Explicación

Todo esquema XML debe usar el prefijo `xsd` o `xs` para identificar al namespace de XML Schema.

Ejemplo

El siguiente ejemplo muestra en la línea 4 la utilización de esta recomendación.

```
1 <xsd:schema
2   targetNamespace="http://www.aem.gob.cl"
3   xmlns="http://www.aem.gob.cl"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   elementFormDefault="qualified"
6   attributeFormDefault="unqualified">
7   ...
```

5.2.3 Uso del default namespace y el target namespace

Recomendación

Si un documento de esquemas XML define un target namespace, el default namespace para el documento DEBE tener el mismo valor que el target namespace.

Explicación

No existen desventajas al definir el default namespace de un documento de esquemas XML igual al target namespace. Sin embargo, cualquier otra combinación en la definición de namespaces puede

causar problemas, si otro documento de esquemas XML que no define un target namespace es *incluido* (usando `<include>`) en otro documento de esquemas XML.

En consecuencia el namespace de XML Schema, así como cualquier otro namespace distinto del target namespace, requerirá de un prefijo. Esto define un uso explícito de namespaces, permitiendo a los diseñadores de esquemas XML una mayor flexibilidad al usar namespaces dentro del esquema XML.

Ejemplo

El siguiente ejemplo muestra, en las líneas 2 y 3, la utilización de esta recomendación.

```
1 <xsd:schema
2   targetNamespace="http://www.aem.gob.cl"
3   xmlns="http://www.aem.gob.cl"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   elementFormDefault="qualified"
6   attributeFormDefault="unqualified">
7   ...
```

5.2.4 Uso de atributos `elementFormDefault` y `attributeFormDefault`

Recomendación

El atributo `elementFormDefault` DEBE estar definido como `qualified` y el atributo `attributeFormDefault` DEBE estar definido como `unqualified`.

Explicación

Esto asegura que un diseñador que desea leer o reutilizar un documento de esquemas XML, pueda confiar en los namespaces y en los prefijos visibles, en lugar de tener que hacer un seguimiento de la estructura interna del documento de esquemas XML.

Ejemplo

El siguiente ejemplo muestra en las líneas 5 y 6 la utilización de esta recomendación.

```
1 <xsd:schema
2   targetNamespace="http://www.aem.gob.cl"
3   xmlns="http://www.aem.gob.cl"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   elementFormDefault="qualified"
6   attributeFormDefault="unqualified">
7   ...
```

5.2.5 Definición de componentes genéricos

Recomendación

Los componentes genéricos PODRIAN no tener asignado un namespace.

Explicación

Un componente genérico podría ir definido en un documento de esquemas XML que no defina un target namespace (diseño Camaleón). Se entiende con *componente genérico* a un término que no es específico a una sola aplicación.

En este caso, el documento de esquemas XML arquitecturales resultante será accedido desde otro documento de esquemas XML, usando el elemento `xsd:include`. En consecuencia, todas las definiciones del documento de esquemas XML incluido, pasan a formar parte del namespace del documento de esquemas XML que los incluye (en caso que este último defina alguno).

La opción de no asignar un namespace a los esquemas XML arquitecturales simplifica el uso de namespaces en documentos instancia XML. Sin embargo, cuando un componente tiene un significado específico dentro de una aplicación, dicho componente debería residir en el namespace de dicha aplicación.

Ejemplos

El siguiente ejemplo muestra el contenido del esquema XML arquitectural almacenado en el documento de esquemas XML nombrado como `complementos.xsd`.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsd:schema
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4   elementFormDefault="qualified">
5   <xsd:simpleType name="stringValidador">
6     <xsd:restriction base="xsd:string">
7       <xsd:pattern value="[\sa-zA-Zá-úÁ-Úà-ùÀ-Ûä-Û-Ü]+"/>
8     <xsd:whiteSpace value="preserve" />
9   </xsd:restriction>
10 </xsd:simpleType>
11 </xsd:schema>
```

El siguiente ejemplo muestra el contenido del documento de esquemas XML nombrado como nombre .xsd. La línea 5 muestra la utilización de esta recomendación.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsd:schema
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4   elementFormDefault="qualified">
5   <xsd:include schemaLocation="complementos.xsd" />
6   <xsd:complexType name="nameType">
7     <xsd:sequence>
8       <xsd:element name="nombres" type="stringValidador" />
9       <xsd:element name="apellidoPaterno" type="stringValidador" />
10      <xsd:element name="apellidoMaterno" type="stringValidador" />
11    </xsd:sequence>
12  </xsd:complexType>
13 </xsd:schema>
```

5.2.6 Definición de componentes particulares

Recomendación

Los componentes particulares de una organización DEBEN tener asignado un namespace definido por dicha organización.

