Ministerio de Tecnologías
de la Información y las Comunicaciones
República de Colombia

Gobierno en línea
·Colombia·

Lenguaje para intercambio
de información
·Intranet Gubernamental·

LENGUAJE COMÚN DE INTERCAMBIO DE INFORMACIÓN
GUÍA DE CREACIÓN DE ESQUEMAS XML

PLATAFORMA DE INTEROPERABILIDAD, PDI
INTRANET GUBERNAMENTAL

Bogotá, D.C., Junio de 2010

FORMATO PRELIMINAR AL DOCUMENTO

Título:	**LENGUAJE COMÚN DE INTERCAMBIO DE INFORMACIÓN GUÍA DE CREACIÓN DE ESQUEMAS XML.**				
Fecha elaboración aaaa-mm-dd:	2006-09-15				
Sumario:	Presentar un conjunto de guías básicas para el desarrollo de esquemas XML dentro de la especificación del lenguaje común de intercambio de información.				
Palabras Claves:	Esquemas, *Schema*				
Formato:	DOC	Lenguaje:	Español		
Fecha de publicación aaaa-mm-dd:	2009-03-11	Fecha de modificación aaaa-mm-dd:	2010-06-30		
Dependencia:	Ministerio de tecnologías de la Información y las Comunicaciones: Programa "Agenda de Conectividad" – Proyecto Intranet Gubernamental.				
Código:		Versión:	4.1	Estado:	Publicado
Categoría:	Estándares				
Autor (es):	Equipo GEL-XML: • Departamento Administrativo Nacional de Estadística, DANE. • Departamento Nacional de Planeación, DNP. • Ministerio de tecnologías de la Información y las Comunicaciones: Programa "Agenda de Conectividad". • Ministerio de Hacienda y Crédito Público: Proyecto de Interoperabilidad SIIF Nación. • Ministerio de la Protección Social, MPS: Programa de Apoyo a la Reforma en Salud, PARS. • Informática Siglo 21 • ISL S.A. • Heinsohn Business Technology				
Revisó:	Equipo GEL-XML				
Aprobó:	Equipo GEL-XML				
Información Adicional:					
Ubicación:	El archivo magnético asociado al documento está localizado en http://lenguaje.intranet.gov.co				

CONTROL DE CAMBIOS

VERSIÓN	FECHA	RESPONSABLE	NATURALEZA
1.0	2005-07-05	Equipo HP-MS	Versión Inicial del documento
1.0a	2006-06-27	Eliécer Vanegas Murcia/ Grupo de trabajo GEL-XML	Ajustes respecto a: Cambios en el formato de presentación del documento
2.0	2008-06-17	Equipo GEL-XML	Complemento de la Guía de Creación de esquemas: - Ajustes ortográficos. - Adición de ejemplos. - Ajustes sobre versionamiento. - Inclusión de ejemplos. - Inclusión regla de usos. - Ajustes de redacción. - Definición de metadatos a incluir en los esquemas. - Ajustes de formato del documento.
3.0	2008-07-21	Equipo GEL-XML	- Ajustes sobre etiquetas *ref* e *import y* trabajo futuro.
3.1	2008-07-23	Equipo GEL XML	- Inclusión ejemplo uso de etiqueta *import*
3.2	2008-09-26	Equipo GEL-XML	- Ajustes generales de ortografía - Inclusión para la especificación de adaptadores.
3.3	2008-11-10	Equipo GEL-XML	- Revisión sección control de cambios - Ajustes en la sección de adaptadores - Revisión sintaxis ejemplos XML
3.4	2008-12-03	César Ariza	- Adición de ejemplos en adaptadores - Correcciones Generales.
3.5	2009-01-30	César Ariza	- Ajustes sobre adaptadores - Recomendaciones sobre el inclusión (import) para evitar referencias circulares - Ajustes explicación al patrón Venetian Blind. - Correcciones ortográficas.
3.6	2009-02-23	César Ariza	- Ajustes en los adaptadores - Adición de guías para la inclusión de elementos de datos en bibliotecas. - Adición de guías para la inclusión de elementos de datos de estándares externos.
3.7	2009-03-04	César Ariza	- Ajuste en ejemplos de inclusión de elementos de dato en bibliotecas.
3.8	2009-03-10	César Ariza	- Ajuste ejemplo sección 6.7.
3.9	2009-03-10	César Ariza	- Ajuste ejemplo sección 6.7 espacios de nombres
4.0	2010-04-21	Roberto Contreras Heinsohn Business Technology	- Ajuste de logos del Ministerio y de Gobierno en Línea. - Ajuste del nombre del Ministerio. - Se eliminaron los literales que describían detalladamente el cómo se deberían cambiar las versiones de los esquemas dependiendo del cambio realizado. - Se eliminaron los literales que explicaban el cómo conseguir las fuentes de las enumeraciones. - Se agregó el literal para la implementación de los elementos de dato de tipo grupo - Se agregó el literal 6.1.8 sobre la codificación de los esquemas. - Ajustes de redacción
4.1	2010-06-30	Roberto Contreras Heinsohn Business Technology	- Se agregó el procedimiento que explica cómo se debe ajustar el paquete completo de esquemas del estándar, con el objetivo de sólo usar los esquemas que se necesitan en los servicios de intercambio de información. Capítulo 10 - Ajustes de redacción

TABLA DE CONTENIDO

LISTA DE FIGURAS Y TABLAS

DERECHOS DE AUTOR

CRÉDITOS

La información y datos contenidos en la versión 1.0 del documento fueron elaborados inicialmente por la Unión Temporal Hewlett Packard-Microsoft y TELECOM dentro del marco del proyecto Plataforma de Interoperabilidad – PDI, en julio de 2005.

A partir de la versión 1.0a del documento, la información y datos incluidos en este documento, hacen parte de las observaciones, comentarios, aportes e investigaciones realizadas por el Grupo de Trabajo Interinstitucional establecido para tal fin y denominado Equipo GEL-XLM. Este equipo ha estado conformado las siguientes Entidades:

- DEPARTAMENTO ADMINISTRATIVO NACIONAL DE ESTADÍSTICA – DANE.
- DEPARTAMENTO NACIONAL DE PLANEACIÓN – DNP.
- MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES: "PROGRAMA AGENDA DE CONECTIVIDAD".
- MINISTERIO DE HACIENDA Y CRÉDITO PÚBLICO.
- MINISTERIO DE LA PROTECCIÓN SOCIAL – MPS: PROGRAMA DE APOYO A LA REFORMA EN SALUD

El equipo GEL-XML ha tomado como base, para la ejecución de las actividades, aspectos metodológicos y las experiencias que sobre el tema se han liderado por parte de otros gobiernos, como los de Alemania[1], Nueva Zelanda[2], Australia[3] , Reino Unido[4] y Hong Kong[5], al igual que la Iniciativa de Metadatos *Dublín Core* – DCMI[6], para efectos de estandarizar el manejo de metadatos para los elementos de dato que se han identificado y para la inclusión de dichos metadatos dentro de los *esquemas* creados.

La empresa Informática Siglo 21, en el año 2008, propuso y aplicó recomendaciones sobre los documentos que definen y describen el lenguaje común de intercambio de información, sobre la estructura y funcionalidad respecto al organismo responsable de la administración y gestión del estándar, en el marco del proyecto de Consultoría para la Administración y Gestión del estándar.

[1] Tomado de www.osci.de el 22 de abril de 2008
[2] Tomado de www.e.govt.nz el 22 de abril de 2008
[3] Tomado de http://www.finance.gov.au/e-government/publications-and-reports.html#Australian20Technical20Framework el 23 de febrero de 2010
[4] Tomado de www.govtalk.gov.uk el 22 de abril de 2008
[5] Tomado de www.ogcio.gov.hk/eng/infra/eif.htm el 22 de abril de 2008
[6] Tomado de www.dublincore.org el 22 de abril de 2008

Las actualizaciones a este documento a partir de la versión 4.0 fueron realizadas por Heinsohn Business Technology, en desarrollo del contrato de mantenimiento del lenguaje común de intercambio de información, las cuales estuvieron enfocadas a la simplificación y evolución de ésta guía acorde con las nuevas necesidades generadas en el maderamiento del Lenguaje común de intercambio de información.

1 AUDIENCIA

Este documento está dirigido al personal que se encuentre a cargo de la administración del estándar. Este cuerpo regulador tiene la responsabilidad de asegurar que los diferentes entes que participan en Gobierno en Línea cumplan con todas las reglas establecidas en el lenguaje común de intercambio de información.

Adicionalmente aquellas entidades u organizaciones interesadas en participar en la iniciativa de Gobierno en Línea, encontrarán en este documento información técnica normativa de la creación de *esquemas* XML que cumplan con el estándar, que les permitirá intercambiar información para integrarse a esta estrategia.

Este documento no es un tutorial de los aspectos técnicos de XML o de *esquemas.* Para el entendimiento del contenido de este escrito se asume que el lector tiene conocimiento sobre XML, *esquemas* y la terminología relacionada. La figura 1 ilustra una guía para abordar el conocimiento y lectura de los documentos que sobre el estándar se han publicado a la fecha.

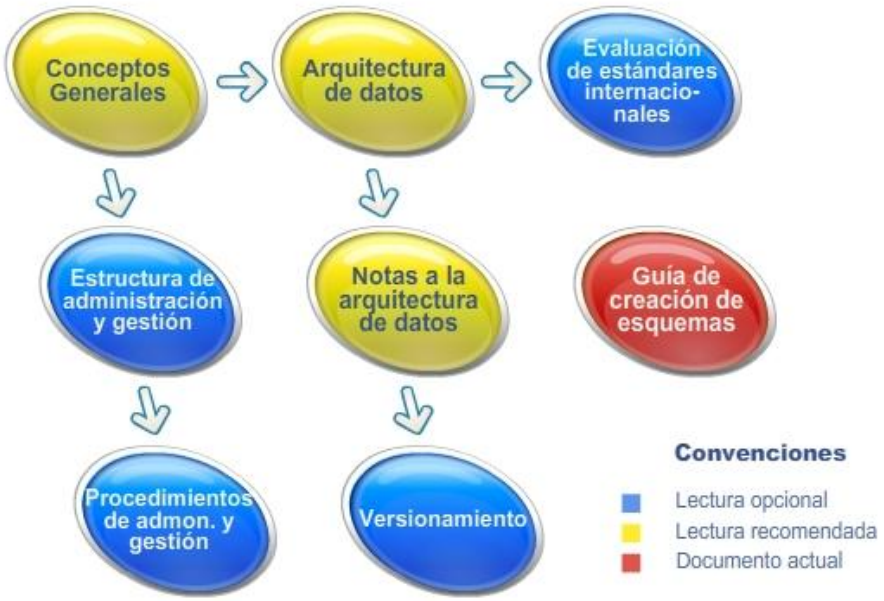


Figura 1. Lecturas recomendadas

2 REQUISITOS

El lector **DEBERÁ** tener sólidos conocimientos en XML[7]. Además, para contextualizar al lector, se recomienda la lectura previa de los siguientes documentos:

Documentos del lenguaje común de intercambio de información[8]

- Conceptos Generales
- Arquitectura de datos
- Notas a la Arquitectura de datos

Información complementaria

- Recomendación para la creación de *esquemas* de la W3C[9]
- Recomendación RFC-2119 sobre palabras clave de nivel de requerimiento (e.g. **DEBE**, **REQUERIDO, OBLIGATORIA, PUEDE**). La traducción de la recomendación está en apéndice 13.1 de este documento.

Si el lector desea crear o utilizar adaptadores deberá tener profundos conocimientos en:

- Expresiones XPath[10]
- Programación en XSL Transformations[11]

[7] Tomado de http://www.w3.org/XML/ el 23 de abril de 2008
[8] Tomado de http://www.gelxml.igob.gov.co/web/gelxml/documentacion/ el 26 de febrero de 2010
[9] Tomado de http://www.w3.org/XML/Schema el 22 de abril de 2008.
[10] XPath es el acrónimo de XML Path Languaje, es un lenguaje para seleccionar nodos en un documento XML. Para mayor información visitar http://www.w3.org/TR/xpath
[11] XSL es el acrónimo de Extensible Stylesheet Language (Lenguaje Extensible para Hojas de Estilo), para mayor información visitar http://www.w3.org/TR/xsl

3 INTRODUCCIÓN

El lenguaje común de intercambio de información, está definido como el estándar a utilizar en el ámbito de la iniciativa de Gobierno en Línea. El estándar está constituido por un juego de guías o reglas para la creación de documentos electrónicos. Estos documentos serán utilizados por diferentes aplicaciones para comunicarse con el núcleo de Gobierno en Línea.

El presente documento tiene como finalidad establecer las reglas básicas para la creación y mantenimiento de los *esquemas* dentro de la especificación del estándar. La recomendación base para la creación de *esquemas* XML la define el W3C (*World Wide Web Consortium*) y se encuentra en la página Web http://www.w3.org/TR/xml11/.

Se entiende por *esquema* en este documento un documento-XML que utiliza la definición *XML-Schema* [12] de la W3C.

El documento está divido en cuatro secciones así: una sección de Guías para la Generación de *esquemas*, donde se encuentran los lineamientos básicos de la generación de *esquemas*, la metodología de versionamiento y otras recomendaciones; una sección con guías para los componentes de *esquemas*, una sección con guías para el manejo de bibliotecas y una última sección con información de *metadatos*.

[12] Tomado de http://www.w3.org/TR/xml11/ el 4 de Junio de 2008

4 CONCEPTOS BÁSICOS

El estándar establece a XML como el lenguaje primario para la integración de datos en el marco de definiciones de la estrategia de Gobierno En Línea. XML permite codificar de manera sencilla y en formato texto la información del negocio requerida para las operaciones de Gobierno en Línea.

Los *esquemas* del estándar se adhieren a los estándares recomendados por W3C[13]. Dado que el W3C por medio de XML permite una gran flexibilidad para la creación de *esquemas*, el objetivo de este documento es proveer recomendaciones específicas y guías para el desarrollo de los *esquemas* dentro del marco lenguaje común de intercambio de información.

En particular, el estándar ofrece mecanismos para la reutilización de las definiciones; éstas **DEBEN** ser utilizadas selectivamente y manejadas cuidadosamente en el contexto de lenguaje común de intercambio de información. Es importante que la reutilización de los *esquemas* sea de fácil compresión para los desarrolladores de las aplicaciones; razón por la cual, las definiciones enfatizan en la simplicidad y facilidad de uso antes que la elegancia técnica.

Este documento integra muchos de los aspectos publicados en el documento *"e-Government Schema Guidelines for XML*[14]", Otros aspectos del documento se han adaptado conforme a la necesidades Colombianas y a la Arquitectura de datos del lenguaje.

[13] Siglas de *World Wide Web Consortium*, abreviadamente W3C, es una organización que produce estándares para la *World Wide Web*. Está conformada por representantes de aquellas compañías que tienen intereses en la Internet (http://www.w3.org/).
[14] Tomado de http://www.govtalk.gov.uk/schemasstandards/developerguide_document.asp?docnum=946 el 15 de enero de 2009 (*Crown Copyright 2002*)

5 ESTRUCTURA Y TERMINOLOGÍA

5.1 ESTRUCTURA

Este documento está dividido en varias secciones relacionadas con diferentes aspectos del diseño de *esquemas*:

- Guías para la generación de *esquemas*
- Guías de componentes de *esquemas*
- Bibliotecas comunes
- *Metadatos* y *Esquemas*

Para facilitar el aprendizaje y la comprensión de las reglas, cada sección incluye una introducción al tema y especifica un conjunto de guías que manejan la siguiente estructura:

- **"Guía":** Provee el resumen del requerimiento o recomendación a utilizar.
- **"Explicación":** Provee información más detallada sobre la razón para la adopción de la guía.
- **"Ejemplo":** Opcionalmente provee algunos ejemplos del uso de la guía en particular.

5.2 TERMINOLOGÍA

- Los términos ***Schema* XML, Esquema XML, Documento de *Schema* XML y Esquema** con frecuencia son utilizados indiferentemente para hacer referencia a los documentos XML que contienen elementos de *esquemas* expresados en XML como lo describe la recomendación del W3C.

- En este documento el término **Esquema** se acoge a la definición del W3C, como la estructura abstracta requerida para validar un documento XML**.** El término **esquema-XML** tiene el mismo significado. Durante el desarrollo del presente documento se hará mención al término *esquema*.

