

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

X.891

(05/2005)

SERIE X: REDES DE DATOS, COMUNICACIONES DE
SISTEMAS ABIERTOS Y SEGURIDAD

Aplicaciones de interconexión de sistemas abiertos –
Aplicaciones genéricas de la notación de sintaxis
abstracta uno

**Tecnología de la información – Aplicaciones
genéricas de ASN.1 – Infoset rápido**

Recomendación UIT-T X.891

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.369
Redes basadas en el protocolo Internet	X.370–X.379
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Compromiso, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.889
Aplicaciones genéricas de la notación de sintaxis abstracta uno	X.890–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999
SEGURIDAD DE LAS TELECOMUNICACIONES	X.1000–

Para más información, véase la Lista de Recomendaciones del UIT-T.

Tecnología de la información – Aplicaciones genéricas de ASN.1 – Infoset rápido

Resumen

En esta Recomendación | Norma Internacional se especifica una forma de representación de un ejemplar del conjunto de información XML del W3C utilizando codificaciones binarias. Dichas codificaciones binarias se especifican mediante la notación de sintaxis abstracta (ASN.1) y la notación de control de codificación (ECN) ASN.1.

La tecnología que se especifica en esta Recomendación | Norma Internacional se denomina conjunto de información rápido (*Fast Infoset*). Proporciona una alternativa a la sintaxis W3C XML para la representación de ejemplares del conjunto de información W3C XML. En general, esta codificación es de menor tamaño y de procesamiento más rápido que una representación W3C XML.

Esta Recomendación | Norma Internacional especifica la utilización de diversas técnicas que minimizan el tamaño de las codificaciones (denominadas documentos infoset rápidos) y que maximizan la velocidad de creación y procesamiento de documentos infoset rápidos. Estas técnicas incluyen la utilización de tablas dinámicas (para cadenas de caracteres y para nombres cualificados), vocabularios iniciales y vocabularios externos.

La presente Recomendación | Norma Internacional también especifica un tipo de medios de ampliaciones multifunción del correo Internet (MIME) que permite identificar un documento infoset rápido.

Orígenes

La Recomendación UIT-T X.891 fue aprobada el 14 de mayo de 2005 por la Comisión de Estudio 17 (2005-2008) del UIT-T por el procedimiento de la Recomendación UIT-T A.8. Incluye las correcciones introducidas por el corrigendum 1 a la Recomendación UIT-T X.891 (2005), aprobado el 13 de junio de 2006 por la Comisión de Estudio 17 (2005-2008) del UIT-T por el establecimiento de la Recomendación UIT-T A.8. Se publica también un texto idéntico como Norma Internacional ISO/CEI 24824-1.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

© UIT 2007

Reservados todos los derechos. Ninguna parte de esta publicación puede reproducirse por ningún procedimiento sin previa autorización escrita por parte de la UIT.

ÍNDICE

	Página
1 Alcance	1
2 Referencias normativas.....	1
2.1 Recomendaciones Normas Internacionales idénticas.....	2
2.2 Referencias adicionales	2
3 Definiciones	3
3.1 Términos ASN.1	3
3.2 Términos ECN	3
3.3 Términos ISO/CEI 10646	3
3.4 Definiciones adicionales.....	3
4 Abreviaturas.....	4
5 Notación.....	4
6 Principios de la construcción y utilización de tablas de vocabulario	5
7 Definiciones de tipos ASN.1	6
7.1 Generalidades.....	6
7.2 El tipo Document	6
7.3 El tipo Element	11
7.4 El tipo Attribute	12
7.5 El tipo ProcessingInstruction	13
7.6 El tipo UnexpandedEntityReference.....	13
7.7 El tipo CharacterChunk	14
7.8 El tipo Comment	14
7.9 El tipo DocumentTypeDeclaration	15
7.10 El tipo UnparsedEntity	15
7.11 El tipo Notation	16
7.12 El tipo NamespaceAttribute	16
7.13 El tipo IdentifyingStringOrIndex	17
7.14 El tipo NonIdentifyingStringOrIndex	18
7.15 El tipo NameSurrogate	19
7.16 El tipo QualifiedNameOrIndex	19
7.17 El tipo EncodedCharacterString	21
8 Construcción y procesamiento de un documento infosec rápido.....	22
8.1 Ordenación conceptual de componentes de un valor abstracto del tipo Document	22
8.2 Tabla de alfabetos restringidos.....	23
8.3 Tabla de algoritmos de codificación.....	23
8.4 Tablas de cadenas dinámicas	24
8.5 Tablas de nombres dinámicas y sustitutos de nombres	24
9 Alfabetos restringidos integrados	25
9.1 Alfabeto restringido "numérico".....	25
9.2 Alfabeto restringido "fecha y hora".....	25
10 Algoritmos de codificación integrados	25
10.1 Generalidades.....	25
10.2 Algoritmo de codificación "hexadecimal"	26
10.3 Algoritmo de codificación "base64".....	26
10.4 Algoritmo de codificación "corto" ("short").....	26
10.5 Algoritmo de codificación "int"	27
10.6 Algoritmo de codificación "largo" ("long")	27
10.7 Algoritmo de codificación "booleano" ("boolean")	27
10.8 Algoritmo de codificación "flotante" ("float").....	28
10.9 Algoritmo de codificación "doble" ("double").....	28
10.10 Algoritmo de codificación "uuid"	29