Explicación

Los componentes específicos de una organización deben estar definidos por esquemas XML con un namespace apropiado. El documento de esquemas XML que define componentes particulares, será accedido desde otros documentos de esquemas XML de la organización (es decir, documentos de esquemas XML con el mismo namespace) usando el elemento `xsd:include`. Si el documento de esquemas XML va a ser accedido desde documentos de esquemas XML de otras organizaciones (es decir, documentos de esquemas XML con distinto namespace), entonces se debe usar el elemento `xsd:import`. Todas las referencias al esquema XML particular deben usar el namespace asignado.

Ejemplos

El siguiente ejemplo muestra el contenido del esquema XML almacenado en el documento de esquemas XML nombrado como `cuerpoExpediente.xsd`.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsd:schema
3   targetNamespace="http://www.aem.gob.cl"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   xmlns:aem="http://www.aem.gob.cl"
6   elementFormDefault="qualified">
7   <xsd:complexType name="cuerpoExpediente">
8     <xsd:sequence>
9       <xsd:element name="documento"/>
10      <xsd:element name="idDocumento" type="xsd:string"/>
11      <xsd:element name="versionDocumento" type="xsd:string"/>
12    </xsd:sequence>
13  </xsd:complexType>
14 </xsd:schema>
```

El siguiente ejemplo muestra el contenido del documento de esquemas XML nombrado como `expediente.xsd`. Las líneas 8 hasta la 11 muestran la utilización de la recomendación `xsd:import`. Las líneas 12 y 13 muestran la utilización de la recomendación `xsd:include`.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsd:schema
3   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   xmlns:aem="http://www.aem.gob.cl"
6   targetNamespace="http://www.aem.gob.cl"
7   elementFormDefault="qualified">
8   <xsd:import
9     namespace="http://www.w3.org/2000/09/xmldsig#"
10    schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-
11    core-schema.xsd"/>
12   <xsd:include schemaLocation="encabezadoExpediente.xsd" />
13   <xsd:include schemaLocation="cuerpoExpediente.xsd" />
14   <xsd:element name="expediente">
15     <xsd:complexType>
16       <xsd:sequence>
17         <xsd:element name="encabezado" type="aem:encabezadoExpediente"/>
18         <xsd:element name="cuerpo" type="aem:cuerpoExpediente"
19           minOccurs="1" maxOccurs="unbounded"/>
20         <xsd:element ref="ds:Signature" minOccurs="0"/>
21       </xsd:sequence>
22     </xsd:complexType>
23   </xsd:element>
24 </xsd:schema>
```

5.3 Estructura del esquema

5.3.1 Tipos de datos y declaración de elementos

Recomendación

Un componente XML DEBE ser definido como un tipo de dato si:

- el componente XML tendrá diferentes nombres en diferentes contextos; o
- el componente XML podría ser usado para definir otros componentes XML (derivación de tipos de datos).

Un componente DEBERIA ser definido como un elemento si:

- el nombre del componente XML será siempre el mismo; o
- el componente XML no será usado para definir nuevos componentes XML.

Es RECOMENDABLE elaborar un diccionario de datos con nombres de componentes que tienen una semántica general conocida (por ejemplo RUN).

Explicación

Al diseñar esquemas XML existen componentes que no cambian su nombre. Por ejemplo, el Rol Único Nacional (RUN) es un número que identifica a cada persona en Chile y siempre tiene el nombre de "RUN". Su significado es conocido por toda organización en Chile de manera que puede ser usado con el mismo nombre en todo documento electrónico. Se recomienda elaborar un diccionario de datos con los nombres de este tipo de componentes.

Existen otras circunstancias en las cuales es más apropiado definir un componente como un elemento. Por ejemplo, una dirección puede tener distintos significados, y en consecuencia distintos nombres dependiendo del contexto o la organización. Ese es el caso de `DirecciónParticular`, `DirecciónDeTrabajo`, `DirecciónDeEnvío`, etc. En este ejemplo, el componente `dirección` debería ser definido como tipo de dato global. Otro motivo para definir un componente como un tipo de

dato, radica en que éste podría ser usado para definir nuevos componentes (por ejemplo mediante derivación).

En el caso de los esquemas XML arquitecturales, existen situaciones en las cuales sería apropiado definir un componente como un elemento y a la vez como un tipo de dato. De esta forma el elemento está disponible para ser usado con una semántica conocida establecida, y el tipo de dato estaría disponible para ser heredado y modificado apropiadamente.

5.3.2 Definición de atributos

Recomendación

- Los atributos DEBEN ser usados sólo para agregar meta-datos que ayuden a clarificar el contenido de un elemento.
- Los atributos DEBERÍAN ser usados para contener unidades de información que no serán extendidas o divididas.
- Un atributo NO DEBE ser usado para calificar otros atributos.
- El valor de un atributo DEBE ser corto.

Explicación

Los esquemas XML deben ser diseñados de manera que los elementos sean los principales contenedores de información en los documentos instancia XML. Los atributos son más apropiados para entregar información adicional a cerca del contenido de un elemento.

Un atributo no debe ser usado para calificar otros atributos, ya que esto puede producir ambigüedad. Si el elemento que contiene al atributo tiene hijos, entonces el atributo deberá ser aplicable a todos los hijos.