- Un **Documento XML** es una pieza completa de XML bien formado como lo define la recomendación de XML. Dado que la mayoría de los documentos XML en lenguaje común de intercambio de información son utilizados para intercambiar información entre diferentes sistemas de información, ocasionalmente también es denominado **Mensaje XML.**

- El término **Bibliotecas Comunes** son el conjunto de componentes del *esquema* que son usados en dos o más instancias.

- El término **Instancia** se refiere a un documento XML bien formado, que es válido según algún *esquema* del lenguaje común de intercambio de información. La forma de referirse a los elementos Documento será como **Instancia del Documento**. Nótese que no existen instancias de documento válidas a partir de un *esquema* basado en bibliotecas comunes. Las bibliotecas comunes son exclusivamente para reutilización.

- En este documento, el término **Elemento** se refiere a un segmento bien formado de XML en una instancia.

Ejemplo

```
<Ciudadano>
        (Otras definiciones)
</Ciudadano>
```

En este caso <Ciudadano> es un elemento.

- El término **componente** se refiere a la definición de un elemento en un esquema.

Ejemplo

```
<xsd:complexType name=”tipoCiudadano”>
    (Otras definiciones)
</xsd:complexType>
```

En este caso “tipoCiudadano” es un componente.

- Las palabras claves como "**DEBE**", "**NO DEBE**", "**REQUERIDO**","**OBLIGATORIO**", "**DEBERÁ**", "**NO DEBERÁ**", "**DEBERÍA**", "**NO DEBERÍA**", "**RECOMENDADO**", "**PUEDE**" y "**OPCIONAL**" en este documento serán interpretadas como se describe en RFC 2119. Ver apéndice 13.1.

6 GUÍAS PARA LA GENERACIÓN DE ESQUEMAS

Este capítulo provee directrices sobre el diseño de datos, los requerimientos y convenciones relacionados con los *esquemas* que se van a utilizar en el estándar. La generación de los *esquemas* basados en XML asegura la portabilidad y la disponibilidad de la información.

6.1 LINEAMIENTOS INICIALES

6.1.1 Lenguaje para la especificación de los *esquemas*

a) Guía

Los *esquemas* XML del W3C **DEBEN** ser utilizados como el lenguaje para la definición de documentos en el estándar.

Explicación

Es importante para la interoperabilidad de todos los sistemas de gobierno, el uso de un mecanismo único para la descripción de los documentos. En la actualidad solo dos mecanismos cubren los requerimientos del lenguaje común de intercambio de información, éstos son los DTD´s y los *esquemas*. De éstos los *esquemas* son preferibles por su soporte de *espacios de nombre*[15], tipos de datos y diseño modular.

b) Guía

Los *esquemas* **DEBEN** usar la cadena de texto "xsd" como prefijo para el *espacio de nombres* del lenguaje XML *esquema*[16].

Explicación

Esta es una convención aceptada internacionalmente para el manejo de XML.

[15] Un espacio de nombres (*namespace*) es un prefijo que se utiliza para cualificar (dar características) a nombres de elementos XML o atributos XML. El espacio de nombres permite que los atributos y elementos tengan nombres únicos dentro de un documento XML, evitando ambigüedades con otros elementos o atributos con nombres iguales. Por ejemplo, en la cadena "xsd:string" el prefijo del *espacio de nombres* es "xsd" y el nombre es "*string*".
[16] Tomado de http://www.w3.org/2001/XMLSchema el 1 de mayo de 2008

Ejemplo

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:secadq = "http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Proyectos/SECOP/Adquisicion/Adquisicion" targetNamespace=
"http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Proyectos/SECOP/Adquisicion" version="1.0"
id="cabeceraAdquisicion">
```

6.1.2 Detalles del diseño en XML

a) Guía

Todas las reglas de diseño de los *esquemas* **DEBEN** estar basadas en las recomendaciones del W3C.

Explicación

El utilizar las recomendaciones de la W3C permite aplicar las mejores prácticas en el diseño de los esquemas buscando que éstos sean fácilmente mantenibles en el tiempo, y minimice su complejidad de uso.

6.1.3 Patrones del diseño de *esquemas* XML

a) Guía

Se **DEBERÁ** utilizar patrones de diseño para la creación de *esquemas.* Se recomienda el uso del patrón de diseño "*Venetian Blind*[17]" por sus características de reutilización elementos.

Explicación

El Patrón de diseño *Venetian Blind* recomienda la creación de declaración de tipos de datos para luego definir elementos con dichos tipos de datos Los elementos-XML **DEBEN** estar en esquemas independientes.

Ejemplo

A continuación se presenta un ejemplo del patrón *Venetian Blind;* el ejemplo define cinco elementos. Cada uno de los elementos-XML (que definen un elemento-XML)

[17] Tomado de http://developers.sun.com/jsenterprise/nb_enterprise_pack/reference/techart/design_patterns.html el 22 de abril de 2008.

del 1 al 4 **PUEDE** estar en esquemas independientes. El elemento-XML 4 es el que agrupa los demás elementos-XML completando así el patrón *Venetian Blind.* El elemento-XML 5 es la utilización del elemento-XML 4.

```
Definición del elemento-XML 1:
<xsd:simpleType name="enumTitulo">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Sr."/>
                <xsd:enumeration value="Sra."/>
                <xsd:enumeration value="Ing."/>
        </xsd:restriction>
</xsd:simpleType>
…

Definición del elemento-XML 2:
<xsd:simpleType name="tipoPrimerNombre">
        <xsd:restriction base="xsd:string">
                <xsd:minLength value="1"/>
        </xsd:restriction>
</xsd:simpleType>
…

Definición del elemento-XML 3:
<xsd:simpleType name="tipoPrimerApellido">
        <xsd:restriction base="xsd:string">
                <xsd:minLength value="1"/>
        </xsd:restriction>
</xsd:simpleType>
…

Definición del elemento-XML 4:
<xsd:complexType name="tipoNombre">
        <xsd:sequence>
                <xsd:element name="Titulo" type=" enumTitulo"/>
                <xsd:element name="Nombre" type="tipoPrimerNombre"/>
                <xsd:element name="Apellido" type="tipoPrimerApellido"/>
        </xsd:sequence>
</xsd:complexType>
…

Definición del elemento-XML 5:
<xsd:element name="NombrePersona" type="tipoNombre"/>
```

6.1.4 Complejidad de los *esquemas*

a) Guía

Las características menos comunes de los *esquemas* **NO DEBERÁN** ser utilizadas cuando existan opciones más simples. Los desarrolladores **DEBERÁN** tener en cuenta la complejidad de las pruebas sobre los *esquemas*.

Explicación

Esta es una de las reglas más importantes. El lenguaje XML es enormemente poderoso y flexible en cuanto a la definición de *esquemas* se refiere. En muchos casos los *esquemas* pueden simplificarse y aún alcanzar el mismo fin. Siendo esta tecnología novedosa, son muchas las personas que necesitarán hacer uso de los *esquemas*, y algunas tendrán poca experiencia con los mismos, por lo que en estas circunstancias, la mejor estrategia es mantener la simplicidad.

Adicionalmente, muchas herramientas para el desarrollo de *esquemas*, tienden a contener errores *(bugs)* en las opciones menos populares. Los *esquemas* simples no solo son más fáciles de probar, sino que también causarán menos confusión al no exponer debilidades en las herramientas de desarrollo.

6.1.5 Modelar datos, no formularios

a) Guía

Los *esquemas* XML **DEBERÁN** modelar los datos requeridos por las aplicaciones, en lugar de formularios o formatos de mensaje predefinidos. Si bien es cierto que los formularios constituyen un buen punto de referencia, **NO DEBERÁN** dominar el diseño final.

Explicación

Hay dos razones para esto. La primera es que los formularios son diseñados para ser utilizados en papel, no en la pantalla de un computador, adicionalmente los formatos preexistentes pueden no representar en forma estructurada y lógica las necesidades reales de los datos en el sistema. En segundo lugar, todos los *esquemas del estándar* **DEBERÁN** derivarse de las necesidades de datos directamente, en lugar de seguir ciegamente formatos preestablecidos.

6.1.6 Nombre de los archivos que contienen los *esquemas*

a) Guía

El nombre de los archivos de los esquemas **DEBEN** utilizar el identificador del tipo de los elementos que contienen (prefijo más identificador). En caso que el esquema sea una biblioteca se **DEBERÁ** utilizar el formato UCC[18], y en caso que sea un elemento de dato se utilizará el formato LCC[19].

Explicación

Esta regla permite identificar fácilmente cada palabra utilizada en el nombramiento y además relacionar los elementos con el archivo que los contiene.

Ejemplo

PresentacionProyectos.xsd O tipoCiudadano.xsd

b) Guía

Los nombres de los *esquemas* **DEBEN** reflejar aspectos funcionales del mensaje que representan. Se **DEBERÍA** utilizar el mismo nombre del elemento de dato que representan y en consecuencia, seguir las reglas de nombramiento para elementos de datos del documento de Arquitectura de Datos.

Explicación

Esta regla busca facilitar la comprensión de los *esquemas*. En la mayoría de los casos, si el mensaje representa un proceso, el nombre del *esquema* podrá ser igual al nombre del proceso.

Ejemplo

tipoCertificadoJudicialConsularEnt.xsd es el elemento de dato que contiene la información de entrada del certificado judicial consular. Éste es expedido por el funcionario consular, de acuerdo con la información suministrada por el departamento administrativo de seguridad DAS, en el cual se certifica que el titular no tiene asuntos pendientes con las autoridades judiciales y de policía

[18] UCC Acrónimo de la lengua inglesa para representar *Upper Camel Case*, que significa que todos los inicios de palabras comienzan con mayúsculas. Ejemplo: TodasEnMayusculas.
[19] LCC Acrónimo de la lengua inglesa para representar *Lower Camel Case*, que significa que todos los inicios de palabras comienzan con mayúsculas, excepto la primera palabra. Ejemplo: primeraEnMinuscula.

6.1.7 Reutilización de elementos

a) Guía

Se **DEBERÁN** reutilizar los elementos que tengan la misma semántica. La reutilización se **DEBERÁ** realizar por medio de la inclusión del esquema o la importación de la librería que contiene elemento de dato junto con el uso de los atributos *name* y *type* dentro de la etiqueta *element*.

Explicación

La reutilización de elementos facilita el intercambio entre entidades.

Ejemplo

En el esquema de la fecha (apenas un extracto para ilustrar la reutilización, nótese la creación del elemento fechaExpedicion**):**

```
…
xmlns:comtem="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Temporal"
…
<xsd:simpleType name="tipoFecha">
        <xsd:restriction base="xsd:date">
                <xsd:minInclusive value="0001-01-01"/>
        </xsd:restriction>
</xsd:simpleType>

…
```

En el esquema que usa la fecha (note la referencia al elemento de dato tipoFecha creado en el esquema de arriba)

```
…
xmlns:comtem="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Temporal"
…
    <xsd:import namespace="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Temporal" schemaLocation="../../../Comun/Temporal/
Temporal.xsd"/>
…

<xsd:complexType name="tipoDatoPersonal">
        <xsd:sequence>
                …
                <xsd:element name="fechaExpedicion" type="comtem:tipoFecha"/>

                …
        </xsd:sequence>
</xsd:complexType>
```

6.1.8 Codificación de los esquemas

a) Guía

El conjunto de caracteres que se **DEBE** usar para codificar los esquemas es la norma ISO-8859-1[20].

Explicación

El conjunto de caracteres incluidos en esta norma soportan todos los caracteres especiales que se utilizan en el lenguaje Español.

Ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

6.2 VERSIONAMIENTO

6.2.1 Políticas de versionamiento

a) Guía

El ciclo de versionamiento de los *esquemas* de Gobierno en Línea será independiente del ciclo de versionamiento del estándar y también será independiente de la versión de la definición del elemento de dato (arquitectura y plantilla de metadatos). Sin embargo, los *esquemas* **DEBERÁN** cumplir con la versión más reciente de la arquitectura al momento de su creación.

Explicación

El objetivo de utilizar la política de versiones es solucionar inconvenientes de incompatibilidad entre dos liberaciones sucesivas de *esquemas* de Gobierno en Línea. Así mismo solucionar los problemas presentados al realizar cualquier tipo de cambio en las bibliotecas comunes.

El ciclo de versionamiento del estándar **PUEDE** ser largo, dado que un cambio en la especificación **DEBE** ser estudiado cuidadosamente. Los *esquemas* generados bajo una versión dada del lenguaje común de intercambio de información no serán forzados a cambiar cuando cambie la versión del estándar. De la misma manera, los *esquemas* generados bajo una versión del lenguaje común de intercambio de

[20] Tomado de http://www.w3.org/TR/html4/sgml/entities.html#h-24.1 el 21 de abril de 2010

información no tienen que esperar un cambio en la versión del estándar para ser publicados bajo una nueva versión.

6.2.2 Mecanismo propuesto para el manejo de versiones

a) Guía

Los *esquemas* no soportarán la compatibilidad hacia adelante.

Explicación

Ejemplificando el concepto podemos decir que la compatibilidad hacia adelante se define como la situación en que una persona llamada Remitente o Emisor crea y envía una Instancia basada en un nuevo *esquema* a otra persona llamada Recipiente o Receptor, el cual al hacer la recepción de la instancia, la valida contra un *esquema viejo* y éste **DEBE** validarse sin que se generen fallas por tratarse de una instancia de esquema posterior.

La compatibilidad hacia adelante se define como la capacidad de diseñar *esquemas* tales que hasta el *esquema* más viejo **PUEDE** validar los documentos de caso creados siguiendo la versión más reciente definida.

6.2.3 Reflejar los números de la versión en el esquema

a) Guía

Tanto las versiones mayores como las versiones menores serán representadas usando solo caracteres numéricos. La representación completa de la versión será del formato mm.nn**,** donde mm es un número que representa la versión mayor y nn es un número que representa la versión menor. Todos los *esquemas* **DEBEN** tener la versión en el atributo *versión* del elemento *xsd:schema***.**
Para reflejar el cambio de una versión se **DEBE** cambiar el número de la versión en el atributo.

Explicación

Esta es una guía que provee consistencia al estándar.

Ejemplo

```
<xsd:schema
targetNamespace="http://www.gobiernorrnlinea.gov.co/GEL-
XML/1.0/schemas/proyectos/artesanias/presentacionproyectos/PresentacionProye
ctos"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:arte="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/proyectos/artesanias/presentacionproyectos/PresentacionProyectos" version="1.2">

Pasa a:

<xsd:schema targetNamespace="http://www.gobiernorrnlinea.gov.co/GEL-XML/1.0/schemas/proyectos/artesanias/presentacionproyectos/PresentacionProyectos"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:arte="http://www.gobiernorrnlinea.gov.co/GEL-XML/1.0/schemas/proyectos/artesanias/presentacionproyectos/PresentacionProyectos" version="1.3">
```

6.2.4 Las reglas de compatibilidad de versiones

a) Guía

Todas las versiones menores de un *esquema* dentro de una versión mayor serán compatibles hacia atrás.

Explicación

Las versiones menores son las liberaciones intermedias de un *esquema* XML que contiene sólo los cambios que son considerados compatibles hacia atrás con la versión existente de este *esquema*.

Las versiones mayores son las liberaciones de un conjunto de *esquemas* XML que contiene los cambios que no son compatibles hacia adelante con el conjunto existente de los mismos *esquemas*.

6.3 RELATIVAS A LOS ELEMENTOS

6.3.1 Modelado de los elementos

a) Guía

Los tipos de datos básicos definidos por WC3 **DEBEN** ser usados siempre que se pueda, en lugar de crear tipos equivalentes con otros nombres.

Explicación

Los tipos básicos están bien definidos por el WC3 y por tanto son de amplia comprensión por parte de los desarrolladores. Usar otros nombres para los mismos tipos puede causar confusión y posiblemente problemas durante la validación.

b) Guía

Se **DEBE** usar la etiqueta *complexContent* cuando se requiera.

Explicación

El uso de esta etiqueta permite adicionar restricciones o extensiones (herencia) sobre un elemento de tipo compuesto.

Ejemplo

```
…
<xsd:complexType name="grupoNumeroIdentificacionDIAN">
<xsd:complexContent>
<xsd:extension base="locide:grupoNumeroIdentificacion">
    <xsd:choice>
      <xsd:element name="numIdExtranjero"
                type="locide:tipoNumIdExtranjeroDIAN"/>
      <xsd:element name="numIdPersonaJuridicaExtranjero"
                type="locide:tipoNumIdPJExtranjeroDIAN"/>
    </xsd:choice>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

c) Guía

Para modelar la estructura de un elemento compuesto se **DEBE** utilizar la etiqueta <xsd:sequence>.