	<i>Página</i>
10.11 Algoritmo de codificación "cdata"	29
11 Restricciones de los infoset XML soportados y otras simplificaciones.....	29
12 Codificación a nivel de bit del tipo Document	30
Anexo A – Módulo ASN.1 y módulos ECN para documentos infoset rápidos	32
A.1 Definición del módulo ASN.1.....	32
A.2 Definiciones del módulo ECN	34
Anexo B – Tipo de medios MIME para documentos infoset rápidos.....	54
Anexo C – Descripción de la codificación de un documento infoset rápido.....	56
C.1 Documento infoset rápido	56
C.2 Codificación del tipo Document	56
C.3 Codificación del tipo Element	58
C.4 Codificación del tipo Attribute	59
C.5 Codificación del tipo ProcessingInstruction	59
C.6 Codificación del tipo UnexpandedEntityReference	60
C.7 Codificación del tipo CharacterChunk	60
C.8 Codificación del tipo Comment	60
C.9 Codificación del tipo DocumentTypeDeclaration.....	60
C.10 Codificación del tipo UnparsedEntity	61
C.11 Codificación del tipo Notation	61
C.12 Codificación del tipo NamespaceAttribute.....	61
C.13 Codificación del tipo IdentifyingStringOrIndex.....	62
C.14 Codificación del tipo NonIdentifyingStringOrIndex comenzando por el primer bit de un octeto	62
C.15 Codificación del tipo NonIdentifyingStringOrIndex comenzando por el tercer bit de un octeto	62
C.16 Codificación del tipo NameSurrogate	63
C.17 Codificación del tipo QualifiedNameOrIndex comenzando por el segundo bit de un octeto.....	63
C.18 Codificación del tipo QualifiedNameOrIndex comenzando por el tercer bit de un octeto.....	64
C.19 Codificación del tipo EncodeCharacterString comenzando por el tercer bit de un octeto.....	64
C.20 Codificación del tipo EncodeCharacterString comenzando por el quinto bit de un octeto.....	65
C.21 Codificación de la longitud de un tipo sequence-of.....	65
C.22 Codificación del tipo NonEmptyOctetString comenzando por el segundo bit de un octeto.....	65
C.23 Codificación del tipo NonEmptyOctetString comenzando por el quinto bit de un octeto.....	66
C.24 Codificación del tipo NonEmptyOctetString comenzando por el séptimo bit de un octeto.....	66
C.25 Codificación de números enteros en la gama de 1 a 2^{20} comenzando por el segundo bit de un octeto.....	66
C.26 Codificación de números enteros en la gama de 0 a 2^{20} comenzando por el segundo bit de un octeto.....	67
C.27 Codificación de números enteros en la gama de 1 a 2^{20} comenzando por el tercer bit de un octeto.....	67
C.28 Codificación de números enteros en la gama de 1 a 2^{20} comenzando por el cuarto bit de un octeto.....	67
C.29 Codificación de números enteros en la gama de 1 a 256.....	68