5.3.3 Definiciones globales

Recomendación

Un componente XML DEBERÍA ser definido globalmente en un documento de esquemas XML si el componente XML:

- será reutilizado dentro del documento de esquemas XML;
- será reutilizado en otros documentos de esquemas XML (al incluir / importar el documento de esquemas XML que contiene la definición del componente); o
- será usado como un elemento raíz en un documento instancia XML.

Explicación

La principal razón para elegir entre una definición global o local es poder controlar los efectos del cambio. Cuando un componente XML es definido localmente, el uso de su definición es controlada. En cambio, cuando un componente es definido globalmente, se pierde el control sobre su uso ya que queda disponible para ser reutilizado o redefinido.

5.3.4 Atributos globales versus atributos locales

Recomendación

Los atributos DEBERÍAN ser definidos localmente.

Explicación

En general, los atributos deberían tener un alcance local y en consecuencia estar definidos dentro del contexto de su propio elemento. Esto permite un modelamiento simple y fácil de comprender.

Si un atributo será usado en distintos lugares con la misma definición, entonces se deberá definir como un tipo de dato para poder ser reutilizado.

Ejemplo

En el siguiente esquema XML, el atributo “fecha” es definido localmente para el elemento “Documento”.

```
1 <xsd:element name="Documento">
2   <xsd:complexType>
3     <xsd:sequence>
4       ...
5     </xsd:sequence>
6   </xsd:complexType>
7   <xsd:attribute name="fecha" type="xsd:date" use="required"/>
8 </xsd:element>
```

5.3.5 Uso de los atributos `default` y `fixed`

Recomendación

El atributo `default` NO DEBERÍA ser usado para agregar información relevante a elementos o atributos en los documentos instancia XML.

El atributo `fixed` NO DEBERÍA ser usado para agregar información relevante a elementos o atributos en los documentos instancia XML, con la posible excepción de atributos obligatorios.

Explicación

Los atributos `default` y `fixed` permiten que un procesador automático de documentos de esquemas XML añada datos predefinidos a un documento instancia, en base a las definiciones del documento de esquemas XML que lo valida. En muchos casos esto puede ser de utilidad, como por ejemplo para agregar automáticamente la versión de un documento instancia. Sin embargo, el uso de valores fijos o predefinidos tiene la desventaja que:

- El procesador de documentos de esquemas XML podría no ser capaz de agregar, en los documentos instancia XML, valores predefinidos en el documento de esquemas XML asociado.
- El documento de esquemas XML contendrá datos que no serán visibles para el usuario.
- Para que un documento instancia XML tenga la información completa, deberá ir acompañado del documento de esquemas XML.

En resumen, los atributos `default` y `fixed` son útiles, pero deben ser usados cuidadosamente.

5.3.6 Definición de componentes obligatorios

Recomendación

Los componentes obligatorios DEBERÍAN aparecer en todo documento instancia XML y no deben permitir contenido vacío.

Explicación

El esquema XML que tiene un componente obligatorio debería asegurar la ocurrencia del componente en todo documento instancia XML.

Ejemplo

El siguiente esquema XML muestra la definición de componentes XML obligatorios. El elemento “Nombre” se define como obligatorio usando el atributo `minOccurs = “1”`. El atributo “id” se define como obligatorio usando el atributo `use = “required”`.

```
1 <xsd:complexType name="PersonaType">
2   <xsd:sequence>
3     <xsd:element name="Nombre" type="xsd:string" minOccurs="1" maxOccurs="1"/>
4     ...
5   </xsd:sequence>
6   <xsd:attribute name="id" use="required"/>
7 </xsd:complexType>
```

5.3.7 Definición de elementos opcionales

Recomendación

Los componentes XML definidos como opcionales NO DEBERÍAN permitir contenido vacío.

Explicación

Si un esquema XML define un elemento como opcional, su definición debería asegurar que: (a) si el elemento tiene contenido, deberá ocurrir en el documento instancia XML; (b) si el elemento no presenta contenido, no deberá ocurrir en el documento instancia XML.

Los elementos sin contenido ocupan recursos del sistema y pueden generar inconsistencias con respecto a su significado. La recomendación de este punto evita la representación de elementos opcionales a través de elementos sin contenido.

Ejemplo

En el siguiente esquema XML, el elemento “Apellidos” está definido como opcional a través del atributo `minOccurs="0"`. Adicionalmente, la posibilidad de un contenido vacío para el elemento es evitada usando una restricción `xsd:minLength` en el tipo del elemento (línea 12), la cual exige que en caso de aparecer el elemento, su contenido debe ser una cadena (`xsd:string`) de longitud “1” como mínimo.

```
1 <xsd:element name="Persona">
2   <xsd:complexType>
3     <xsd:sequence>
4       <xsd:element name="Nombres" type="xsd:string" minOccurs="1" maxOccurs="1"/>
5       <xsd:element name="Apellidos" type="NoVacioType" minOccurs="0" maxOccurs="1"
6         />
7     </xsd:sequence>
8   </xsd:complexType>
9 </xsd:element>
10 <xsd:simpleType name="NoVacioType">
11   <xsd:restriction base="xsd:string">
12     <xsd:minLength value="1"/>
13   </xsd:restriction>
14 </xsd:simpleType>
```

5.3.8 Representación de componentes alternativos

Recomendación

Los componentes alternativos DEBEN representarse a través de valores, en lugar de usar la presencia o ausencia de los componentes.