Explicación

La etiqueta <*xsd:sequence*> permite indicar los elementos hijos que lo componen y el orden en el que **DEBEN** ser leídos.

Ejemplo

```
<xsd:complexType name="tipoNomPersona">
        <xsd:sequence>
                <xsd:element name="primerApellido"
                type="comper:tipoPrimerApellido"/>
                <xsd:element name="tipoSegundoApellido"
                type="comper:tipoSegundoApellido"/>
                <xsd:element name="primerNombre"
                type="comper:tipoPrimerNombre"/>
                <xsd:element name="segundoNombre"
```

```
                type="comper:tipoSegundoNombre" />
        </xsd:sequence>
</xsd:complexType>
```

6.3.2 *Data Types* vs. Declaración de Elementos

a) Guía

En muchos casos se presentará la disyuntiva sobre si un componente reutilizable **DEBE** ser definido como un *tipo de dato* (simple o compuesto) o como una declaración de elemento. Un componente **PODRÍA** ser definido como *tipo de dato* sí:

- Va a ser utilizado con diferentes nombres en diferentes contextos.
- Se espera que otros tipos sean derivados de él.

Un componente **PODRÍA** ser definido como una declaración de un elemento sí:

- No existe la intención de derivar otros elementos de él.
- El elemento será utilizado siempre con el mismo nombre.

Todos los elementos de la capa Tipos de datos GEL-XML **DEBERÁN** ser definidos como tipos de datos.

Explicación

Se presentarán múltiples situaciones en las que un elemento **DEBE** utilizarse sin cambiar su nombre. Por ejemplo el Número de Cédula de un ciudadano siempre **DEBE** aparecer como numeroCedula, de manera que su semántica sea siempre clara y cuando dos sistemas independientes lo utilicen, será claro que se refieren al mismo elemento de dato.

Sin embargo, en otras circunstancias no es apropiado fijar el nombre de un elemento de dato. Por ejemplo, una dirección tiene siempre la misma estructura, pero **PUEDE** aparecer como elementos diferentes con los nombres direccionCasa, direccionTrabajo o direccionCorrespondencia. En este caso, el elemento Dirección **DEBE** ser definido como un tipo de dato. En este caso se dice que direccionCasa o direccionTrabajo son *usos* de Dirección, de acuerdo con la arquitectura definida.

El otro criterio para decidir entre una declaración o un tipo, es la necesidad de derivar otros elementos de él. En este caso el uso de tipos de datos permite una semántica más clara.

Ejemplos

En el ejemplo se muestra la reutilización del elemento lugar y de elementos de la capa Tipos de Dato GEL-XML que son definidos como tipos de dato.

Ejemplo de utilización del tipo de dato cadena256, de la capa Tipos de Dato GEL-XML (ver elemento nombreDireccion).

```
<xsd:complexType name="tipoDireccion">
        <xsd:sequence>
                <xsd:element name=" codPais"  type="comubi:tipoCodPais" />
                <xsd:element name="codDivisionTerritorial"
                type=" comubi:codDivisionTerritorial" minOccurs="0"/>
                <xsd:element name="nomCiudad" type="comubi:nomCiudad" />
                <xsd:element name="nombreDireccion"
                        type="geltd:tipoCadena256”/>
                <xsd:element name="zonaPostal" type="comubi:tipoZonaPostal"
                minOccurs="0"/>
        </xsd:sequence>
</xsd:complexType>
```

Ejemplo de la reutilización del elemento de dato lugar:

Esquema del Elemento de dato lugar, incluyendo sus usos.

```
<xsd:complexType name="tipoTipoLugar">
        <xsd:sequence>
                <xsd:element name=" codPais"  type="comubi:tipoCodPais" />
                <xsd:element name="codDivisionTerritorial"
                type=" comubi:codDivisionTerritorial" minOccurs="0"/>
                <xsd:element name="pais  type="comubi:tipoPais"/>
                <xsd:element name="divisionTerritorial
                name="comubi:tipoDivisionTerritorial "/>
                <xsd:element name="subDivisionTerritorial"
                name="comubi:tipoSubDivisionTerritorial"/>
        </xsd:sequence>
</xsd:complexType>
```

Utilización de los usos lugarExpedición y lugarNacimiento, del elemento de dato Lugar, dentro del elemento de dato tipoDatoPersonal

```
<xsd:complexType name="tipoDatoPersonal">
        <xsd:sequence>
                <xsd:element name="nomPersona"
                type="comper:tipoNomPersona"/>
                <xsd:element name="codTipoIdIdPersona"
```

```
                type="locide:tipoCodTipoIdIdPersona"/>
                …
                <xsd:element name="lugarNacimiento"
                type="comubi:tipoTipoLugar"/>
                …
                <xsd:element name="lugarExpedicion"
                type="comubi:tipoTipoLugar"/>
                …
        </xsd:sequence>
</xsd:complexType>
```

b) Guía

Los *esquemas* del estándar **NO DEBEN** tener un *espacio de nombres* por defecto.

Explicación

Dado que el lenguaje común de intercambio de información está construido bajo una arquitectura de capas y áreas o módulos, la utilización de múltiples *espacios de nombres* permite, que los diferentes componentes mantengan la ubicación en la cual fueron definidos, permitiendo tener una consistencia entre sí.

6.3.3 Atributos vs. Elementos

a) Guía

Los *esquemas* del estándar **DEBEN** estar definidos de manera que los elementos sean los principales contenedores de información. Los atributos son más apropiados para almacenar datos adicionales y elementos simples que proveen más información sobre el contenido. Los atributos **NO DEBEN** ser utilizados para calificar a otros atributos, ya que esto causa confusión.

Explicación

A diferencia de los elementos, los atributos no pueden almacenar datos estructurados. Por esta razón, los elementos son el mecanismo preferido para almacenar datos. Los atributos son más apropiados para almacenar metadatos acerca de los elementos.

Ejemplo

Una fecha **PUEDE** ser representada en un mensaje como:

```
        <FechaNacimiento>2002-06-11</FechaNacimiento>
```

Si se requiere información adicional, como por ejemplo, el método de verificación, **PUEDE** usarse un atributo como el siguiente:

```
<fechaNacimiento verificadaMediante=”Visualización de cedula”>
        2002-06-11
</fechaNacimiento>
```

La siguiente estructura sería inapropiada:

```
<fechaNacimiento codigo=”2” verificadaMediante=”Visualización de cedula”>
   1965-06-06
</fechaNacimiento>
```

En este caso, no es claro si el atributo código aplica a “verificadaMediante” o al elemento en sí. La estructura correcta en este caso sería:

```
<fechaNacimientoVerificada >
        <VerificadaMediante >
                Visualización de cedula
        </VerificadaMediante>
        <codigoVerificacion>
                2
        </codigoVerificacion>
        < fechaNacimiento>2005-12-04</fechaNacimiento >
</fechaNacimientoVerificada>
```

6.4 OTRAS RECOMENDACIONES

6.4.1 Referencias absolutas y relativas

a) Guía

Cuando los *esquemas* estén estrechamente relacionados de tal forma que se espere que siempre sean almacenados juntos en una estructura de directorios consistente, las sentencias *import* y *export* **DEBERÁN** usar referencias relativas.

Explicación

Es posible que los *esquemas* sean movidos de un lugar a otro. En estos casos, es importante que las referencias a otros *esquemas* se mantengan. Los grupos de *esquemas* que probablemente sean movidos juntos pueden preservar las relaciones usando referencias relativas. Si este no es el caso, se necesitan referencias absolutas.

b) Guía

Los *esquemas* del estándar estarán almacenados organizadamente en una estructura de directorios que represente la taxonomía de la arquitectura del Lenguaje común de intercambio de información.

Explicación

Almacenar los *esquemas* de manera organizada facilita el uso y mantenimiento de los mismos.

Ejemplo

Esquema propuesto:

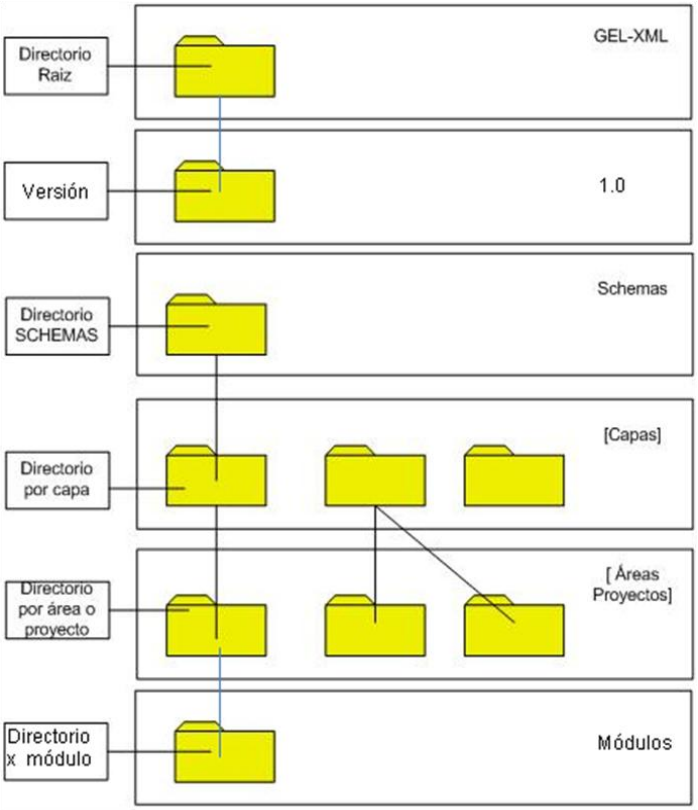


Figura 2. Estructura de directorios para el manejo de *Esquemas*

6.5 REPRESENTANDO CONDICIONES ALTERNAS

a) Guía

Las condiciones alternas (una condición alterna denota un estado que puede tener un componente) **DEBEN** ser representadas utilizando elementos en lugar de codificarlas mediante la presencia o ausencia.

Explicación

En XML es posible tener un elemento opcional, lo cual permite codificar la presencia de un elemento como un "SI" y la ausencia como un "NO". En el lenguaje común de intercambio de información para aumentar la claridad de los *esquemas*, los elementos siempre **DEBEN** estar presentes y se incluye un elemento cuyos valores serán los literales "SI" o "NO".

Ejemplo

Supongamos que tenemos un elemento que representa a un ciudadano y que a su vez ésta tiene un hijo para indicar si es un contribuyente especial:

```
<Ciudadano>
        … (otros datos)
        <ContribuyenteEspecial/>
</Ciudadano>
```

Un estilo para representar el hecho de que el ciudadano NO es un contribuyente especial, es omitir el hijo, esta manera de representación es válida para XML pero no para el estándar. La falta del elemento <ContribuyenteEspecial/> **PUEDE** interpretarse como que el ciudadano NO es un contribuyente especial:

```
<Ciudadano>
        … (otros datos)
</Ciudadano>
```

Para evitar este tipo de dualidades, en las interpretaciones, en los *esquemas* del estándar se **DEBEN** representar de la siguiente manera:

```
<Ciudadano>
                … (otros datos)
                <ContribuyenteEspecial>NO </ContribuyenteEspecial/>
</Ciudadano>
```

6.6 COMENTARIOS EN LOS ESQUEMAS

a) Guía

En el estándar, los comentarios sobre los esquemas no se **DEBEN** expresar utilizando comentarios XML.

Explicación

La forma de realizar comentarios sobre los esquemas es utilizando las etiquetas *xsd:annotation , xsd:appInfo* y *xsd:documentation*. La ventaja principal de usar esta estructura, en contraposición a utilizar comentarios XML es que pueden ser procesados con hojas de estilos para producir documentación de usuario. Además, se **DEBERÁN** utilizar elementos *Dublin Core* para incluir los comentarios y especificar el lenguaje de los comentarios con la marca *xml:lang*.
La etiqueta *xsd:annotation* describe el elemento padre, y las etiquetas hijo *xsd:documentation* describe el esquema para lectores humanos y *xsd:appInfo* describe el esquema para programas de computador.

Ejemplo

```
<xsd:annotation>
        <xsd:appinfo>
                <dc:identifier>http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Proyectos/SECOP/Adquisicion/tipoLineaItemAdj</dc:identifier>
                <dc:creator xml:lang="es"> Ministerio de tecnologías de la Información y las Comunicaciones: Programa Agenda de Conectividad</dc:creator>
                <dc:issued>2007-12-19</dc:issued>
                <dc:description xml:lang="es">Elementos de identificación y descriptivos básicos de bienes, servicios u obras para los distintos elementos involucrados en el proceso de contratación con información especifica de la adjudicación.</dc:description>
                <dc:hasVersion>1.0</dc:hasVersion>
                …
                <xsd:annotation><xsd:documentation>Dirección de Desarrollo</xsd:documentation></xsd:annotation>
        </xsd:appinfo>
</xsd:annotation>
```

6.7 MECANISMOS DE REUTILIZACIÓN

a) Guía

El uso de *xsd:redefine* en lo posible **DEBE** evitarse.

Explicación

Esta regla evita efectos secundarios en componentes reutilizados, al tiempo que aumenta la claridad y legibilidad.

Ejemplo

Esquema principal:

```
<xsd:complexType name="tipoDatoPersonal">
        <xsd:sequence>
                <xsd:element name="nomPersona"
                type="comper:tipoNomPersona"/>
                …
                <xsd:element name=" lugarExpedicion"
                type="comubi:tipoLugar"/>
                <xsd:element name="fechaExpedicion" type="comtem:tipoFecha"/>
                <xsd:element name="comubi:lugarNacimiento"
                type="comubi:tipoLugar"/>
                …
        </xsd:sequence>
</xsd:complexType>
```

Esquema redefinido (la lectura es compleja):

```
<xs:redefine schemaLocation="[…]../esquemaprincipal.xsd">
      <xs:complexType name="tipoDatoPersonal">
        <xs:complexContent>
                <xs:extension base="tipoDatoPersonal">
                        <xs:sequence>
                                <xs:element name="PaisResidencia"
                                type="comubi:tipoPais"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
      </xs:complexType>
</xs:redefine>
```

b) Guía

El elemento *xsd:import* **NO DEBE** usarse sin *espacio de nombres*.

Explicación

Si es usado sin *espacio de nombres*, *xsd:import* permite referenciar elementos externos sin *targetNamespace*. Esto causa que los *esquemas* sean difíciles de depurar y actualizar.

c) Guía

La instrucción <*xsd:include*> **DEBE** ser utilizada, siempre y cuando el esquema incluido tenga el mismo *espacio de nombres* que el que lo incluye.

Explicación

Esta instrucción permite la reutilización de los *esquemas*. Sin embargo, se pueden presentar problemas si se utiliza para incluir *esquemas* que tengan un *espacio de nombres* diferente al contexto donde se utiliza. En estos casos **DEBE** usarse la instrucción <xsd:import>.

d) Guía

La sentencia <*xsd:import*> siempre **DEBE** incluir el atributo *schemaLocation.*

Explicación

Mediante este atributo puede localizarse el *esquema* que está siendo importado.

e) Guía

Cuando se utilice la sentencia <*xsd:import*> **DEBEN** evitarse referencias circulares.

Explicación

Una referencia circular puede causar problemas a los sistemas de validación.

f) Guía

Cuando se utilice la sentencia <*xsd:import*> **DEBEN** evitarse las declaraciones repetidas.

Explicación

Las declaraciones repetidas aumentan la complejidad del esquema innecesariamente.

Ejemplo

```
<xsd:import targetNamespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
schemaLocation="../../../Comun/General/tipoCantidad.xsd" />
<xsd:import targetNamespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
schemaLocation="../../../Comun/General/tipoCantidad.xsd" />
```

g) Guía

Al incluir otros esquemas con la etiqueta *import*, se **DEBE** utilizar un único elemento *import* por cada espacio de nombres incluido.