	<i>Página</i>
Anexo D – Ejemplos de codificación de infosets XML como documentos infoiset rápidos	69
D.1 Introducción a los ejemplos	69
D.2 Tamaño de los documentos ejemplo (incluida la compresión basada en redundancia)	69
D.3 Ejemplo de orden UBL.....	70
D.4 Documento infoiset rápido de orden UBL con vocabulario externo	72
D.5 Documento infoiset rápido de orden UBL sin vocabulario inicial.....	80
Anexo E – Asignación de valores de identificador de objetos.....	91
BIBLIOGRAFÍA	92

Introducción

En esta Recomendación | Norma Internacional se especifica una representación de un ejemplar del conjunto de información XML del W3C utilizando codificaciones binarias (especificadas mediante la notación ASN.1 y notación de control de codificación ASN.1). La codificación especificada en la presente edición de esta Recomendación | Norma Internacional se identifica mediante el número de versión 1 (véase 12.9)

La tecnología que se especifican en esta Recomendación | Norma Internacional se denomina conjunto de información rápido (*Fast Infoset*). Proporciona una alternativa a la sintaxis W3C XML para la representación de ejemplares del conjunto de información W3C XML. En general, esta codificación es de menor tamaño y de procesamiento más rápido que una representación W3C XML.

La representación de un ejemplar del conjunto de información W3C XML especificada en esta Recomendación | Norma Internacional se denomina documento infoset rápido. Cada documento infoset rápido es la codificación de un valor abstracto de un tipo de datos ASN.1 (el tipo **Document** – véase 7.2) que representa un ejemplar del conjunto de información W3C XML.

Esta Recomendación | Norma Internacional especifica la utilización de diversas técnicas que minimizan el tamaño de un documento infoset rápido y maximizan la velocidad de creación y procesamiento de dichos documentos.

Estas técnicas se basan en la utilización de tablas de vocabulario, que permiten valores enteros relativamente pequeños (índices de tablas de vocabulario) que pueden ser utilizados en lugar de las cadenas de caracteres que forman (por ejemplo) los nombres de elementos o atributos en la serialización XML 1.0 de un ejemplar del conjunto de información W3C XML.

Existe un conjunto de tablas de vocabulario (véase la cláusula 8) de las que la más básica (tabla de cadenas de ocho caracteres) hace corresponder enteros normalmente pequeños con cadenas de caracteres. Sin embargo, existen tablas de vocabulario (la tabla de nombre de elemento y la tabla de nombre de atributo) que proporcionan un nivel adicional de indirección, con un índice de tabla de vocabulario que establece una correspondencia con un conjunto de tres índices de tabla de vocabulario que identifican un prefijo, un nombre de espacio de nombres y un nombre local.

Otra técnica relevante consiste en utilizar una tabla de vocabulario de alfabetos restringidos. Contiene entradas que enumeran un subconjunto de caracteres ISO/CEI 10646. Si es necesario codificar una cadena de caracteres para la que existe una entrada en dicha tabla, puede hacerse identificando que se utiliza dicha tabla de vocabulario, proporcionando el índice de la tabla de vocabulario y codificando cada carácter con el número mínimo de bits necesarios para dicho subconjunto de caracteres ISO/CEI 10646 en particular. Existe un conjunto de alfabetos restringidos integrados que siempre constituyen las primeras entradas de esta tabla, incluyendo cadenas de ocurrencia habitual tales como fecha, hora y valores numéricos.

Una optimización adicional importante utiliza la tabla de vocabulario de algoritmos de codificación. Esta tabla permite identificar codificaciones especializadas que pueden emplearse para cadenas que aparecen de forma habitual, incluyendo también en este caso varios algoritmos integrados. Por ejemplo, si existe una cadena que parece ser la representación decimal de un entero del rango -32768 a 32767 , la cadena puede codificarse identificando que se utiliza esta tabla de vocabulario, proporcionando el índice de la tabla de vocabulario y codificando a continuación el número entero como un entero con signo y dos octetos. Los números representados en coma flotante y las matrices de dichos tipos de números se soportan de la misma forma.

Para garantizar un procesamiento rápido sin sacrificar la compactibilidad, muchos componentes de un documento infoset rápido (tales como las cadenas de caracteres y los componentes que representan elementos de información del infoset XML) se alinean según los octetos, mientras que otros componentes (tales como longitudes e índices de tablas de vocabulario) no están necesariamente alineados según los octetos, pero sí terminan siempre en el último bit de un octeto. Para conseguir una especificación formal de dichas codificaciones optimizadas, se utiliza (véase A.2) la notación de control de codificación (ECN, *encoding control notation*) ASN.1 (definida en la Rec. UIT-T X.692 | ISO/CEI 8825-3), no siendo necesaria la utilización de herramientas ECN para la implementación, y proporcionándose una descripción completa de la codificación (véase anexo C).