Explicación

La ausencia de un elemento o atributo que forma parte de un conjunto de alternativas debe indicarse a través de un valor especial en su contenido. Por ejemplo, se podría usar el valor de “0” cuando el contenido del elemento debiera ser un número entero positivo. Otra opción es el uso de un atributo

especial (por ejemplo, un atributo “presente” con valores “si” o “no”). Esta indicación explícita permite una mejor comprensión de los documentos instancia XML.

Ejemplo

El siguiente esquema XML muestra el elemento “DocumentosAdjuntados”, cuyo contenido corresponde a los elementos alternativos “CopiaRut”, “CertificadoAntecedentes” y “CertificadoDeTitulo”. La ocurrencia de cada uno de estos elementos se indica asignando el valor de “si” o “no” al atributo “presente”.

```
1 <xsd:element name="DocumentosAdjuntados">
2   <xsd:complexType>
3     <xsd:sequence>
4       <xsd:element name="CopiaRUT" type="AlternativaType" />
5       <xsd:element name="CertificadoAntecedentes" type="AlternativaType" />
6       <xsd:element name="CertificadoDeTitulo" type="AlternativaType" />
7     </xsd:sequence>
8   </xsd:complexType>
9 </xsd:element>
10 <xsd:complexType name="AlternativaType">
11   <xsd:attribute name="presente" type="xsd:string" use="required"/>
12 </xsd:complexType>
13 <xsd:simpleType name="SiNoType">
14   <xsd:restriction base="xsd:string">
15     <xsd:enumeration value="si"/>
16     <xsd:enumeration value="no"/>
17   </xsd:restriction>
18 </xsd:simpleType>
```

5.3.9 Texto y códigos

Recomendación

Un código NO DEBERÍA ser usado en lugar de texto sin alguna indicación de su significado.

Explicación

En caso de usar códigos, estos deberían incluir información que permita comprender su significado. Por ejemplo, se puede incluir una referencia a un documento que entregue información a cerca de un código usado. La información adicional debería ser útil tanto para personas como para aplicaciones. Esta recomendación no sólo ayuda a evitar errores de interpretación, también facilita el procesamiento automático de los datos (como por ejemplo, su visualización).

5.3.10 Uso de elementos con contenido mixto

Recomendación

Si un componente es definido orientado a modelar sus datos, su esquema XML NO DEBERÍA permitir un contenido mixto.

Explicación

Al declarar un elemento con el atributo `mixed="true"`, se está definiendo un elemento cuyo contenido es mixto, es decir texto y sub-elementos (por ejemplo, código XHTML). Si el diseño del documento está orientado a modelar sus datos, es importante poder procesar y extraer estos datos de una manera simple. En ese caso, no se deberían usar componentes XML de contenido mixto. En lugar de eso se deberían usar elementos o atributos que modelen los datos de una manera estructurada.

Ejemplos

En siguiente documento instancia XML muestra un diseño orientado a modelar el texto de una solicitud.

```
1 <Solicitud>
2   Señores Registro Civil
3   <cuero>El motivo de la presente ... Atentamente.</cuero>
4   Luis Silva
5 </Solicitud>
```

En siguiente documento instancia XML muestra un diseño orientado a modelar los datos de la solicitud del ejemplo anterior.

```
1 <Solicitud>
2   <Destinatario>Registro Civil</Destinatario>
3   <Cuerpo>El motivo de la presente ... Atentamente.</Cuerpo>
4   <Remite>Luis Silva</Remite>
5 </Solicitud>
```

5.3.11 Redefinición de componentes de esquemas XML

Recomendación

La redefinición de componentes de esquemas XML (a través del elemento `xsd:redefine`) DEBERÍA evitarse.

Explicación

Esta recomendación está orientada tanto a evitar efectos no deseados debido a la reutilización de los componentes, como a incrementar la claridad y legibilidad de los documentos de esquemas XML.

5.3.12 Importación de documentos de esquemas XML

Recomendación

El elemento `xsd:import` NO DEBERÁ ser usado sin el atributo `namespace`.

Explicación

Cuando el elemento `xsd:import` especifica el atributo `namespace`, éste debe coincidir con el target namespace definido en el documento de esquemas XML importado; los componentes XML importados mantienen su namespace. Cuando no se especifica el atributo `namespace` (sólo se especifica el atributo `schemaLocation`), el documento de esquemas XML a importar no debe definir un target namespace; los nuevos componentes XML no tendrán namespace.