Explicación

Debido a que algunos validadores no implementan en su totalidad las reglas del lenguaje de creación de esquemas de la W3C en lo referente a inclusión de otros esquemas con la etiqueta *import*, el seguimiento de esta guía facilita la validación de los esquemas con validadores que no implementan dichas reglas.

Ejemplo

Suponiendo que un elemento compuesto de la capa Uso Proyectos importa elementos del área General de la capa Uso Común, un uso incorrecto de un espacio de nombres con múltiples etiquetas *import* es como sigue:

```
<xsd:schema … targetNamespace= "http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Proyectos/[proyecto]/[módulo]" …
xlmns:comgen:"http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
…
<xsd:import namespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
schemaLocation="../../../Comun/General/tipoCantidad.xsd" />

<xsd:import namespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
schemaLocation="../../../Comun/General/tipoMagnitud.xsd" />
```

El uso correcto de un espacio de nombres con una única etiqueta *import* es como sigue (nótese el uso de la biblioteca *General.xsd* que deberá agrupar los esquemas incluidos en el ejemplo anterior):

```
<xsd:schema … targetNamespace= "http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Proyectos/[proyecto]/[módulo]"
xlmns:comgen:"http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
…
<xsd:import namespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/General"
schemaLocation="../../../Comun/General/General.xsd" />
```

h) Guía

Al incluir esquemas con la etiqueta *include* en bibliotecas **NO SE DEBE** incluir esquemas con elementos-XML que ya han sido incluidos por otros elementos (inclusión indirecta).

Explicación

La inclusión única de un elemento en una biblioteca evita que se generen referencias circulares.

Ejemplo

Teniendo el elemento de dato tipoDepartamento, que es compuesto por los elementos de dato enumCodDepartamentoAlf2 y enumNomDepartamento, una forma incorrecta de incluir elementos dentro de una biblioteca con la etiqueta *include* es como sigue (el error radica en que tipoDepartamento incluye previamente los otros dos elementos de dato):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Local/Ubicacion" elementFormDefault="qualified" version="2.0">
…
<xsd:include schemaLocation="enumNomDepartamento.xsd"/>
<xsd:include schemaLocation="enumCodDepartamentoAlf2.xsd"/>
<xsd:include schemaLocation="tipoDepartamento.xsd"/>
…
</xsd:schema>
```

El uso correcto de inclusión se presenta a continuación, solo debe ser incluido el elemento de dato tipoDepartamento :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Local/Ubicacion" elementFormDefault="qualified" version="2.0">
…
<xsd:include schemaLocation="tipoDepartamento.xsd"/>
…
</xsd:schema>
```

i) Guía

Se **DEBERÁ** utilizar un módulo común en cada proyecto, de la capa de uso Proyectos o capa de uso Plataforma de Interoperabilidad, cuando existan elementos de dato que sean utilizados por otros elementos de dato que se encuentren en varios módulos del mismo proyecto. Se **RECOMIENDA** nombrar al

módulo común con la palabra *core*. Si un elemento de dato importa otros elementos de dato de la misma capa, éstos **DEBERÁN** estar ubicados en el mismo módulo o área.

Explicación

El uso de una estructura de árbol para importar elementos evita las referencias circulares.

Ejemplo

Como lo muestra la sección izquierda de la Figura 3, se tiene el elemento A que requiere los elementos B y C (y son incluidos mediante la sentencia import), los tres elementos están en la misma capa, pero en diferente área o módulo; lo que puede generar referencias circulares si son importados en otras capas. La sección derecha de la Figura 3 muestra la forma correcta de incluir elementos de la misma área o módulo para evitar así referencias circulares.

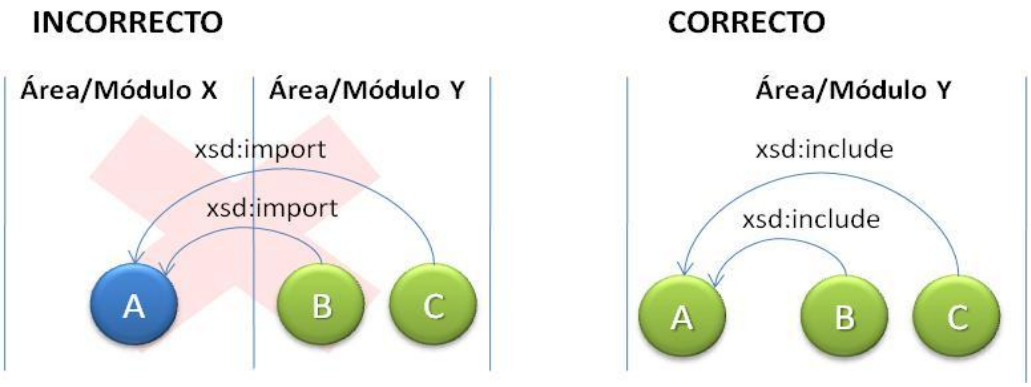


Figura 3. Inclusión de elementos de área o módulos iguales

j) Guía

El esquema XML de una biblioteca **DEBE** incluir los esquemas de elementos de dato que no estén incluidos dentro de otros elementos de dato de la misma área o módulo. Consecuentemente una biblioteca **NO DEBE** incluir aquellos esquemas que estén incluidos dentro de otros elementos de dato de la misma área o módulo.

Explicación

El uso de una estructura de árbol para importar elementos evita las referencias circulares o doble inclusión.

Ejemplo

La sección izquierda de la Figura 4 presenta una forma incorrecta de incluir elementos, ya que un elemento de dato (B) es incluido varias veces: por la biblioteca (Bibliotk.) y por el elemento que lo contiene (A). Por su parte, la sección

derecha de la misma figura muestra una correcta inclusión del elemento (B), ya que solo es incluido por el elemento (A) y esta a su vez, por la biblioteca (Bibliotk.).

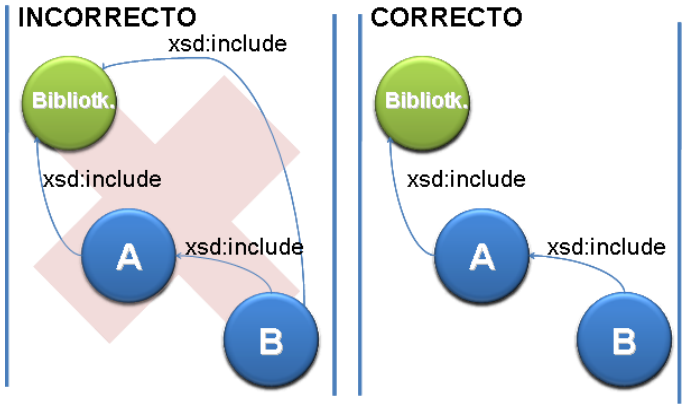


Figura 4. Inclusión de elementos en bibliotecas

k) Guía

Para el uso de elementos XML de otros estándares se **DEBE** utilizar el mismo espacio de nombres del estándar externo y en el atributo *schemaLocation* la ubicación donde **DEBE** estar el esquema XML externo.

Explicación

Se **DEBEN** conservar los espacios de nombres de los elementos de otros estándares.

Ejemplo

El siguiente recuadro muestra cómo incluir un elemento XML externo (CoordinatesType) dentro de un esquema del estándar. Nótese que el espacio de nombres con prefijo *gmlbas* (*xmlns:gmlbas="http://www.opengis.net/gml*) es incluido con el elemento *xsd:import,* conservando el espacio de nombres original y utilizando una ubicación (*schemaLocation*) en donde está el esquema XML externo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
xmlns:gmlbas="http://www.opengis.net/gml" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
targetNamespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Local/Ubicacion"
elementFormDefault="qualified" version="1.0" id="tipoCoordenadaPlanaIGAC">
<xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.0.0/base/basicTypes.xsd"/>
 <xsd:complexType name="tipoTipoCoordenadaPlanaIGAC">
   <xsd:sequence>
     <xsd:element name="coordinatesTypeIGAC" type="gmlbas:CoordinatesType"/>
     <xsd:element name="dimensionIGAC" type="xsd:int"/>
  </xsd:sequence>
 </xsd:complexType>
```

7 GUÍAS DE COMPONENTES DE ESQUEMAS

Este capítulo provee directrices sobre las especificaciones, los requerimientos y convenciones relacionados con los componentes de los *esquemas* que se van a utilizar en el lenguaje común de intercambio de información.

7.1 CONVENCIONES DE NOMBRES

a) Guía

Todos los nombres de los componentes y tipos de datos **DEBEN** ir en español. En el caso de términos técnicos o específicos a un campo **PUEDE** usarse el idioma en que normalmente son utilizados. Sin embargo, la inclusión de términos en otros idiomas **DEBE** hacerse con sumo cuidado y tomando en cuenta la legibilidad de los componentes. Los campos de *documentation* **PUEDEN** usarse para aclarar términos en otros idiomas en conjunto con *xml:lang* y *annotation*.

Explicación

Se aumenta la facilidad de comprensión de los *esquemas* considerando diferentes audiencias.

b) Guía

Cuando se utilicen componentes definidos en estándares Internacionales **PUEDEN** traducirse al español mediante un *Sustitutiongroup* el cual estará ubicado en una biblioteca común ubicada como corresponde en la Capa de estándares Internacionales.

Explicación

Esta guía permite mantener la unicidad en el idioma utilizado por todos los *esquemas* del estándar, con el fin de facilitar su comprensión y entendimiento considerando diferentes audiencias.

c) Guía

Los nombres de los elementos de datos **DEBERÁN** comenzar por ***tipo,*** excepto para los elementos de dato que correspondan a enumeraciones, que comenzarán por ***enum,*** y los elementos de dato que representen grupos, que comenzarán por el prefijo ***grupo.***

Explicación

Esto provee un marco consistente que aumenta la legibilidad de los *esquemas*.

Ejemplo

tipoDepartamamento se utilizaría para representar un departamento,

O

Si tenemos una lista con todos los departamentos, se **DEBERÍA** identificar con el siguiente nombre: **enumDepartamentos**

O

Si tenemos un tipo simple para representar el Número de Cédula o la Tarjeta de Identidad, se **DEBERÍA** identificar con el siguiente nombre: **grupoTipoDocumentoIdentificación**.

d) Guía

NO DEBEN usarse abreviaciones o acrónimos. Los nombres extremadamente largos **DEBERÁN** evitarse; en su lugar, **DEBEN** diseñarse nombres concisos e informativos. Abreviaturas o acrónimos ampliamente conocidos y únicos **PUEDEN** ser utilizados, utilizando la convención *LCC*[21]. Sin embargo, **DEBE** tomarse en cuenta que una abreviatura puede ser muy clara para un grupo de trabajo e incomprensible para otro.

Explicación

El objetivo de esta guía es la creación de nombres que sean comprensibles en todos los entes gubernamentales.

e) Guía

Todos los nombres de los espacios de nombres **DEBEN** usar la convención *UCC*, la cual indica que los nombres **DEBEN** tener la primera letra de cada palabra en Mayúscula. Si el nombre lleva más de una palabra, **NO DEBEN** usarse guiones ni ningún otro tipo de separador. Si el nombre incluye un acrónimo (por ejemplo GELXML), la siguiente palabra empieza en minúscula.

Explicación

Esta es una convención que provee consistencia.

[21] Sigla para *Lower Camel Case* que significa que la primera letra de cada palabra va en minúscula y las primeras letras de las demás palabras van en mayúscula.

Ejemplo

republicaDeColombia, ministerioDeIndustraYComercio

f) Guía

Los valores descriptivos de una enumeración **DEBERÀN** estar en letras mayúsculas. Si la descripción tiene más de una palabra, estas se separan mediante un espacio o un guión bajo.

Explicación

Esta es una regla que provee mejor legibilidad.

g) Guía

Los elementos **DEBERÁN** usar como parte de su nombre el *metadato* de Identificador.

Explicación

Esta regla permite referenciar el nombre del elemento con el nombre del Identificador.

7.2 USO DE HERENCIA (EXTENSIÓN Y RESTRICCIÓN)

a) Guía

Si un componente preexistente no cumple los requerimientos completamente, **PUEDE** usarse el mecanismo de herencia para definir un nuevo tipo de dato basado en uno preexistente.

En ciertos casos, es posible que un tipo de dato enumerado liste todos los valores permitidos de forma que la lista deseada sea solo un subconjunto de estos. Si este es el caso, **DEBERÍA** utilizarse la herencia para limitar los valores aceptables. En otras palabras, es importante asegurar que el nuevo tipo enumerado es consistente y mantiene los estándares del que se ha definido previamente.

Explicación

Los mecanismos de herencia permiten derivar nuevos tipos de datos de manera sencilla para los usuarios al tiempo que pueden ser utilizados por diferentes herramientas para la generación de árboles de dependencias. Es necesario utilizar

estos mecanismos cuidadosamente para preservar el significado correcto de los datos.
Existen cuatro variantes de la herencia:

- Restricción de un tipo simple (utilizando *restriction*).
- Extensión de un tipo simple (utilizando *extension*).
- Restricción de un tipo complejo (utilizando *restriction*).
- Extensión de un tipo complejo (utilizando *extension*).

Es importante destacar que cuando se restringe un tipo complejo, la sintaxis de los *esquemas* exige que se incluya toda la definición original. Esto puede hacer difícil identificar los cambios requeridos en los tipos derivados si ocurren cambios en el tipo base. Por esta razón, la restricción de tipos complejos **DEBE** ser utilizada cuidadosamente. Además, el uso de los mecanismos de herencia puede estar restringido por la plataforma informática de implementación, por lo que se recomienda analizar la plataforma previa a la implementación de los mismos.

Ejemplo

```
<xsd:complexType name="tipoHitoPago">
        <xsd:complexContent>
                <xsd:extension base="seccor:tipoNotaEstructurada">
                        <xsd:sequence>
                                <xsd:element type="fechaPago"
                                type="comtem:tipoFecha"/>
                                …
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

7.3 USO DE LOS ATRIBUTOS *DEFAULT* (POR DEFECTO) Y *FIXED* (FIJO)

a) Guía

El atributo *default* **NO DEBE** usarse para especificar información importante en los componentes.

El atributo *fixed* **NO DEBE** usarse para especificar información importante en los componentes.

El único caso en el que un atributo **PUEDE** ser utilizado es cuando va en conjunto con el atributo *"required".*

Explicación

Estos dos mecanismos pueden ser muy útiles, porque permiten que los procesadores compatibles con los *esquemas* puedan generar información a la instancia basándose en el *esquema*. Por ejemplo, se puede hacer una definición tal que el procesador agregue automáticamente la información de versión a la instancia.

Sin embargo las desventajas son:

- Puede que ciertos procesadores no agreguen la información.
- Para una persona que está observando una instancia puede no ser evidente que debe referirse al *esquema* para ver toda la información.
- Una instancia archivada necesita el *esquema* para completar la información.

Al final, las desventajas de usar este mecanismo pesan más que los beneficios.

Ejemplo

(Nótese la complejidad de la definición de un atributo para un elemento):

```
<xsd:simpleContent>
        <xsd:extension base="xsd: base64Binary">
                <xsd:attribute name="pdidoc" type="xsd:nonNegativeInteger"
                use="required" default="1"/>
        </xsd:extension>
</xsd:simpleContent>
```

7.4 CONTENIDO DE LOS ELEMENTOS

a) Guía

Si un componente es definido como opcional **NO DEBE** permitirse su ocurrencia vacía. En este caso el componente no representa condiciones alternas según lo presentado en el numeral 6.5.

Explicación

En el caso de los componentes opcionales, la falta de datos **PUEDE** representarse mediante la ausencia del mismo. De esta manera se mantienen limpias las instancias.

Ejemplo

Supongamos que tenemos un elemento que representa a un ciudadano y que a su vez ésta tiene un elemento para indicar el teléfono de ubicación, lo cual **PUEDE**

ser opcional (no quiere decir que en el ejemplo el ciudadano no tenga teléfono sino que el elemento, al ser opcional, no se utilizó):

Un estilo válido para representar en XML; pero no para representar en el lenguaje común de intercambio de información, el hecho de que el ciudadano NO tenga un teléfono de ubicación, es enviar el elemento vacío.

```
<Ciudadano>
        … (otros datos)
        <telefono/>
</Ciudadano>
```

Con el objetivo de mantener limpias las instancias, en los *esquemas* del estándar se omite el elemento:

```
<Ciudadano>
        … (Otros datos)
</Ciudadano>
```

b) Guía

Si un elemento es definido como requerido **NO DEBE** permitirse su ocurrencia vacía. A diferencia de la regla anterior el elemento es requerido y **DEBERÁ** tener un valor.