Las tablas de vocabulario para un documento infoset rápido determinado pueden ser inicializadas mediante la información que figura en la cabecera del documento, y normalmente se añaden de forma dinámica, proporcionando flexibilidad al codificador. Las tablas de vocabulario iniciales pueden proporcionarse mediante una referencia al conjunto de tablas de vocabulario finales de algún otro documento infoset rápido identificado (o por otros medios). Esta referencia de vocabulario puede ser complementada mediante nuevas adiciones a la tabla a fin de proporcionar las tablas de vocabulario iniciales para el documento en cuestión. Generalmente, durante la creación o procesado del documento se realizan otras adiciones dinámicas.

Finalmente, se proporciona un mecanismo para que el generador de un documento infoset rápido incluya datos (denominados datos de procesamiento adicionales) relacionados con el procesamiento adicional facultativo del documento infoset rápido, junto con un URI que identifique completamente la forma y semántica de dichos datos de procesamiento adicionales. Éstos son ignorados por cualquier procesador subsiguiente del documento infoset rápido si el URI es desconocido, o si no se soporta o no se necesita el procesamiento especificado.

NOTA – Un ejemplo de dichos datos de procesamiento adicionales serían los datos que proporcionan índices que permiten el acceso inmediato a partes del documento infoset rápido, de forma que no sea necesario procesar todo el documento si sólo son de interés las partes del documento infoset rápido que se correspondan con una etiqueta XML específica.

El anexo A es parte integrante de esta Recomendación | Norma Internacional, y contiene un módulo ASN.1 (véase la Rec. UIT-T. X.680 | ISO/CEI 8824-1) y dos módulos ECN (EDM y ELM – véase la Rec. UIT-T X.692 | ISO/CEI 8825-3) que conjuntamente especifican el contenido abstracto y la codificación a nivel de bit de un valor del tipo **Document**, que incluye el valor de un ejemplar del conjunto de información W3C XML.

El anexo B es parte integrante de esta Recomendación | Norma Internacional, y contiene la especificación de un tipo de medios MIME que identifica un documento infoset rápido.

El anexo C no es parte integrante de esta Recomendación | Norma Internacional, y proporciona una descripción completa de las codificaciones que se especifican formalmente en las cláusulas 12 y en A.2.

El anexo D no es parte integrante de esta Recomendación | Norma Internacional, y proporciona ejemplos de documentos infoset rápidos generados a partir de documentos XML. Este anexo también proporciona el tamaño de la representación XML y de la representación infoset rápido de dichos ejemplos.

**NORMA INTERNACIONAL
RECOMENDACIÓN UIT-T**

Tecnología de la información – Aplicaciones genéricas de ASN.1 – Infoset rápido

1 Alcance

En esta Recomendación | Norma Internacional se especifica un tipo ASN.1 (véase la Rec. ITU-T X.680 | ISO/CEI 8824-1) cuyos valores abstractos representan ejemplares del conjunto de información W3C XML. También se especifican las codificaciones binarias para dichos valores, utilizando la notación de control de codificación (ECN, *encoding control notation*) ASN.1 (véase la Rec. UIT-T X.692 | ISO/CEI 8825-3).

NOTA – Dichas codificaciones se denominan documentos infoset (conjunto de información) rápidos.

Esta Recomendación | Norma Internacional también especifica técnicas para:

- minimizar el tamaño de los documentos infoset rápidos;
- maximizar la velocidad de creación y procesamiento de documentos infoset rápidos;
- permitir la especificación (mediante el generador de un documento infoset rápido) de datos de procesamiento adicionales.

Las primeras dos técnicas implican la utilización de tablas de vocabulario conceptuales. El conjunto de tablas de vocabulario y la naturaleza de sus entradas se define completamente en esta Recomendación | Norma Internacional, pero su representación en la memoria de una computadora queda fuera del alcance de la misma. La provisión de la transferencia o almacenamiento, o notación formal para la representación o especificación de tablas de vocabulario utilizadas como vocabulario externo queda también fuera del alcance de esta Recomendación | Norma Internacional.