Esta recomendación está orientada a evitar componentes XML externos que no estén vinculados a un namespace, ya que esto podría llevar al uso de documentos de esquemas XML con referencias no calificadas difíciles de depurar y actualizar.

Ejemplos

El elemento `xsd:import` deberá contener los atributos `schemaLocation` y `namespace` como se muestra en la siguiente declaración.

```
<xsd:import schemaLocation="http://www.aem.gob.cl/basales/complementos.xsd"  
namespace="http://www.aem.gob.cl" />
```

Un uso no recomendado del elemento `xsd:import` es mostrado en la siguiente declaración.

```
<xsd:import schemaLocation="http://www.aem.gob.cl/basales/complementos.xsd" />
```

Nótese que el elemento `xsd:import` no incluye el atributo `namespace`, lo cual no es recomendado por las razones mencionadas anteriormente.

5.3.13 Uso de derivación de tipos

Recomendación

La herencia de tipos DEBERÍA mantener el estándar de datos subyacente.

Explicación

La herencia de tipos (a través de los elementos `xsd:extension` o `xsd:restriction`) debería usarse cuidadosamente, para mantener la consistencia entre las definiciones originales y aquellas creadas producto de la herencia. Por ejemplo, si el tipo base define un conjunto estándar de valores permitidos, entonces un tipo derivado debería permitir un subconjunto de los valores definidos por el tipo base.

Ejemplos

Considere el siguiente esquema XML el cual define un tipo simple `STprioridad` cuyo contenido está restringido a tres valores posibles: “alta”, “media” y “baja”.

```
<xsd:simpleType name="STprioridad">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="alta"/>
    <xsd:enumeration value="media"/>
    <xsd:enumeration value="baja"/>
  </xsd:restriction>
</xsd:simpleType>
```

Considere el siguiente esquema XML el cual reutiliza el tipo `STprioridad` definido anteriormente.

```
<xsd:simpleType name="STimportancia">
  <xsd:restriction base="STprioridad">
    <xsd:enumeration value="alta"/>
    <xsd:enumeration value="baja"/>
    <xsd:enumeration value="muy-baja"/>
  </xsd:restriction>
</xsd:simpleType>
```

Nótese el uso incorrecto del concepto de restricción, ya que el tipo derivado `STimportancia` está agregando un nuevo valor (`muy-baja`), en lugar de restringir el conjunto de valores definidos por el tipo base. El siguiente esquema XML muestra un uso adecuado del concepto de restricción.

```
<xsd:simpleType name="STimportancia">  
  <xsd:restriction base="STprioridad">  
    <xsd:enumeration value="alta"/>  
    <xsd:enumeration value="baja"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

En este esquema, el conjunto de valores iniciales (“alta”, “media” y “baja”) ha sido restringido a un subconjunto de ellos (“alta” y “baja”).

5.3.14 Referencias absolutas y relativas

Recomendación

Si dos o más documentos de esquemas XML están estrechamente relacionados (respecto a la URI donde están declarados), las declaraciones `xsd:include` y `xsd:import` DEBERÍAN usar referencias relativas. Cuando los documentos de esquemas XML no tienen una relación cercana, se DEBERÍAN usar referencias absolutas.

Sólo DEBEN ser referenciados esquemas que están en el Administrador de Esquemas y Metadatos (AEM) en estado de “Uso Preliminar” o “Uso Consolidado”.

Explicación

Si un conjunto de documentos de esquemas XML usa referencias relativas, estos pueden moverse de un lugar a otro y mantener sus referencias. Si un documento de esquemas XML no está directamente relacionado al grupo, entonces es más apropiado usar su referencia absoluta (por ejemplo, si el documento de esquemas XML será usado por distintas aplicaciones).

Al momento que un esquema pase a estado de “Uso Preliminar”, todos los esquemas a los que referencia deben ser esquemas que se encuentren ya incorporados en el AEM en estado de “Uso Preliminar” o “Uso Consolidado” (según lo establece el D.S. 271/2008), de modo que no ingresen esquemas XML con declaraciones `xsd:include` y `xsd:import` que referencien a esquemas desconocidos por el AEM.

Ejemplos

Considere un documento de esquemas XML llamado "complementos.xsd", el cual contiene definiciones generales de una aplicación bajo el namespace "http://www.aem.gob.cl". Si un documento de esquemas XML, dentro de la misma aplicación, desea hacer uso de las definiciones en "complementos.xsd", este último deberá ser incluido usando el elemento `xsd:include`, junto a una referencia relativa de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.aem.gob.cl"
            targetNamespace="http://www.aem.gob.cl">
  <xsd:include schemaLocation="complementos.xsd"/>
  ...
</xsd:schema>
```