Explicación

Si las reglas de negocio dictan que un elemento es requerido, normalmente también dictarán que contenga datos. En este caso, el *esquema* **DEBE** reflejar este hecho.

Ejemplo

Supongamos que tenemos un elemento que representa a un ciudadano y que a su vez ésta tiene un elemento para indicar el número de identificación. La representación de un estilo inválido sería la siguiente:

```
<Ciudadano>
        … (otros datos)
        <NumIdentificacion/>
</Ciudadano>
```

La representación de un estilo válido sería la siguiente:

```
<Ciudadano>
        … (otros datos)
        <NumIdentificacion>
                52700141
        </NumIdentificacion>
</Ciudadano>
```

7.5 ATRIBUTOS LOCALES Y GLOBALES

a) Guía

Los atributos **DEBEN** siempre ser definidos con un alcance local, definiéndolos dentro del contexto del elemento que los contiene.

Explicación

De esta forma se mantendrá la fácil comprensión del *esquema.*

7.6 TEXTO VS. CÓDIGOS

a) Guía

NO DEBEN utilizarse códigos sin proveer alguna forma de identificar el significado del código. Esto **PUEDE** conseguirse incluyendo tanto el código como el texto o proveyendo una referencia a otro documento que permita realizar la asociación. De esta manera se implementa el metadato *correlación* del elemento de dato indicándolo de una manera informativa.

Explicación

Esta guía ayuda a la mejor comprensión de las instancias.

```
<xsd:simpleType name="enumCodPaisAlf2">
  <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="AF">
                <xsd:annotation>
                        <xsd:documentation>AFGANISTAN</xsd:documentation>
                </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="AL">
```

```
                <xsd:annotation>
                        <xsd:documentation>ALBANIA</xsd:documentation>
                </xsd:annotation>
        </xsd:enumeration>
  …
  </xsd:restriction>
</xsd:simpleType>
```

7.7 CONTENIDO MIXTO EN LOS ELEMENTOS

a) Guía

El contenido de un elemento normalmente caerá en una de estas dos categorías: texto libre, como por ejemplo un campo de comentarios, o datos, por ejemplo un RUT.

En el caso de que el contenido sea datos, **DEBERÍA** evitarse el modelo mixto, es decir aquel que incluye tanto contenido como otros elementos. Para comentarios se utilizará el elemento *xsd:documentation.*

Explicación

Usualmente un elemento será identificado como datos o como texto, pero no ambas al mismo tiempo. Si se presentara el caso (por ejemplo, codificación XHTML), **DEBE** evaluarse cuidadosamente la opción de incluir los elementos en el texto.

Si el elemento de dato, **DEBE** poder extraerse la información de la manera más sencilla. Esta guía persigue ese objetivo.

Ejemplo

En este ejemplo, el elemento contiene texto y XHTML, la estructura es aceptable:

```
<xsd:documentation xml:lang=”es”>
  Este es un <Emphasize>comentario</Emphasize>
</xsd:documentation>
```

Esta estructura no es aceptable, porque el elemento contiene tanto texto como otros elementos:

```
< xsd:documentation >
  Este es un comentario
  <DatosAdicionales>
```

```
      Y unos datos adicionales
   </DatosAdicionales>
</xsd:documentation>
```

b) Guía

En el caso de que el contenido no sea datos, **DEBERÁN** utilizarse elementos *Dublin Core* para agregar semántica al contenido mixto incluido.

Explicación

Esta práctica permitiría l uso semántico de los elementos y datos.

7.8 USO DE [CDATA]

a) Guía

El uso de [CDATA] es aceptable en los siguientes casos: Cuando se incluye un XML dentro de otro, inclusión de Imágenes, inclusión de Sonidos o cualquier tipo de datos binarios codificados en base 64, en otros casos **DEBERÍA** evitarse.

Explicación

La estructura CDATA permite agregar datos sin estructura a una instancia. En el lenguaje común de intercambio de información se busca preservar siempre estructuras coherentes, por lo que [CDATA] permite la existencia de estructuras que no son validadas, son inconvenientes. Un caso en que **PUEDE** ser necesario utilizar [CDATA], es cuando se necesita incluir toda una instancia de un XML dentro de otro, en este caso el segundo al hacer la validación del documento XML ocurriría un error (<?xml…), por lo que la única solución viable es utilizar un [CDATA] para encapsular el segundo XML. De presentarse este caso es importante que el XML encapsulado, corresponda con las especificaciones del estándar, aunque el procesador de XML no podrá validarlo. Otro caso donde puede ser aceptable un [CDATA] es para el almacenamiento de datos binarios como, por ejemplo, imágenes.

Ejemplo

```
<Documento>
        <![CDATA[example.xml: -<?xml version="1.0"?><!DOCTYPE root [ - <! ...]]>
</Documento>
```

7.9 USO DE <ANY>, <ANYATTRIBUTE>

a) Guía

El uso de <*any*> y <*anyattribute*> no es **RECOMENDADO.**

Explicación

Los elementos <*any*> y <*anyattribute*> permiten expandir las instancias de manera no prevista en los *esquemas*. En el estándar, se persigue que toda instancia responda a un esquema.

7.10 ALMACENAMIENTO DE LOS ARCHIVOS

a) Guía

Los *esquemas* **DEBERÁN** ser almacenados en subdirectorios apropiados para agrupar *esquemas* relacionados. Los *esquemas* **DEBERÁN** ser almacenados en texto, codificado en *Unicode*.

Explicación

Los *esquemas* del estándar **DEBEN** estar disponibles, en una ubicación de acceso público para que cualquier ente interesado en conocerlos pueda acceder. Para lograr que los *esquemas* puedan ser indexados correctamente y los metadatos sean de utilidad, estos **DEBERÍAN** estar almacenados en texto, codificado en *Unicode* para permitir que el conjunto de caracteres del alfabeto Español como vocales tildadas, ñ entre otras sean interpretados correctamente, y en archivos .zip, no en .pdf o cualquier otro formato que no sea indexable.

7.11 ATRIBUTO TARGETNAMESPACE

a) Guía

Los *esquemas* del estándar **DEBEN** tener un *targetNamespace*. Este *espacio de nombres*, **DEBE** estar ubicado bajo el URL
http://www.gobiernoenlinea.gov.co/GEL-XML/schemas

Explicación

Esta guía provee un punto de partida para la organización de los *espacio de nombres* en el estándar.

7.12 USO DE PATRONES DE VALIDACIÓN DE DATOS EN LOS ESQUEMAS

a) Guía

DEBEN usarse patrones en el lenguaje común de intercambio de información para asegurar la integridad de los datos. Sin embargo, **DEBERÍA** evitarse el uso de patrones que resulten difíciles de cumplir o no sean claros. En este caso, se **DEBE** evaluar la posibilidad de usar un tipo complejo.

Explicación

Los patrones permiten validar de manera más efectiva los datos. Sin embargo, **DEBE** considerarse la legibilidad de las instancias resultantes, así como también la facilidad de análisis y verificación de las instancias.

Ejemplo

Este es un mal uso de un patrón:

```
<xsd:simpleType name=”tipoNomenclaturaDomiciliariaUrbana”>
        <xsd:restriction base=”xs:string”>
                <xsd:pattern value=”(Calle.+ [0-9]+ Carrera + [0-9]+)| Carrera.+ [0-9]+ Calle + [0-9]+ Numero [0-9] [0-9] ”/>
        </xsd:restriction>
</xsd:simpleType>
```

Este patrón limita lo que se puede escribir en un campo de texto a algo como **“Calle 7 Carrera 100”,** forzando los espacios entre las palabras. Este tipo de patrón no es claro para quien necesita generar la instancia. En este caso, sería mejor manejar algo como:

```
<xsd:simpleType name=tipoNomenclaturaDomiciliariaUrbana">
        <xsd:restriction base="xsd:string">
                <xsd:minLength value="10"/>
                <xsd:maxLength value="200"/>
                <xsd:pattern value="[a-zA-Z0-9áéíóúÁÉÍÓÚñÑüÜ\s]"/>
        </xsd:restriction>
</xsd:simpleType>
```

Un ejemplo de patrón bien utilizado:

```
<xsd:simpleType name="tipoNumCedulaExtranjero">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[a-zA-Z0-9áéíóúÁÉÍÓÚñÑüÜ]+"/>
        </xsd:restriction>
</xsd:simpleType>
```

7.13 CONSTRUCCIÓN DE ESQUEMAS PARA ELEMENTOS DE DATO TIPO GRUPO

a) Guía

Los *esquemas* para los elementos de dato de tipo grupo **NO DEBERÀN** usar la etiqueta *group*.

Explicación

La utilización de ésta etiqueta presenta las siguientes restricciones:

- Un esquema de tipo compuesto (complextType) **NO PUEDE** invocar más de un elemento de tipo *group.*
- Esta etiqueta no está implementada en la mayoría de librerías que generan clases a partir de los esquemas.

Ejemplo

Un ejemplo de cómo se DEBE implementar un esquema para un elemento de dato tipo grupo es el que se describe a continuación:

```
<xsd:complexType name="grupoDatoSolicitudAutorizacion">
…..
  <xsd:choice>
    <xsd:element name="informacionPoseeRegistroSanitario"
                 type="invreg:tipoDatoPoseeRegistroSanitario"/>
    <xsd:element name="informacionNoPoseeRegistroSanitario"
                 type="invreg:tipoDatoNoPoseeRegistroSanitario"/>
  </xsd:choice>
</xsd:complexType>
```

8 LAS BIBLIOTECAS COMUNES

Las bibliotecas comunes son el conjunto de componentes del *esquema* que son usados en dos o más instancias. La colocación de componentes compartidos en una biblioteca común aumenta la interoperabilidad y simplifica el mantenimiento de los *esquemas*. Sin embargo, esto también puede causar algunas complejidades adicionales.

8.1 PROBLEMAS EN EL DISEÑO DE BIBLIOTECAS COMUNES

Los problemas que normalmente se presentan en el diseño de bibliotecas comunes de XML, son la sobre inclusión y la carencia de un ciclo de vida separado.

8.1.1 Sobre inclusión

a) Guía

Considerando que son varios los componentes que residen en cada *esquema*, cuando se incluye un *esquema* dentro de otro, muchos elementos son incluidos innecesariamente. Una instancia que tiene que usar solamente un componente de *esquema* **PUEDE** incluir el *esquema* entero.

Ejemplo

Como un ejemplo usual de la complejidad de las bibliotecas comunes, considere el documento de *esquema* **tipoHombresFamilia.xsd**. Este incluye **tipoFamiliaPaterna.xsd** para tener acceso a algunos componentes compartidos que se relacionan con los **Primos**. **tipoFamiliaPaterna.xsd**, a su vez, incluye **tipoBienes.xsd** únicamente porque tiene que usar un tipo de código de lista llamado **tipoTipoBienes.xsd**. Sin embargo, en el proceso, se incluyen otros componentes innecesarios, que a su vez hacen referencia a otros componentes los cuales son incluidos también. El resultado es que todos los componentes de **tipoFamiliaPaterna.xsd** son incluidos en la estructura del *esquema* **tipoHombresFamilia.xsd** aun cuando ellos no sean necesarios en aquella instancia.

Esta situación es problemática por varios motivos:

- Entre más dependencias innecesarias sean incluidas, más cambios innecesarios afectan los *esquemas*. Por ejemplo, un cambio en tipoBienes.xsd afecta el grupo de registros del *esquema*, aun cuando no debiera.

- Es innecesariamente complejo para los usuarios porque tienen que trabajar con los *esquemas* suplementarios que son irrelevantes a sus Procesos de Negocio.

De vez en cuando surgen los problemas circulares, donde un componente en un documento del *esquema* es dependiente de un componente en otro documento de *esquema*, y viceversa. Las especificaciones de los *esquemas* del W3C XML no permiten la inclusión circular.

8.1.2 Carencia de un ciclo de vida separado

a) Guía

Los componentes en el diseño de la biblioteca común no tienen ningún *espacio de nombres* separado. Ellos toman el *espacio de nombres* de los *esquemas* utilizados en los documentos. Esto significa que un cambio realizado a la biblioteca común no afecta solamente la biblioteca común, sino también al resto de los documentos del esquema.

Si los componentes comunes estuvieran en *espacio de nombres* separado, podrían ser versionados separadamente. Por ejemplo, la versión 2 del esquema del grupo de registros podría usar la versión 2 de la biblioteca común. Más tarde, una nueva versión 2.1 de la biblioteca común podría ser creada. Sin embargo, la versión 2.2 y hasta la versión 2.3 del grupo de registros **PUEDE** seguir usando la versión 2.0 de la biblioteca común a no ser que tenga que actualizarse a la biblioteca común más nueva para aprovechar nuevas características.

En el escenario actual, sin *espacio de nombres* separados o versionados, los *esquemas* del grupo son obligados en conjunto a actualizarse con la biblioteca común. Esto lo hace mucho más vulnerable a los cambios de la biblioteca común. Se **DEBEN** seguir las siguientes políticas debido a la carencia de ciclo de vida separado:

- **Estructuración de los documentos del *esquema*:** Separación de los documentos del *esquema* en unidades más pequeñas para aliviar el problema de sobre inclusión.

- ***Espacios de nombres* y versionamiento:** Creación de uno o varios *espacios de nombres* separados, que alivian la carencia de un ciclo de vida separado.

- **Gestión de configuración:** Los *esquemas* **DEBEN** estar almacenados y publicados en archivos de texto. Para la publicación el URI[22] del *espacio de nombres* del *esquema* **DEBERÍA** ser el mismo en donde se encuentra publicado. .

[22] URI es el acrónimo del término ingles ***U**niform **R**esource **I**dentifier*, que significa identificador uniforme de recurso, definido en RFC 2396 (*Uniform Resource Identifiers: Generic Syntax*). Un ejemplo típico de URI es una dirección de una página Web.

8.2 ESTRUCTURACIÓN DE LOS ESQUEMAS

La estructuración de los *esquemas* implica decisiones tales como que tan grande debería ser cada documento, y cuales componentes deberán ser incluidos juntos en un solo *esquema*.

Para estructurar los *esquemas* se aconseja utilizar el sentido común. Un diseñador de *esquemas* **PUEDE** determinar cuántos y cuáles componentes **DEBERÍA** colocar en *esquemas* separados, basados en un juego de directrices.

a) Guía

El juicio humano **DEBERÍA** ser usado en la determinación de cuales componentes deberán estar definidos dentro de un mismo *esquema*. Las directrices descritas en la siguiente sección pueden ser usadas para hacer esta determinación.

Explicación

Para estructurar de manera apropiada los *esquemas* se requiere analizar el uso actual, así como las predicciones de uso futuro. Los siguientes factores **DEBERÍAN** ser tomados en cuenta para determinar la ubicación de un componente dentro de un *esquema*

b) Guía

Se **DEBERÁ** evitar el uso de inclusiones a más de tres niveles.

Explicación

Esta regla permite la mejor compresión del *esquema* y evita el uso de referencias circulares

8.3 COMPARTIR COMPONENTES PADRE

a) Guía

¿Cuántos componentes dependen de este componente? Si sólo es usado en otro tipo complejo, **PODRÍA** ser incluido en el mismo *esquema* del documento padre. Pero si es usado por diferentes tipos complejos, **PODRÍA** ser separado de ellos.

Ejemplo

Supóngase que ***primerApellido*** sólo es usado una vez, en ***nombrePersona***. Por lo tanto, **PUEDE** ser colocado en el mismo documento de *esquema* que ***nombrePersona***.

Supóngase que ***tipoDocumentoIdentificacion*** es usado en otros dos *esquemas* por separado: ***tipoSolicitanteRegistro***, y ***tipoSolicitenteCertificadoJudicial***. Por lo tanto se **DEBERÁ** crear el *esquema* ***tipoDocumentoIdentificacion*** por separado, para que pueda ser reutilizado.