La tercera técnica implica la provisión de datos de procesamiento adicionales y de un URI que identifique la forma y semántica de dichos datos. La especificación de formas concretas de datos de procesamiento adicionales y su utilización queda fuera del alcance de esta Recomendación | Norma Internacional.

Los URI pueden utilizarse para identificar vocabularios finales utilizables como una parte o el conjunto completo de algunos vocabularios iniciales nuevos, pero la asignación de URI específicos a vocabularios finales concretos queda fuera del alcance de esta Recomendación | Norma Internacional.

En esta Recomendación | Norma Internacional se especifican alfabetos restringidos integrados, la adición de tablas de vocabulario de alfabetos restringidos adicionales por enumeración y la utilización de dichas tablas de vocabulario para la codificación eficiente de cadenas de caracteres.

En esta Recomendación | Norma Internacional también se especifican alfabetos de codificación integrados para la codificación óptima de determinadas cadenas de caracteres, y la adición a las tablas de vocabulario de algoritmos de codificación adicionales identificados mediante URIs, sin embargo la definición de dichos algoritmos de codificación adicionales y sus URI asociados queda fuera del alcance de esta Recomendación | Norma Internacional.

Además, esta Recomendación | Norma Internacional especifica un tipo de medios de ampliaciones multifunción del correo Internet (MIME, *multipurpose Internet mail extensions*) que identifica un documento infoset rápido.

2 Referencias normativas

Las siguientes Recomendaciones y Normas Internacionales contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación | Norma Internacional. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y Normas Internacionales son objeto de revisiones, por lo que se preconiza que los participantes en acuerdos basados en la presente Recomendación | Norma Internacional investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y Normas Internacionales citadas a continuación. Los miembros de la CEI y de la ISO mantienen registros de las Normas Internacionales actualmente vigentes. La Oficina de Normalización de las Telecomunicaciones de la UIT publica periódicamente una lista de las Recomendaciones UIT-T vigentes. El IETF mantiene la lista de las RFC, junto con aquellas que han quedado obsoletas por ulteriores RFC. El W3C mantiene una lista de Recomendaciones W3C vigentes. En esta Recomendación | Norma Internacional, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación o Norma Internacional.

2.1 Recomendaciones | Normas Internacionales idénticas

- Recomendación UIT-T X.667 (2004) | ISO/CEI 9834-8:2005, *Tecnología de la información – Interconexión de sistemas abiertos – Procedimientos para el funcionamiento de autoridades de registro OSI: Generación y registro de Identificadores únicos universales (UUID) y su utilización como componentes de identificador de objetos ASN.1.*
- Recomendación UIT-T X.680 (2002) | ISO/CEI 8824-1:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica.*
- Recomendación UIT-T X.681 (2002) | ISO/CEI 8824-2:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de objetos de información.* †
- Recomendación UIT-T X.682 (2002) | ISO/CEI 8824-3:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones.* †
- Recomendación UIT-T X.683 (2002) | ISO/CEI 8824-4:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Parametrización de especificaciones de notación de sintaxis abstracta uno.* †
- Recomendación UIT-T X.690 (2002) | ISO/CEI 8825-1:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación básica, de las reglas de codificación canónica y de las reglas de codificación distinguida.* †
- Recomendación UIT-T X.691 (2002) | ISO/CEI 8825-2:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación compactada.* †
- Recomendación UIT-T X.692 (2002) | ISO/CEI 8825-3:2002, *Tecnología de la información – Reglas de codificación ASN.1: Especificación de la notación de control de codificación.*
- Recomendación UIT-T X.693 (2001) | ISO/CEI 8825-4:2002, *Tecnología de la información – Reglas de codificación ASN.1: Especificación las reglas de codificación XML (XER).* †

NOTA – Se ha enumerado el conjunto completo de Recomendaciones | Normas Internacionales ASN.1, pues todas ellas pueden ser aplicables a utilizaciones concretas de esta Recomendación | Norma Internacional. En los casos en que no son expresamente referenciadas en esta Recomendación | Norma Internacional, se añade a la referencia el símbolo †.