Por otra parte, si un documento de esquemas XML, perteneciente a otra aplicación, desea hacer uso de las definiciones en "complementos.xsd", este último deberá ser importado usando el elemento `xsd:import`, junto a una referencia absoluta la siguiente manera.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:aem="http://www.aem.gob.cl"
            targetNamespace="http://www.otroOrganismo.gob.cl">
  <xsd:import schemaLocation="http://www.aem.gob.cl/basales/complementos.xsd"
            namespace="http://www.aem.gob.cl"/>
  ...
</xsd:schema>
```

5.4 Metadatos y documentación de esquemas

5.4.1 Metadatos estándar para documentos de esquemas XML

Recomendación

Un documento de esquemas XML DEBERÍA incluir metadatos estándar Dublin Core.

Explicación

Los metadatos para un documento de esquemas XML deberían estar basados en el vocabulario definido por Dublin Core. Dublin Core es un modelo de metadatos elaborado y auspiciado por la DCMI (Dublin

Core Metadata Initiative), una organización dedicada a fomentar la adopción extensa de los estándares interoperables de metadatos. Dublin Core se define en la norma ISO 15836 del año 2003.

Ejemplo

El siguiente ejemplo muestra el uso de metadatos Dublin Core para brindar información adicional sobre el documento de esquemas XML. Las líneas 9 hasta la 20 muestran la utilización de la recomendación.

```
1 <?xml version="1.0" ?>
2 <xsd:schema ...
3   <xsd:annotation>
4     <xsd:documentation xml:lang = "es">
5       ...
6     </xsd:documentation>
7
8   <xsd:appinfo>
9     <Metadata>
10      <Identificador>ESXML4536-2</Identificador>
11      <Titulo>Esquema para boleta</Titulo>
12      <Autor>Juan Rojas</Autor>
13      <Descripción>Este esquema define ...</Descripción>
14      <Creación>2008-05-24</Creación>
15      <Modificado>2008-06-04</Modificado>
16      <Modificado>2008-06-30</Modificado>
17      <Estado>Aprobado</Estado>
18      <Versión>3.0</Versión>
19      ...
20    </Metadata>
21  </xsd:appinfo>
22 </xsd:/annotation>
23 ...
24 <xsd:schema>
```

5.4.2 Versionamiento de documentos de esquemas XML

Recomendación

Los documentos de esquemas XML DEBEN indicar su versión utilizando el atributo `versión` del elemento `xsd:schema`.

Explicación

Indicar la versión de un documento de esquemas XML es una buena práctica y ayuda a prevenir el uso de versiones incorrectas.

Ejemplo

El siguiente ejemplo muestra en la línea 4 la utilización de esta recomendación.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3           elementFormDefault="unqualified"
4           version="1.0" >
5   ...
6 </xsd:schema>
```

5.4.3 Indicación de la versión de los documentos de esquemas XML en los documentos instancia XML

Recomendación

Los esquemas XML DEBEN requerir que su versión se indique en sus instancias. Esta versión DEBERIA ser indicada por medio de uno de los siguientes métodos:

- Usando namespace versionado, definido como el namespace en el esquema.
- Obligando al elemento (usualmente el elemento raíz) a incluir la versión usando el atributo SchemaVersion. Este atributo PUEDE usar un valor fixed junto al atributo required.

Explicación

Varios documentos instancia XML pudieran estar almacenados sin tener la versión del documento de esquemas XML que los valide. Indicar la versión del documento de esquemas XML usado en un documento instancia XML permite identificar qué versión se está utilizando dentro del conjunto de documentos de esquemas XML.

Ejemplos

El siguiente ejemplo muestra un documento de esquemas XML donde se aplica el primer método. En la línea 7 se declara la versión.

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

El siguiente ejemplo muestra una declaración del elemento VAT, usando el segundo método. Las líneas 6 hasta la 10 muestran la utilización de la recomendación.

```
1 <xsd:element name="VAT100">
2   <xsd:complexType>
3     <xsd:sequence>
4       <!-- Contenido del Elemento -->
5     </xsd:sequence>
6     <xsd:attribute
7       name="schemaVersion"
8       type="xsd:NMTOKEN"
9       use="required"
10      fixed="1.0"/>
11   </xsd:complexType>
12 </xsd:element>
```

5.4.4 Uso del atributo ID en el elemento `xsd: schema`

Recomendación

Los documentos de esquemas XML DEBERIAN ser identificados con el atributo `id`. El `id` DEBERÍA ser único en el `targetNamespace`.

Explicación

Es una buena práctica proporcionar un identificador de un documento de esquemas XML. Para ello, el atributo `id` es más genérico y por lo tanto debería utilizarse con tal fin. Por ser el atributo `id` un identificador, éste debe ser único; en caso contrario pierde gran parte de su potencial utilidad.

Ejemplo

El siguiente ejemplo muestra en la línea 8 la utilización de la recomendación.