8.4 USO COMÚN CON OTROS COMPONENTES

a) Guía

Cuando es usado este componente, ¿Existen allí otros componentes que son usados siempre con ellos?, si es así, **PUEDEN** ser colocados en el mismo *esquema* con los otros componentes. De otra manera, **DEBERÍA** ser separado de ellos.

Ejemplo

Si ***tipoTipoDocumentoIdentificacion*** y ***tipoTipoUnidadMedida*** nunca son usados juntos, entonces ellos no **DEBERÁN** aparecer en el mismo esquema.

Supóngase que ***tipoSegundoApellido*** y ***tipoPrimerApellido*** sólo son usados juntos, entonces ellos **PUEDEN** aparecer en el mismo *esquema*.
tipoTipoDocumentoIdentificacion y ***tipoLugar*** son usados (en ***tipoSolicitenteCertificadoJudicial***. y ***tipoSolicitanteRegistro***), pero ellos también son usados separadamente varias veces, entonces ellos **DEBERÁN** estar en *esquemas* separados.

8.5 PROBABILIDAD DE CAMBIO

a) Guía

Algunos componentes tienen mayor probabilidad de cambiar que otros. Específicamente, los tipos códigos de listas tienden a cambiarse con frecuencia y a depender del contexto. Por esta razón, los códigos de listas **DEBERÁN** ser separados de los tipos complejos que validan la estructura del documento.

Ejemplo

enumTipoTramite, un tipo de enumeración de trámites, **DEBERÍA** ser separado de los tipos estructurales que lo refieren, como ***tipoSolicitenteCertificadoJudicial.***

8.6 ÁMBITO / INSTANCIA

a) Guía

Los componentes que son relevantes en ámbitos diferentes **NO DEBERÁN** ser agrupados juntos, independientemente de las reglas establecidas.

Ejemplo

tipoDocumentoIdentidad y **tipoAfiliacionPOS DEBERÍAN** estar en *Esquemas* separados. Aunque ambos sean relacionados con la identificación, son conceptos separados.

Los tipos que se relacionan con descripciones (como **tipoDescripcion**) **NO DEBERÍAN** estar en el mismo *esquema* que los tipos que se relacionan con medidas (como **tipoMedida**).

8.7 ELEMENTOS LOCALES CON PREFIJOS

a) Guía

Los componentes definidos en las bibliotecas comunes **DEBEN** especificar *elementFormDefault = "qualified".*

Explicación

Tener *espacio de nombres* establecidos en las bibliotecas comunes aunque complica las instancias, evita el uso de referencia circulares o que un elemento igual pertenezca a varios *espacio de nombres*.

8.8 ESPACIOS DE NOMBRES Y VERSIONAMIENTO

a) Guía

Los *esquemas* de las bibliotecas comunes seguirán la misma estrategia de versionamiento que todos los otros *esquemas.*

Explicación

Esto significa que ellos tendrán *espacio de nombres* que incluyen la versión mayor, pero no la menor, y que los cambios sólo serán compatibles hacia atrás en versiones menores. Como otros *esquemas*, usarán el atributo de versión sobre el elemento xsd:*schema* para indicar el número de la versión menor.
Los *esquemas* de las bibliotecas comunes normalmente tendrán *espacio de nombres* separados de las instancias que los usan, de modo que ellos puedan ser versionados separadamente. Esto aliviará las cuestiones relacionadas con la carencia del ciclo de vida separado.

b) Guía

Los archivos individuales que constituyen la biblioteca común **PUEDEN** tener versiones menores, con cambios compatibles hacia atrás. Sin embargo, cuando la

biblioteca común cambia a una versión mayor, TODAS las instancias que usan la biblioteca **DEBEN** ser actualizadas.

Explicación

Esta guía permite asegurar que todas las instancias que manejan la misma biblioteca común sean válidas frente a la última versión.

8.9 DEFINICIONES COMUNES Y ESPACIOS DE NOMBRES

a) Guía

Toda definición en el lenguaje común de intercambio de información **DEBE** estar asociada a un *espacio de nombres* apropiado. En el caso de un esquema basado en bibliotecas comunes, **DEBE** utilizarse el mecanismo *xsd:import* para reutilizarlo. Todo *esquema* basado en bibliotecas comunes que sea definido para un propósito de negocio específico, **DEBE** tener su propio *targetNamespace*. Cuando se definan componentes genéricos, estos **PUEDEN** ser definidos en un *esquema* sin *espacio de nombres*. En este caso, **DEBE** utilizarse el mecanismo *xsd:include*. De esta forma, los elementos definidos adquieren el *espacio de nombres* del *esquema* que los incluye.

Explicación

El uso de *esquemas* basado en bibliotecas comunes sin *espacio de nombres* (conocidos como *esquemas* camaleón) simplifica el uso de los mismos en instancias de documento. Sin embargo, aquellos componentes que sean de uso específico gubernamental **DEBEN** estar enmarcados en un *espacio de nombres.*

Ejemplos

Los componentes genéricos son aquellos que no son de uso específico por el Gobierno Colombiano, por ejemplo, el RUT es un componente específico, mientras que una dirección de correo electrónico no lo es.

8.10 LOS ESPACIOS DE NOMBRES

a) Guía

Aquellas bibliotecas comunes que contengan al menos un componente específico tendrán un *espacio de nombres* (*namespaces*) de la forma:

http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/[capa]/(contexto)

Explicación

Los componentes específicos pertenecen a un contexto de uso gubernamental particular, lo que hace necesario enmarcarlos en un *espacio de nombres* que represente este hecho.

Ejemplo

Una biblioteca común de la Cancillería podría tener el *espacio de nombres*:

http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Proyectos/Cancilleria/Apostilla

Donde “Apostilla” representa una manera de organizar los documentos dentro del contexto de la Cancillería. Esta organización **DEBERÁ** ser coherente con la ubicación del elemento de dato según lo definido por la arquitectura de datos.

9 GUÍAS PARA USO DE METADATOS

Este capítulo provee las directrices sobre las especificaciones, los requerimientos y convenciones relacionados con los componentes de los *esquemas* que se van a utilizar en el estándar.

9.1 ESQUEMAS Y METADATOS

a) Guía

Todos los *esquemas* en el estándar **DEBEN** contener metadatos. Los documentos específicos **PUEDEN** extender el estándar con definiciones locales. Los metadatos de los *esquemas* **DEBEN** ser los mismos que los de la definición del elemento de dato y **DEBERÁN** utilizar los elementos *Dublin Core* (DC). Si no existen metadatos que correspondan en DC se **DEBERÁN** utilizar metadatos definidos en la capa Uso Proyectos Plataforma de Interoperabilidad, proyecto GEL-XML, módulo de metadatos. Los metadatos que **DEBEN** ser incluidos en los *esquemas* son: Nombre (Elemento de dato), Identificador, Autor(es)/Creador(es), Fecha de Incorporación al estándar, Descripción, Versión y Entidad u organización de contacto. Siempre que se use el metadato *descrption* de DC en los *esquemas* se **DEBERÁ** incluir el atributo xml:lang y su valor **DEBERÁ** ser el idioma con el que la descripción fue escrita.

Explicación

Es importante la inclusión de los metadatos correspondiente en los *esquemas*, para brindar más información al usuario del esquema sobre el elemento de dato.

Ejemplos

Ejemplos de inclusión de metadatos en *esquemas* se presentan a continuación:

Ejemplo de documentación completa:

```
<xsd:annotation>
        <xsd:appinfo>
                <pdigel:elementoDato
                xsi:schemaLocation="http://www.gobiernoenlinea.gov.co/GEL-
                XML/1.0/schemas/PDI/GEL-XML/Metadato tipoElementoDato.xsd"
                        xmlns:pdigel="http://www.gobiernoenlinea.gov.co/GEL-
                        XML/1.0/schemas/PDI/GEL-XML/Metadato"
```

```
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
                    Lugar Expedición
                </pdigel:elementoDato>
                <dc:identifier>http://www.gobiernoenlinea.gov.co/GEL-
                    XML/1.0/schemas/Local/Documental/tipoLugarExpedicion
                </dc:identifier>
                <dc:creator xml:lang="es">
                    Departamento Administrativo Nacional de Estadística.
                </dc:creator>
                <dc:creator xml:lang="es">
                    Departamento Nacional de Planeación.
                </dc:creator>
                <dc:creator xml:lang="es">
                    Ministerio de tecnologías de la Información y las
Comunicaciones: Programa Agenda de Conectividad.
                </dc:creator>
                <dc:creator xml:lang="es">
                    Ministerio de Hacienda y Crédito Público.
                </dc:creator>
                <dc:creator xml:lang="es">Ministerio de la Protección Social:
Programa de                 Apoyo a la Reforma en Salud.
                </dc:creator>
                <dc:issued>2005-03-01</dc:issued>
                <dc:description xml:lang="es">Elemento  que hace referencia al
                    departamento y al municipio en donde se genera
formalmente un                 documento.
                 </dc:description>
                <dc:hasVersion>1.0</dc:hasVersion>
                <pdigel:entidadContacto
                xsi:schemaLocation="http://www.gobiernoenlinea.gov.co/GEL-
                    XML/1.0/schemas/PDI/GEL-XML/Metadato
                        tipoEntidadContactoElementoDato.xsd"

        xmlns:locorg="http://www.gobiernoenlinea.gov.co/GEL-
        XML/1.0/schemas/Local/Organizacion"
        xmlns:pdigel="http://www.gobiernoenlinea.gov.co/GEL-
        XML/1.0/schemas/PDI/GEL-XML/Metadato"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <pdigel:nombreEntidadContacto> Ministerio de tecnologías
de la Información y las Comunicaciones:                 Programa
Agenda de Conectividad
                    </pdigel:nombreEntidadContacto>
                    <pdigel:dependenciaEntidadContacto>
                        <locorg:codEmpleo>009</locorg:codEmpleo>
                        <xsd:annotation>
                            <xsd:documentation>DIRECTOR
ADMINISTARTIVO O FINANCIERO O TÉCNICO U OPERATIVO
```

```
                                    </xsd:documentation>
                                </xsd:annotation>
                            </pdigel:dependenciaEntidadContacto>
                            <pdigel:cargoEntidadContacto>
                                <locorg:codEmpleo>009</locorg:codEmpleo>
                                <xsd:annotation>
                                    <xsd:documentation>DIRECTOR
ADMINISTRATIVO O FINANCIERO O TÉCNICO U OPERATIVO
                                    </xsd:documentation>
                                </xsd:annotation>
                            </pdigel:cargoEntidadContacto>
                            <pdigel:correoEntidadContacto>
                                gelxml@gobiernoenlinea.gov.co
                            </pdigel:correoEntidadContacto>
                        </pdigel:entidadContacto>
                    </xsd:appinfo>
                </xsd:annotation>
```

Ejemplo de documentación de un *uso* en un esquema:

```
<xsd:element name="fechaSalidaPais" type="comtem:tipoFecha">
        <xsd:annotation>
            <xsd:appinfo>
              <dc:description xml:lang="es">Fecha en la que una Persona sale
del  País.
              </dc:description>
            </xsd:appinfo>
        </xsd:annotation>
</xsd:element>
```

9.2 VERSIONAMIENTO DE LOS ESQUEMAS MEDIANTE EL ATRIBUTO “VERSION”

a) Guía

De acuerdo con las recomendaciones de W3C los *esquemas* **DEBEN** contener un número de versión utilizando el atributo “*version*” del elemento “*schema*”. Dicho número de versión aplicará a todos los componentes definidos en dicho *esquema*. Los números de versión **DEBEN** seguir la siguiente convención: MM.NN donde:

- **MM** es el número de versión mayor.
- **NN** es el número de versión menor.

Explicación

Versionar los *esquemas* es una buena práctica que previene errores causados por el uso de *esquemas* incorrectos.

9.3 INDICANDO LA VERSIÓN DEL ESQUEMA EN LAS INSTANCIAS

a) Guía

Los *esquemas* **PUEDEN** requerir que las instancias incluyan la información de versionamiento del *esquema*. Esto **PUEDE** conseguirse mediante el siguiente método:

- Incluir en algún elemento (regularmente en el elemento raíz) Este atributo **PUEDE** ser *"fixed"* en conjunción con *"required".*

Explicación

Muchas instancias serán documentos permanentes, los cuales estarán vigentes aun cuando el *esquema* al que se adhieren ya no sea la versión corriente. Al incluir la versión del *esquema* en la instancia queda establecido el *esquema* particular que se utilizó para su generación.

Por el contrario, los mensajes XML no serán permanentes, en este caso existe la posibilidad de que alguna aplicación utilice una instancia generada a partir de un esquema obsoleto al intentar una comunicación. En este caso, la versión incluida en la instancia le permitirá a la aplicación receptora rechazar dichos mensajes.

9.4 EL ATRIBUTO ID EN EL ELEMENTO ESQUEMA

a) Guía

El elemento *esquema* **DEBE** siempre contener un atributo **id**, el cual **DEBE** coincidir con el *identificador* (metadato de la definición del elemento) del documento, omitiendo el URL.

Explicación

Es buena práctica tener un identificador para cada esquema y relacionar la definición del elemento de dato con el esquema.

10 PROCEDIMIENTO DE SIMPLIFICACIÓN DE LOS ESQUEMAS XSD DEL ESTÁNDAR

El objetivo de este capítulo es el de explicar un conjunto de pasos que permitirán al usuario técnico de los esquemas XSD del estándar, usar solamente el grupo de esquemas que realmente necesita en sus servicios de intercambio de información, lo anterior con el fin de disminuir los tiempos de validación de los esquemas, mejorando de esta manera los tiempos de respuestas de los servicios que los usan.

Para realizar este procedimiento se **DEBEN** tener en cuenta las siguientes precondiciones:

- Tener conocimiento profundo de la arquitectura del estándar.
- Disponer de la versión aprobada del catálogo.
- La persona que va a realizar este ajuste **DEBE** tener un conocimiento sobre la estructura del catálogo y el significado de cada uno de sus columnas.
- Disponer de la versión completa y oficial de los esquemas XSD del estándar, con los elementos creados y modificaciones realizadas.
- Verificar que la biblioteca común que pertenece al módulo del proyecto que se está desarrollando incluya todos los esquemas de entrada y salida.
- Verificar que la biblioteca común valide correctamente utilizando un validador XSD.
- Siempre se DEBE validar la biblioteca común del proyecto, cada vez que se realice un paso, para verificar los cambios no afecten la correcta validación de los esquemas de entrada y de salida.
- El análisis y ajuste de los esquemas se **DEBE** realizar por capas iniciando por la más especializada (Proyectos) hasta la más general (Común).
- La capa Tipos de datos GEL-XML no se **DEBE** ajustar.

A continuación se presentan los pasos a seguir:

10.1 ELIMINAR LA CAPA PDI Y LOS OTROS PROYECTOS

a) Guía

Identificar el nombre del proyecto y módulo que se utiliza en el catálogo, seguido a esto identificar la capa PDI y los proyectos diferentes al anteriormente identificado y proceder a eliminarlos.

Explicación

Eliminar esquemas XSD que no tienen incidencia en los servicios de intercambio de información.

10.2 ELIMINAR LA CAPAS QUE NO SE USAN

a) Guía

Se **DEBEN** identificar las capas que no son importadas por los esquemas XSD que implementan el(los) servicios de intercambio.