2.2 Referencias adicionales

- ISO 8601:2004, *Data elements and interchange formats – Information interchange – Representation of dates and times.*
- ISO/CEI 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set (UCS).*
- *The Unicode Standard, Version 4.0*, The Unicode Consortium (Reading, MA, Addison-Wesley).
NOTA 1 – Los caracteres gráficos (y sus codificaciones) definidos mediante Unicode son idénticos a los definidos mediante ISO/CEI 10646-1, no obstante, se incluye Unicode como referencia porque también especifica los nombres de caracteres de control y define la abreviatura UTF-16BE.
- W3C XML 1.0:2004, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2000/REC-xml-20040204/>.
- W3C XML 1.1:2004, *Extensible Markup Language (XML) 1.1*, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2000/REC-xml11-20040204/>.
NOTA 2 – Se incluyen las referencias a W3C XML 1.0 y a W3C XML 1.1 puesto que ninguna de ellas es un subconjunto de la otra. Estas referencias sólo se utilizan en 3.4.10.
- W3C XML Information Set:2004, *XML Information Set (Second Edition)*, W3C Recommendation, Copyright © [04 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2004/REC-xml-infoset-20040204/>.
- W3C XML Namespaces 1.0:1999, *Namespaces in XML*, W3C Recommendation, Copyright © [14 January 1999] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/1999/REC-xml-names-19990114/>.
- W3C XML Namespaces 1.1:2004, *Namespaces in XML 1.1*, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut

National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2004/REC-xm-l-names11-20040204/>.

NOTA 3 – Se incluyen las referencias a W3C XML Namespaces 1.0 y a W3C XML Namespaces 1.1 puesto que ninguna de ellas es un subconjunto de la otra. Estas referencias sólo se utilizan en 3.4.10.

- IETF RFC 2045 (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*.
- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.
- IEEE 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*.

3 Definiciones

A los efectos de esta Recomendación | Norma Internacional, se aplican las siguientes definiciones.

3.1 Términos ASN.1

En esta Recomendación | Norma Internacional se utilizan los términos siguientes definidos en la Rec. UIT-T X.680 | ISO/CEI 8824-1:

- a) tipo elección (*choice*);
- b) tipo secuencia (*sequence*);
- c) tipo secuencia de (*sequence-of*).

3.2 Términos ECN

En esta Recomendación | Norma Internacional se utilizan los términos siguientes, definidos en la Rec. UIT-T X.692 | ISO/CEI 8825-3:

- a) Módulos de definición de codificación (EDM, *encoding definition modules*);
- b) Módulo de enlace de codificación (ELM, *encoding link module*).

3.3 Términos ISO/CEI 10646

En esta Recomendación | Norma Internacional se utiliza el término siguiente definido en ISO/CEI 10646:

- a) Plano multilingüe básico.

3.4 Definiciones adicionales

3.4.1 Base64: Un mecanismo de codificación que representa un valor cadena de octetos como una cadena de caracteres utilizando un alfabeto restringido de 65 caracteres (véase 10.3 e IETF RFC 2045).

3.4.2 cadena de caracteres: Una cadena de caracteres abstractos ISO/CEI 10646, sin relación alguna con la forma en la que hayan sido codificados.

3.4.3 algoritmo de codificación: Especificación precisa sobre cómo codificar de forma eficiente en octetos una cadena de caracteres con características especificadas.

NOTA – Un ejemplo de ello es la codificación de una cadena tal como "-32176" mediante dos enteros binarios complemento a dos en dos octetos. La codificación de dos octetos estaría acompañada de un índice de tabla de vocabulario que identificara dicho algoritmo de codificación.

3.4.4 vocabulario externo: Un conjunto de tablas de vocabulario referenciadas mediante un URI (véase 7.2.14).

3.4.5 documento infoset rápido: Un conjunto de información (infoset) XML representado tal como se especifica en esta Recomendación | Norma Internacional.

3.4.6 vocabulario final: El contenido de las tablas de vocabulario al concluir la creación o procesado de un documento infoset rápido.

3.4.7 elemento de información: Cada uno de los tipos de elementos que constituyen un infoset XML.

3.4.8 vocabulario inicial: Conjunto de tablas de vocabulario establecidas mediante información de la cabecera del documento infoset rápido que opcionalmente hace referencia a un vocabulario externo y, también opcionalmente, proporciona entradas adicionales a las tablas.

3.4.9 sustituto de nombre: Un conjunto de tres índices de tabla de vocabulario (siendo las dos primeras opcionales) que se utilizan para representar un nombre cualificado (véase 3.4.11).