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

5.4.5 Uso de referencia para namespaces

Recomendación

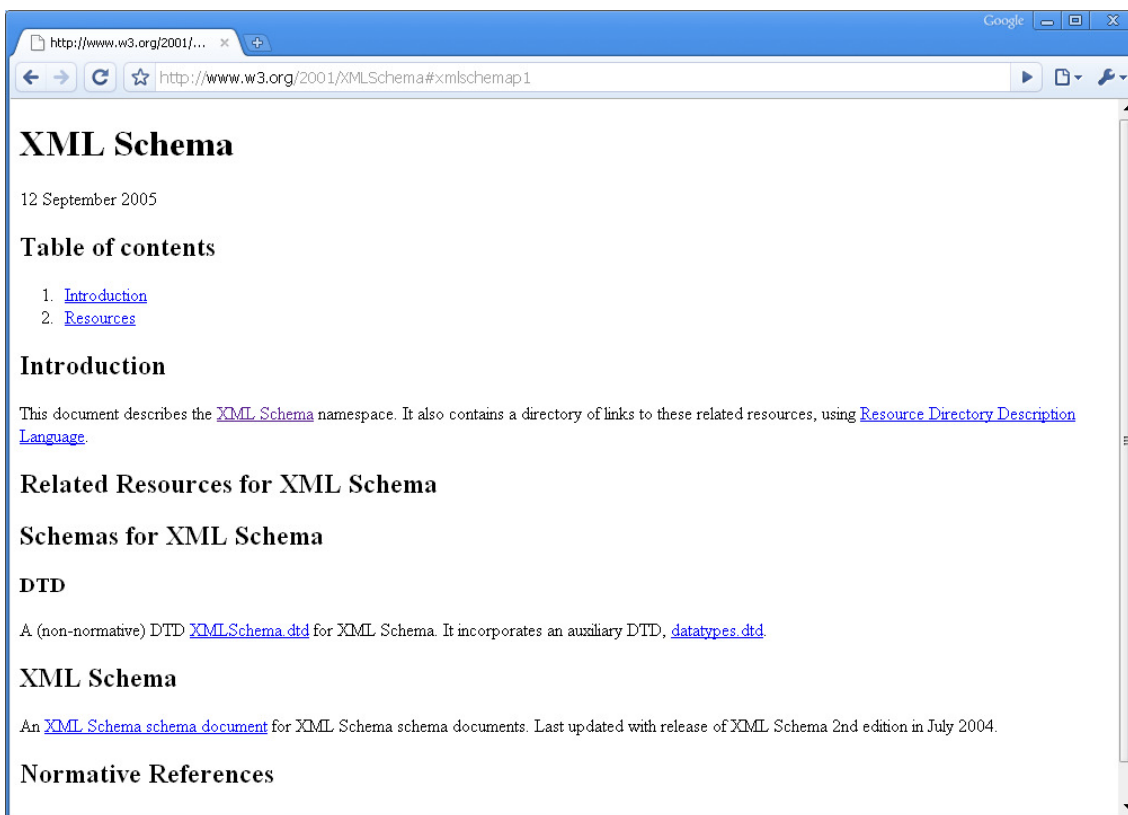
La información sobre un namespace DEBERIA ser accesible públicamente en un lugar que tenga permanencia en el tiempo. Si la URI (Uniform Resource Identifier) del namespace es una URL (Uniform Resource Locator), entonces la documentación DEBERIA estar en la misma localización (mismo URL).

Explicación

Es útil a menudo poder hacer referencia a la información sobre un namespace. Dicha información podría incluir una descripción de la finalidad del namespace y referencias a recursos relacionados, tales como especificaciones y esquemas. El consorcio de la W3C hace esto para sus propios namespaces. Ejemplos de ello se pueden encontrar accediendo al namespace de la W3C a través de un navegador.

Ejemplo

El siguiente ejemplo muestra la información sobre el namespace de XML Schema, al colocar su URI en un navegador.



5.4.6 Asignación de nombres a archivos de documentos de esquemas XML

Recomendación

Los documentos de esquemas XML DEBEN ser guardados en un archivo cuyo nombre identifique su versión.

Explicación

Los documentos de esquemas XML deben indicar en su nombre de archivo, la versión del documento en forma completa. Para ello, se debe seguir el siguiente formato para nombrar a dichos archivos:

NombreDelArchivo-vm-n.xsd.

Primero se debe colocar el nombre que identifica al documento de esquemas XML y añadir al final un guión "- ". Segundo, se debe añadir el carácter "v", el número mayor de versión y añadir al final un guión "- ". Finalmente, se debe incorporar el número menor de versión y la extensión del archivo .xsd.

Si el documento de esquemas XML va a ser cambiado a lo largo del tiempo, el nombre del archivo debe incorporar los cuatro dígitos del año. Para ello, se debe seguir el siguiente formato:

NombreDelArchivo-Año-vm-n.xsd

Ejemplos

El siguiente ejemplo muestra la utilización de la recomendación sin el uso del año:

```
EsquemaRegistroCivil-v1-2.xsd
```

El siguiente ejemplo muestra la utilización de la recomendación con el uso del año:

```
EsquemaRegistroCivil-2008-v1-2.xsd
```

5.4.7 Incorporación de comentarios en documentos de esquemas XML

Recomendación

Los documentos de esquemas XML DEBEN documentarse utilizando el elemento `documentation`.