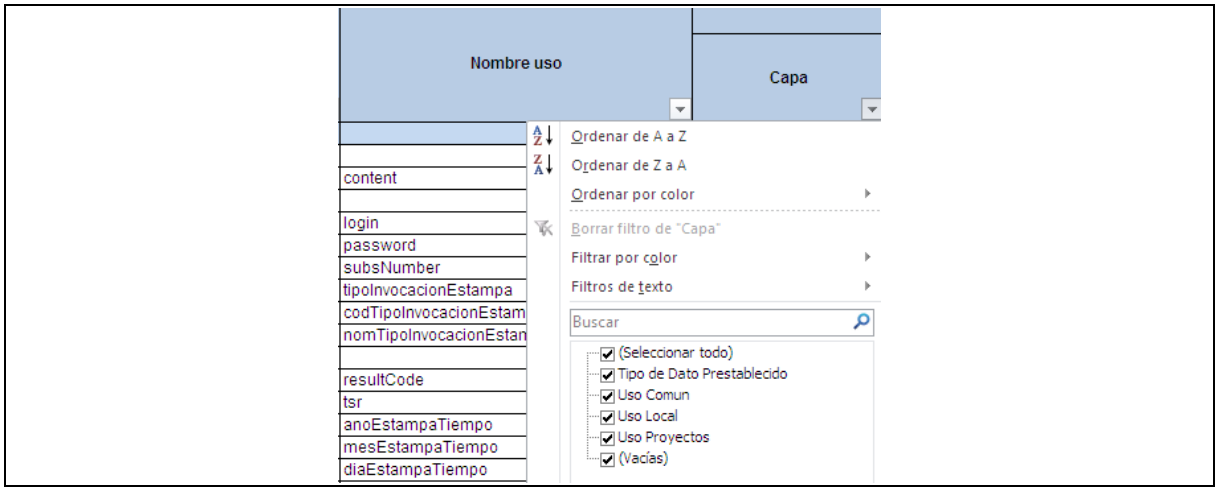
NOTA: En el catálogo, los elementos de tipo compuesto que se reutilizan o a los que se les realiza alguna acción diferente a creación no se les describe su formato, por tal razón también se **DEBEN** tener en cuenta las capas que utilizan los hijos, para evitar borrar una capa que indirectamente se esté importando.

Explicación

Eliminar esquemas XSD que no tienen incidencia en los servicios de intercambio de información.

Ejemplo

Para realizar ésta verificación se **DEBE** filtrar en el catálogo por capa y de ésta manera identificar las capas que no se utilizan.



Para este caso se identifica que las capas que no se utilizan y se **DEBEN** eliminar son: Uso Macrosector Económico, Uso Macrosector Social, Uso Macrosector Recursos Naturales y Medio Ambiente, Uso Información No Desglosable, Uso Internacional, seguido a esto se procede a eliminar las capas identificadas.

10.3 AJUSTAR LAS BIBLIOTECAS COMUNES QUE SE IMPORTAN

a) Guía

Se **DEBEN** incluir solamente en las bibliotecas comunes los elementos que se van a utilizar en el servicio de intercambio de información.
Al finalizar el ajuste de la biblioteca común siempre se **DEBERÁ** validar.
Este procedimiento se **DEBE** hacer para todas las áreas de la misma capa.

NOTA: En el catálogo, los elementos de tipo compuesto que se reutilizan o a los que se les realiza alguna acción diferente a creación no se les describe su formato, por tal razón, también se **DEBEN** tener en cuenta en la inclusión estos elementos hijos, para evitar borrar un esquema que indirectamente se esté utilizando.

Explicación

Minimizar el tiempo de carga de las bibliotecas comunes, cuando se importan desde otros esquemas.

Ejemplo

Para realizar éste ajuste se **DEBE** filtrar en el catálogo por capa y área, para de ésta manera identificar los elementos que se utilizan.
En este ejemplo se filtró por la capa Uso Común y el área Temporal.

Listado de Elementos	Ocurrencias	Nombre uso	Arquitectura				Tipo	NOMBRE Doc	Nombre XSD
			Capa	Área	Proyecto	Módulo			
Año	0..1	anoEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoAno	tipoAno.xsd
Mes	0..1	mesEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoMes	tipoMes.xsd
Día	0..1	diaEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoDia	tipoDia.xsd
Hora	0..1	horaEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoHora	tipoHora.xsd
Minuto	0..1	minutoEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoMinuto	tipoMinuto.xsd
Segundo	0..1	segundoEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoSegundo	tipoSegundo.xsd
Milisegundo	0..1	miliSegundoEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoMilisegundo	tipoMilisegundo.xsd
Microsegundo	0..1	microsegundoEstampaTiempo	Uso Comun	Temporal	N/A	N/A	Simple	tipoMicrosegundo	tipoMicrosegundo.xsd

Para este caso se identifica que los esquemas XSD que se solamente se **DEBEN** incluir en la biblioteca común Temporal.xsd que corresponde a la capa Uso Común y área Temporal son: tipoAno.xsd, tipoMes.xsd, tipoDia.xsd, tipoHora.xsd, tipoMinuto.xsd,
tipoSegundo.xsd, tipoMilisegundo.xsd, tipoMicrosegundo.xsd.

La biblioteca ajustada quedaría de la siguiente forma:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Temporal"
elementFormDefault="qualified" version="1.0">

        <xsd:include schemaLocation="tipoAno.xsd"/>
        < xsd:include schemaLocation ="tipoMes.xsd"/>
        < xsd:include schemaLocation ="tipoDia.xsd"/>
        < xsd:include schemaLocation ="tipoHora.xsd"/>
        < xsd:include schemaLocation ="tipoMinuto.xsd"/>
        < xsd:include schemaLocation ="tipoSegundo.xsd"/>
        < xsd:include schemaLocation ="tipoMilisegundo.xsd"/>
        < xsd:include schemaLocation ="tipoMicrosegundo.xsd"/>

</xsd:schema>
```

10.4 ELIMINAR LOS ESQUEMAS QUE NO SE UTILIZAN

a) Guía

Se **DEBEN** eliminar todos los esquemas que no se hayan incluido en la librería común o no se utilicen indirectamente.
Al finalizar la eliminación de los esquemas XSD, la biblioteca común siempre se **DEBERÁ** validar.
Este procedimiento se **DEBE** hacer para todas las áreas de la misma capa.
Si al finalizar la eliminación de los esquemas XSD, un área queda vacía o sólo con la biblioteca común, se **DEBERÁ** proceder a eliminar el área, debido a que no se está utilizando.

NOTA: En el catálogo, los elementos de tipo compuesto que se reutilizan o a los que se les realiza alguna acción diferente a creación no se les describe su formato, por tal razón también se **DEBEN** tener en cuenta estos elementos hijos, para evitar borrar un esquema que indirectamente se esté utilizando.

Explicación

Depurar el grupo de esquemas que implementan el servicio de intercambio de información.

11 ADAPTADORES

Un adaptador se implementa por medio de transformaciones en el lenguaje XSLT (XSL Transformations[23]). La transformación se realiza entre documentos XML de otros estándares y/o instancias de elementos de dato del lenguaje común de intercambio de información.

Esta sección pretende mostrar la forma de incluir adaptadores de elementos de dato del estándar del lenguaje común de intercambio de información, que no es de otra forma sino dentro de los esquemas de los elementos de dato. Para el entendimiento de esta sección, son necesarios conocimientos profundos en XML, XML Schemas, expresiones XPath[24] y programación XSLT.

El propósito de los adaptadores es ser incluidos/utilizados en documentos y/o programas de transformación específicos, debido a que un adaptador transforma un elemento y no un documento completo. Esto debido al infinito número de combinaciones de instancias de documentos que pueden existir en los estándares.

Los adaptadores en ningún momento son utilizados por los esquemas o dentro de los esquemas. Los adaptadores permiten transformar elementos-XML de instancias del lenguaje común de intercambio de información a otro estándar XML y viceversa.

Los datos necesarios para un adaptador son:

- **Elemento de Entrada:** Las instancias de los elementos de dato de entrada que son convertidos en el elemento de dato de salida. El elemento de entrada puede ser del estándar externo o del lenguaje común de intercambio de información.

- **Elemento de Salida:** La instancia del elemento de dato resultante, bien sea externa o del lenguaje común de intercambio de información.

- **Transformación:** La transformación necesaria para obtener el elemento de dato de salida. Esta **DEBE** ser un conjunto de sentencias XSLT. La transformación puede ser del estándar externo hacia el lenguaje común de intercambio de información o viceversa.

[23] XSL es el acrónimo de *Extensible Stylesheet Language* (Lenguaje Extensible para Hojas de Estilo), para mayor información visitar http://www.w3.org/TR/xsl
[24] XPath es el acrónimo de *XML Path Languaje,* es un lenguaje para seleccionar nodos en un documento XML. Para mayor información visitar http://www.w3.org/TR/xpath

A continuación se presentan las guías para la especificación de adaptadores:

11.1 ESPECIFICACIÓN DE ADAPTADORES

b) Guía

Los adaptadores **DEBEN** ser especificados con el lenguaje XSLT. Debido a que el adaptador está relacionado con el elemento de dato, la especificación del adaptador se anexa al esquema del elemento de dato.

Explicación

La especificación de transformaciones deber realizarse por medio de un lenguaje formal. En el caso de trasformaciones XML se realizará con el lenguaje XSLT.

c) Guía

Los elementos XML de un adaptador se **DEBEN** anexar a la documentación del esquema que representa el elemento de dato que es incorporado.

Los tipos de elementos de dato a utilizar dentro de la documentación se presentan en **negrita** a continuación, y la sangría representa la jerarquía:

- **Lista Adaptador**: Utilizado para agrupar un conjunto de adaptadores.
  - **Adaptador:** Elemento que define un adaptador, contiene los elementos XML de entrada, salida (*Elemento Transformación*) y la transformación (*Transformación*).
    - **Elemento Transformación:** Elemento de dato que indica la instancia del elemento de dato a utilizar en la transformación. A su vez contiene dos elementos de dato, uno para el nombre del elemento y otro para la versión del mismo. Son necesarios **dos** “Elemento Transformación”, uno para la entrada y otro para la salida.
    - **Transformación:** Elemento de dato que contiene la especificación de la conversión del elemento de dato de entrada en el elemento de dato de salida, en lenguaje XSLT.

Explicación

Se utiliza una lista de adaptadores porque se puede presentar una situación en la que un mismo elemento-XML de entrada generará varios elementos de salida. Por ejemplo, un elemento-XML que contenga la fecha y hora se puede convertir en dos elementos XML: uno que tenga la información de la hora y otro, con la fecha. El nombre del elemento XML de entrada/salida, al contener el espacio de nombres, permite identificar el estándar al cual pertenece el elemento-XML de entrada o salida.

d) Guía

Inclusión de los adaptadores en la documentación de los esquemas.

Explicación

Debido a que cada adaptador es propio de un esquema, se incluye el adaptador dentro de la etiqueta *annotation* ya que el adaptador como tal no especifica elementos de dato y es información agregada al esquema.

Ejemplo

A continuación se presenta la forma genérica de incluir los adaptadores dentro de un esquema XML (en la siguiente sección se muestra un ejemplo completo):

```
<xsd:annotation>
    <xsd:appinfo>
        <gelada:listaAdaptadores xmlns:gelada="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/PDI/GEL-XML/Adaptador"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/PDI/GEL-
XML/Adaptador C:\proyectos\schemas\PDI\GEL-
XML\Adaptador\tipoListaAdaptadorb.xsd">
   <gelada:adaptadorGELXML>

      <gelada:elementoTransformacionEntrada>
         <gelada:nomElementoTransformacionXML>
               [elemento de entrada con espacio de nombres]
         </gelada:nomElementoTransformacionXML>
         <gelada:versionElementoTransformacion>
               [versión del elemento de entrada]
         </gelada:versionElementoTransformacion>
      </gelada:elementoTransformacionEntrada>

      <gelada:elementoTransformacionSalida>
         <gelada:nomElementoTransformacionXML>
               [elemento de salida con espacio de nombres]
         </gelada:nomElementoTransformacionXML>
         <gelada:versionElementoTransformacion>
               [versión del elemento de salida]
         </gelada:versionElementoTransformacion>
      </gelada:elementoTransformacionSalida>

      <gelada:transformacionXML>
         [especificación del adaptador]
      </gelada:transformacionXML>

   </gelada:adaptadorGELXML>
        </gelada:listaAdaptadores>
```

```
    </xsd:appinfo>
  </xsd:annotation>
```

11.2 FACTIBILIDAD EN LA CREACIÓN DE ADAPTADORES

a) Guía

Es posible realizar una transformación y por ende crear un adaptador cuando se dispone de los datos necesarios para realizar dicha transformación, de lo contrario no es posible realizar la transformación. La transformación puede ser realizada de una instancia de un elemento de dato del lenguaje común de intercambio de información a un elemento instanciado del estándar externo o viceversa. Es posible que la transformación se pueda hacer en un solo sentido, es decir de una instancia de un elemento de dato del estándar a un elemento instanciado del estándar externo o de un elemento instanciado del estándar externo a una instancia de un elemento de dato del estándar. Cuando se puede hacer la transformación en un solo sentido se debe a que faltan datos en un estándar que el otro necesita. Sin embargo pueden existir ocasiones en donde la transformación se haga en los dos sentidos.

Explicación

En caso que se necesite realizar la transformación de un elemento de dato del lenguaje común de intercambio de información a uno externo, por ejemplo de un *nombre de una persona* en el estándar a un *nombre de una persona* en un lenguaje externo, los elementos de dato instanciados del lenguaje común de intercambio de información de *nombre* deben existir para realizar la transformación. Para el caso del estándar es posible ya que en otros países y a nivel internacional en general, se utiliza apenas el primer nombre (*first name*) y el apellido de familia (*last name*) como parte de la identificación de personas. En el lenguaje común de intercambio de información, se dispone del primer apellido y de primer nombre, por lo que se podría realizar la transformación. Adelante en el ejemplo, se nota que el segundo apellido y el segundo nombre no son transformados (o se pierden en la transformación) debido a que el estándar destino no los necesita, lo cual se ajusta a la realidad, ya que en algunas culturas diferentes a la Colombiana no se acostumbra utilizar el segundo apellido y el segundo nombre.

Ejemplo

El siguiente es un elemento instanciado del lenguaje común de intercambio de información, note que el estándar utiliza dos apellidos y dos nombres:

Con el siguiente esquema que utiliza el elemento de dato tipoNomPersona del estándar:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:comper="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Personal">
<xsd:import schemaLocation="tipoNomPersona.xsd"
namespace="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/Personal"
/>
    <xsd:element name="nombreGEL-XML" type="comper:tipoNomPersona"/>
</xsd:schema>
```

Se tiene la el siguiente de instancia del un documento XML (donde p1 es el prefijo del espacio de nombres identificado por la URI http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/Personal):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nombreGEL-XML xmlns:p1="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Personal"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Personal tipoNomPersonatest.xsd">
    <p1:primerApellido>Ariza</p1:primerApellido>
    <p1:segundoApellido>Ávila</p1:segundoApellido>
    <p1:primerNombre>Cesar</p1:primerNombre>
    <p1:segundoNombre>Elberto</p1:segundoNombre>
</nombreGEL-XML>
```

El siguiente es elemento instanciado de un estándar externo (se tomó un extracto del estándar VCard[25]). En el estándar VCard se utiliza solamente el apellido de familia (*last Name*) y el primer nombre (*first Name*), por lo que en una transformación del lenguaje común de intercambio de información a VCard se pierde un apellido y un nombre:

```
<?xml version="1.0"?>
 <rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:vCard = "http://www.w3.org/2001/vcard-rdf/3.0#" >
   <rdf:Description>
     <vCard:N rdf:parseType="Resource">
      <vCard:Family>César</vCard:Family>
```

[25] Tomado de http://www.w3.org/TR/vcard-rdf el 10 de octubre de 2008. Los elementos-XML utilizados en la especificación VCard se encuentran el apéndice B de este documento.

```
      <vCard:Given>Ariza</vCard:Given>
    </vCard:N>
   </rdf:Description>
 </rdf:RDF>
```

El programador de XSLT, después de crear el adaptador de acuerdo al elemento-XML de entrada del lenguaje común de intercambio de información y el elemento-XML de salida, incluirá el adaptador de la siguiente manera:

```
<xsd:annotation>
…
 <xsd:appinfo>
   <gelada:listaAdaptadores xmlns:gelada="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/PDI/GEL-XML/Adaptador" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/PDI/GEL-XML/Adaptador ..\..\..\..\schemas\PDI\GEL-XML\Adaptador\tipoListaAdaptadorb.xsd">
…
   <gelada:adaptadorGELXML>

        <gelada:elementoTransformacionEntrada>
            <gelada:nomElementoTransformacionXML>http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/Personal/tipoNomPersona</gelada:nomElementoTransformacionXML>
        <gelada:versionElementoTransformacion>1.0</gelada:versionElementoTransformacion>
        </gelada:elementoTransformacionEntrada>

        <gelada:elementoTransformacionSalida>
            <gelada:nomElementoTransformacionXML>http://www.w3.org/2001/vcard-rdf/3.0</gelada:nomElementoTransformacionXML>
        <gelada:versionElementoTransformacion>3.0</gelada:versionElementoTransformacion>
        </gelada:elementoTransformacionSalida>

        <gelada:transformacionXML><![CDATA[<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:comper="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/Personal"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#">
                    <xsl:template match="/">
                        <rdf:RDF>
                            <rdf:Description>
                                <vCard:N rdf:parseType="Resource">
                                        <xsl:apply-templates/>
                                </vCard:N>
                            </rdf:Description>
                        </rdf:RDF>
                    </xsl:template>
                    <xsl:template match="comper:primerApellido">
                        <vCard:Family>
                            <xsl:value-of select="."/>
                        </vCard:Family>
                    </xsl:template>
                    <xsl:template match="comper:primerNombre">
```

```
                        <vCardGiven>
                          <xsl:value-of select="."/>
                        </vCardGiven>
                      </xsl:template>
                      <xsl:template match="text()|@*" mode="#all"/>
                    </xsl:transform>]]></gelada:transformacionXML>
    </gelada:adaptadorGELXML>
  </gelada:listaAdaptadores>
 </xsd:appinfo>
</xsd:annotation>
```

En el caso de VCard, es posible hacer la transformación inversa (de VCard al lenguaje común de intercambio de información) debido a que el segundo apellido y el segundo nombre en el estándar son opcionales. El adaptador quedaría así:

```
<xsd:annotation>
…
 <xsd:appinfo>
        <gelada:listaAdaptadores xmlns:gelada="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/PDI/GEL-XML/Adaptador" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/PDI/GEL-XML/Adaptador ..\..\..\..\schemas\PDI\GEL-
XML\Adaptador\tipoListaAdaptadorb.xsd">
  <gelada:adaptadorGELXML>
      <gelada:elementoTransformacionEntrada>
          <gelada:nomElementoTransformacionXML>
              http://www.w3.org/2001/vcard-rdf/3.0
           </gelada:nomElementoTransformacionXML>
          <gelada:versionElementoTransformacion>3.0</gelada:versionElementoTransformacion>
      </gelada:elementoTransformacionEntrada>
      <gelada:elementoTransformacionSalida>
          <gelada:nomElementoTransformacionXML>http://www.gobiernoenlinea.gov.co/GEL-
XML/1.0/schemas/Comun/Personal/tipoNomPersona</gelada:nomElementoTransformacionXML>
  <gelada:versionElementoTransformacion>1.0</gelada:versionElementoTransformacion>
      </gelada:elementoTransformacionSalida>
      <gelada:transformacionXML><![CDATA[<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:comper="http://www.gobiernoenlinea.gov.co/GEL-XML/1.0/schemas/Comun/Personal"
xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#">
      <xsl:template match="/">
          <nombreGEL-XML>
              <xsl:apply-templates/>
          </nombreGEL-XML>
      </xsl:template>
      <xsl:template match="vCard:Family">
          <comper:primerApellido>
              <xsl:value-of select="."/>
          </comper:primerApellido>
      </xsl:template>
      <xsl:template match="vCard:Given">
          <comper:primerNombre>
```

```
            <xsl:value-of select="."/>
          </comper:primerNombre>
      </xsl:template>
      <xsl:template match="text()|@*" mode="#all"/>
    </xsl:transform>
       </ pdigel:transformacion>]]></gelada:transformacionXML>
  </gelada:adaptadorGELXML>
    </gelada:listaAdaptadores>
  </xsd:appinfo>
</xsd:annotation>
```

11.3 SOFTWARE PARA LA EJECUCIÓN DE TRANSFORMACIONES

a) Guía

Para la realización de transformaciones es necesario tener los datos de entrada (documento XML), la especificación de la transformación (documento XSLT) y un procesador para transformaciones (software para transformaciones).

Explicación

Para realizar una transformación es necesario invocar una pieza de software que interprete el adaptador y los datos de entrada que realice la transformación. Dicho software de transformación puede ser invocado por medio de una API de un lenguaje de alto nivel o por medio de líneas de comandos en sistemas operativos.

Ejemplos

Existen procesadores de código abierto para transformaciones como SAXON-B[26] con versiones para Java y .Net. Otro procesador libre es XALAN-J[27], que también provee una API para ser utilizado dentro de programas Java.

[26] SAXON-B acrónimo de *Simple API for XML ON* (Interface de programación para aplicaciones sencillas para XML) disponible en http://saxon.sourceforge.net/
[27] XALAN-J acrónimo de XALAN Java está disponible en http://xml.apache.org/xalan-j/

12 TRABAJO FUTURO

Para las futuras versiones de este documento se recomienda estudiar la inclusión de los siguientes aspectos:

- Utilizar la etiqueta *ref*, una vez sea soportada por las soluciones incluidas en la Arquitectura de Gobierno en Línea.
- Analizar la opción de crear un esquema independiente por cada uso de un elemento de dato, a través de etiquetas *ref*.

13 REFERENCIAS

nformación general sobre XML:

- http://www.w3.org/XML/ (accedido el 20 de abril de 2010)
- http://www.w3schools.com/xml/default.asp (accedido el 20 de abril de 2010)
- http://www.xmlfiles.com/xml/ (accedido el 20 de abril de 2010)

Información general sobre Esquemas:

- http://www.w3.org/XML/Schema (accedido el 20 de abril de 2010)
- http://www.xfront.com/GlobalVersusLocal.pdf (accedido el 20 de abril de 2010)
- http://www.w3schools.com/Schema/default.asp (accedido el 20 de abril de 2010)
- http://www.xml.com/pub/a/2000/11/29/schemas/part1.html (accedido el 20 de abril de 2010)

Información relacionada con las metodologías utilizadas

- http://www.govtalk.gov.uk/ (accedido el 20 de abril de 2010)
- http://www.dublincore.org/ (accedido el 20 de abril de 2010)

14 APÉNDICES

14.1 APÉNDICE A: PALABRAS CLAVES A UTILIZAR PARA INDICAR NIVELES DE REQUERIMIENTO (RFC 2119).

Palabras clave a utilizar para indicar niveles de requerimiento (RFC 2119). Versión original en inglés en http://www.faqs.org/rfcs/rfc2119.html.

Network Working Group
Request for Comments: 2119
BCP: 14
Categoría:

S.Bradner
Harvard University
Marzo 1997
Mejor Práctica Actual

ESTATUS DE ESTE MEMORÁNDUM:

Este documento especifica una Mejor Práctica Actual de Internet para la comunidad Internet, y solicita su discusión y sugerencias para posibles mejoras. La distribución de este memorándum es ilimitada.

RESUMEN:

En muchos documentos de seguimiento estándar se usan varias palabras para indicar los requerimientos de la especificación. Estas palabras a menudo están en mayúsculas. Este documento define cómo **DEBERÁN** ser interpretadas estas palabras en documentos IETF. Los autores que sigan estas instrucciones **DEBERÁN** incorporar esta frase cerca del principio de sus documentos:

Las palabras claves "**DEBE**", "**NO DEBE**", "**REQUERIDO**", "**OBLIGATORIO**", "**DEBERÁ**", "**NO DEBERÁ**", "**DEBERÍA**", "**NO DEBERÍA**", "**RECOMENDADO**", "**PUEDE**" y "**OPCIONAL**" en este documento serán interpretadas como se describe en RFC 2119.

Nótese que la contundencia de estas palabras está modificada por el nivel de requerimiento del documento en el que son usadas.

1. **DEBE**: Esta palabra, o los términos "**REQUERIDO**", "**OBLIGATORIO**" o "**DEBERÁ**", significa que la definición es un requerimiento insoslayable de la especificación.

2. **NO DEBE**: Esta frase, o la frase "**NO DEBERÁ**", significa que la definición es una prohibición insoslayable de la especificación.

3. **DEBERÁ**: Esta palabra, o el adjetivo "**RECOMENDADO**", significa que pueden existir razones válidas en determinadas circunstancias para ignorar un elemento determinado, pero que la totalidad de las consecuencias deben ser comprendidas y cuidadosamente sopesadas antes de elegir otros derroteros.

4. **NO DEBERÁ**: Esta frase, o la frase "**NO RECOMENDADO**", significa que pueden existir razones válidas en determinadas circunstancias en las que el comportamiento en particular sea útil o incluso aconsejable, pero que la totalidad de las consecuencias deben ser comprendidas y cuidadosamente sopesadas antes de implementar cualquier comportamiento descrito bajo esta etiqueta.

5. **PUEDE**: Esta palabra, o el adjetivo "**OPCIONAL**", significa que un elemento es realmente opcional. Un proveedor puede elegir incluir el elemento porque un mercado en particular lo necesite o porque el proveedor sienta que realza el producto aunque otro proveedor pueda omitir el mismo elemento. Una implementación que no incluya una opción determinada **DEBE** estar preparada para interoperar con otra implementación que incluya la opción, aunque quizá con reducida funcionalidad. En el mismo orden de cosas, una implementación que incluya una opción en particular **DEBE** estar preparada para interoperar con otra implementación que no incluya la opción (excepto, por supuesto, para la característica que aporte la opción).

6. Guía de uso de estos Imperativos: Los imperativos del tipo definido en este memorando deben ser usados con cuidado y con mesura. En particular, sólo **DEBEN** ser utilizados donde sea realmente necesario para la ínteroperación o para limitar un comportamiento potencialmente dañino (p.ej., limitando retransmisiones). Por ejemplo, no deben ser usados para intentar imponer un método concreto a los implementadores cuando el método no sea necesario para la interoperabilidad.

7. Consideraciones de seguridad: Estos términos se utilizan normalmente para especificar comportamientos con implicaciones de seguridad. Los efectos sobre la seguridad de no implementar un **DEBE** o **DEBERÍA**, o hacer algo que la especificación dice **NO DEBE** o **NO DEBERÍA** ser hecho, pueden ser muy sutiles. Los autores de documentos **DEBERÁN** tomarse su tiempo para elaborar las implicaciones de seguridad respecto a no seguir recomendaciones o requerimientos, ya que la mayoría de los implementadores no tienen el beneficio de la experiencia y de la discusión que produjo la especificación.

8. Agradecimientos: Las definiciones de estos términos son una amalgama de las definiciones tomadas de numerosos documentos RFC. Además, se han incorporado sugerencias de numerosas personas incluyendo a Obert Ullmann, Thomas Nartenm Neal McBurnett, y Robert Elz.

DIRECCIÓN DEL AUTOR:

Scott Bradner Harvard University 1350 Mass.
Ave. Cambridge, MA 02138
phone - +1 617 495 3864
email - sob@harvard.edu
Traducción: José M. Cainzos jmcainzos@airtel.net SEVILLA –España.

14.2 APÉNDICE B: VOCABULARIO VCARD EXPRESADO EN XML.

A continuación se presentan los elementos del estándar *VCard*[28] expresado en XML. *VCard* es un estándar para el intercambio electrónico de tarjetas de presentación.

```
<?xml version='1.0'?>
 <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#">

 <rdf:Description ID="FN">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>Formatted Name</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="NICKNAME">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>Nickname</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="BDAY">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>Birthday</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="MAILER">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>EMail Program</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="GEO">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>Geographical Information</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="TITLE">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>Position Title</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="ROLE">
   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:label>Position Role</rdfs:label>
 </rdf:Description>
```

[28] Tomado de http://www.w3.org/TR/vcard-rdf el 5 de octubre de 2008.

```
<rdf:Description ID="CATEGORIES">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Categories</rdfs:label>
</rdf:Description>

<rdf:Description ID="NAME">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>vCard Name</rdfs:label>
</rdf:Description>

<rdf:Description ID="SOURCE">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Source</rdfs:label>
</rdf:Description>

<rdf:Description ID="NOTE">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Notes</rdfs:label>
</rdf:Description>

<rdf:Description ID="PRODID">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Product ID</rdfs:label>
</rdf:Description>

<rdf:Description ID="REV">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Revision</rdfs:label>
</rdf:Description>

<rdf:Description ID="SORT-STRING">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Sort String</rdfs:label>
</rdf:Description>

<rdf:Description ID="CLASS">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Class</rdfs:label>
</rdf:Description>


<rdf:Description ID="TEL">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Telephone</rdfs:label>
  <rdfs:range rdf:resource="#TELTYPES"/>
</rdf:Description>
```

```
<rdfs:Class rdf:ID="TELTYPES"/>
 <TELTYPES rdf:ID="home"/> <TELTYPES rdf:ID="msg"/>
 <TELTYPES rdf:ID="work"/> <TELTYPES rdf:ID="fax"/>
 <TELTYPES rdf:ID="cell"/> <TELTYPES rdf:ID="video"/>
 <TELTYPES rdf:ID="pager"/> <TELTYPES rdf:ID="bbs"/>
 <TELTYPES rdf:ID="modem"/> <TELTYPES rdf:ID="car"/>
 <TELTYPES rdf:ID="isdn"/> <TELTYPES rdf:ID="pcs"/>

<rdf:Description ID="EMAIL">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Email Address</rdfs:label>
  <rdfs:range rdf:resource="#EMAILTYPES"/>
</rdf:Description>

<rdfs:Class rdf:ID="EMAILTYPES"/>
 <EMAILTYPES rdf:ID="internet"/>
 <EMAILTYPES rdf:ID="x400"/> <EMAILTYPES rdf:ID="pref"/>
<rdf:Description ID="ADR">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Address</rdfs:label>
  <rdfs:range rdf:resource="#ADRTYPES"/>
</rdf:Description>

<rdfs:Class rdf:ID="ADRTYPES"/>
 <ADRTYPES rdf:ID="dom"/> <ADRTYPES rdf:ID="intl"/>
 <ADRTYPES rdf:ID="postal"/> <ADRTYPES rdf:ID="parcel"/>
 <ADRTYPES rdf:ID="home"/> <ADRTYPES rdf:ID="work"/>
 <ADRTYPES rdf:ID="pref"/>
<rdfs:Class rdf:ID="ADRPROPERTIES">
   <rdfs:subClassOf rdf:resource="#ADRTYPES"/>
</rdfs:Class>

<rdf:Description ID="Pobox">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Post Office Box</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Extadd">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Extended Address</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Street">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Street</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
```

```
</rdf:Description>

<rdf:Description ID="Locality">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Locality/Suburb</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Region">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Region/State</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Pcode">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Postal Code</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Country">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Country</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ADRPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="LABEL">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Address Label</rdfs:label>
  <rdfs:range rdf:resource="#ADRTYPES"/>
</rdf:Description>

<rdf:Description ID="TZ">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Timezone</rdfs:label>
  <rdfs:range rdf:resource="#TZTYPES"/>
</rdf:Description>

<rdfs:Class rdf:ID="TZTYPES"/>
 <TZTYPES rdf:ID="text"/>

<rdfs:Class rdf:ID="NPROPERTIES"/>

<rdf:Description ID="N">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Name</rdfs:label>
  <rdfs:range rdf:resource="#NPROPERTIES"/>
</rdf:Description>
```

```
<rdf:Description ID="Family">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Family Name</rdfs:label>
  <rdfs:subClassOf rdf:resource="#NPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Given">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Given Name</rdfs:label>
  <rdfs:subClassOf rdf:resource="#NPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Other">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Other Names</rdfs:label>
  <rdfs:subClassOf rdf:resource="#NPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Prefix">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Prefix Name</rdfs:label>
  <rdfs:subClassOf rdf:resource="#NPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Suffix">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Suffix Name</rdfs:label>
  <rdfs:subClassOf rdf:resource="#NPROPERTIES"/>
</rdf:Description>


<rdfs:Class rdf:ID="ORGPROPERTIES"/>

<rdf:Description ID="ORG">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Organisation</rdfs:label>
  <rdfs:range rdf:resource="#ORGPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Orgname">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Organisation Name</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ORGPROPERTIES"/>
</rdf:Description>

<rdf:Description ID="Orgunit">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
```

```
  <rdfs:label>Organisation Unit</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ORGPROPERTIES"/>
 </rdf:Description>


 <rdf:Description ID="PHOTO">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Photograph</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="LOGO">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Logo Image</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="SOUND">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Audio Sound</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="KEY">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Public Key</rdfs:label>
 </rdf:Description>


 <rdf:Description ID="AGENT">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Agent</rdfs:label>
 </rdf:Description>

<rdf:Description ID="UID">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Unique ID</rdfs:label>
 </rdf:Description>

 <rdf:Description ID="GROUP">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Group vCard Properties</rdfs:label>
 </rdf:Description>

</rdf:RDF>
```