Explicación

El elemento `documentation` es un sub-elemento de elemento `annotation`, y existe para ayudar a documentar los esquemas XML. La ventaja de utilizar este elemento, en lugar de poner texto en comentarios XML, es que el contenido puede ser procesado de manera más fácil con una hoja de estilo (stylesheet). Por ejemplo, para preparar la documentación de usuario. La información en comentarios XML no es tenida en cuenta por un procesador de XML, por ejemplo XSLT.

Ejemplo

El siguiente ejemplo muestra en las líneas 4 hasta la 7 la utilización de la recomendación.

```
1 <?xml version="1.0" ?>
2 <xsd:schema ...
3   <xsd:annotation>
4     <xsd:documentation xml:lang = "es">
5       <!-- Documentación del Documento de Esquemas XML -->
6       ...
7     </xsd:documentation>
8
9   <xsd:appinfo>
```

```
10     <Metadata>
11         <!-- Metadatos del Documento de Esquemas XML -->
12         ...
13     </Metadata>
14 </xsd:appinfo>
15 <xsd:/annotation>
16     ...
17 <xsd:schema>
```

6 Bibliografía

- [1] Bradner, S. Key words for use in rfc's to indicate requirement levels, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>. Traducción al castellano: <http://www.rfc-es.org/rfc/rfc2119-es.txt>
- [2] Brun, M.H., Nielsen, B. Naming and Design Rules for E-Government - The Danish Approach. March 2003. <http://xml.coverpages.org/NDR-DanishXML2003.pdf>.
- [3] Cabinet Office e-Government Unit. United Kingdom. e-Government Interoperability Framework version 6.1, March 2005. [http://www.govtalk.gov.uk/documents/eGIF%20v6_1\(1\).pdf](http://www.govtalk.gov.uk/documents/eGIF%20v6_1(1).pdf)
- [4] Costello, R. Xml schemas: Best practices, June 2001. <http://www.xfront.com/BestPracticesHomepage.html>
- [5] Cover, R. (OASIS). Extensible markup language (xml) version 1.1 published as a w3c recommendation, February 2004. <http://xml.coverpages.org/ni2004-02-05-a.html>.
- [6] Department of the Navy, United States of America. Xml naming and design rules, January 2005. <http://xml.gov/documents/completed/don/ndr2.pdf>.
- [7] Dublin Core Metadata Initiative (DCMI), 2009. <http://dublincore.org/>
- [8] Environmental Protection Agency (EPA), United States of America. Xml design rules and conventions for the environmental information exchange network, September 2003. <http://xml.gov/documents/completed/epa/EPAGuideSec1.pdf>, <http://xml.gov/documents/completed/epa/EPAGuideSec2.pdf>.
- [9] Federal Ministry of the Interior, Germany. Standards and architectures for e-government applications version 2.0, December 2003. http://www.bsi.bund.de/english/topics/egov/download/5_SAGA2_en.pdf
- [10] Government of the Hong Kong Special Administrative Region. Hksarg interoperability framework, November 2005. <http://www.ogcio.gov.hk/eng/infra/eif.htm>.
- [11] Hors, A.L. (IBM). Xml 1.1 and namespaces 1.1 revealed, May 2004. <http://www-128.ibm.com/developerworks/xml/library/x-xmlns11.html>.
- [12] Office of the e-Envoy. United Kingdom. e-Government metadata standard, April 2004. http://www.govtalk.gov.uk/schemasstandards/metadata_document.asp?docnum=872.
- [13] Office of the e-Envoy. United Kingdom. e-Government schema guidelines for xml, June 2004. http://www.govtalk.gov.uk/schemasstandards/developerguide_document.asp?docnum=946
- [14] Office of the Government, Chief Information Officer. Hong Kong. Xml schema design and management guide, November 2004. <http://www.ogcio.gov.hk/eng/infra/download/g55-2.pdf>.
- [15] Stephenson, D. XML Schema best practices, December 2004. <http://xml.coverpages.org/HP-StephensonSchemaBestPractices.pdf>

- [16] The Unicode Consortium. Unicode 2.0 standard, July 1996.
http://www.unicode.org/versions/enumeratedversions.html#Unicode_2_0_0.
- [17] U.S. Federal CIO Council XML Working Group. Draft federal xml developers guide, April 2002.
http://www.xml.gov/documents/in_progress/developersguide.pdf.
- [18] W3C XML Working Group (WG). Extensible markup language (xml) 1.0 (third edition), February 2004. <http://www.w3.org/TR/2004/REC-xml-20040204/>.
- [19] W3C XML Working Group (WG). Extensible markup language (xml) 1.1 (third edition), February 2004. <http://www.w3.org/TR/2004/REC-xml11-20040204/>.
- [20] W3C XML Working Group (WG). Xml schema part 1: Structures second edition, October 2004.
<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.
- [21] W3C XML Working Group (WG). Xml schema part 2: Datatypes second edition, October 2004.
<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.
- [22] Woodley, M. Glosario dcmi (dublin core metadata initiative), September 2003.
http://www.sedic.es/glosario_DCMI.pdf.