3.4.10 documento XML de nombres de espacio bien constituido: Un documento W3C XML 1.0 que está correctamente constituido de conformidad con W3C XML Namespaces 1.0, o bien, un documento W3C XML 1.1 que está correctamente constituido de conformidad con W3C XML Namespaces 1.1.

3.4.11 nombre cualificado: Conjunto que consta de las propiedades **[prefix]**, **[namespace name]**, y **[local name]** de un elemento de información **element** o de un elemento de información **attribute**.

3.4.12 alfabeto restringido: Un conjunto ordenado de diferentes caracteres ISO/CEI 10646, que permiten una codificación compacta de cualquier cadena de caracteres que conste exclusivamente de caracteres de dicho conjunto.

3.4.13 índice de tabla de vocabulario: Un valor entero positivo que identifica una entrada en una tabla de vocabulario.

3.4.14 tablas de vocabulario: Un conjunto de tablas conceptuales (típicamente, pero no necesariamente, construidas dinámicamente) asociadas a un documento infoset rápido, que contiene cadenas de caracteres u otra información, y que permite la utilización de valores enteros positivo generalmente reducidos (índices de tabla de vocabulario) que identifican sus entradas.

NOTA – Son ejemplos de tablas de vocabulario las que contienen cadenas de caracteres que constituyen la propiedad **[local name]** de los elementos de información **attribute** o **element**, o cadenas de caracteres correspondientes a secuencias de elementos de información **character** que son miembros de la propiedad **[children]** de los elementos de información **element**.

3.4.15 declaración XML: Codificación UTF-8 de una cadena de caracteres especificada (véase también 12.3) que puede incluirse al comienzo de un documento infoset rápido para identificar la codificación como documento infoset rápido y para distinguirlo de un documento W3C XML 1.0 o W3C XML 1.1.

3.4.16 infoset (conjunto de información) XML: Un conjunto de datos abstractos que describen la información incluida en un documento XML de espacio de nombres bien constituido, tal como se especifica en el conjunto de información W3C XML.

3.4.17 espacio en blanco XML: Uno o más caracteres TABULACIÓN HORIZONTAL (9), SALTO DE LÍNEA (10), RETORNO DE CARRO (13), o ESPACIO (32) de Unicode.

NOTA – Estos caracteres son los que se corresponden con la producción "S" en W3C XML 1.0 y en W3C XML 1.1 (véase W3C XML 1.0, 2.3 y W3C XML 1.1, 2.3). Los caracteres LÍNEA SIGUIENTE (133) y SEPARADOR DE LÍNEA (8232), que pueden existir en un documento de espacio de nombres bien constituido W3C XML 1.1 (véase W3C XML 1.1, 2.11), se convierten en caracteres SALTO DE LÍNEA gracias a un tratamiento de final de línea (véase W3C XML 1.1, 2.11). Si dichos caracteres existen en un infoset XML generado a partir de un documento de espacio de nombres bien constituido W3C XML 1.1, no se trata de espacios en blanco XML.

4 Abreviaturas

A los efectos de esta Recomendación | Norma Internacional, se utilizan las siguientes siglas.

ASN.1	Notación de sintaxis abstracta uno (<i>abstract syntax notation one</i>)
BMP	Plano multilingüe básico (<i>basic multilingual plane</i>)
ECN	Notación de control de codificación (<i>encoding control notation</i>)
MIME	Ampliaciones multifunción del correo Internet (<i>multipurpose Internet mail extensions</i>)
UBL	Lenguaje de negocios universal (<i>universal business language</i>)
URI	Identificador uniforme de recursos (<i>uniform resource identifier</i>)
UTF-8	Función de transformación universal de 8 bits (<i>universal transformation function 8-bit</i>) (véase ISO/CEI 10646, Anexo D)
UTF-16BE	Función de transformación universal de 16 bits big endian (<i>universal transformation function 16-bit big endian</i>) (véase Unicode, 2.6)
UUID	Identificador universalmente único (<i>universally unique identifier</i>)
XML	Lenguaje de marcaje extensible (<i>eXtended markup language</i>)

5 Notación

5.1 En esta Recomendación | Norma Internacional se utiliza la notación ASN.1 definida en la Rec. UIT-T X.680 | ISO/CEI 8824-1 para la definición formal de tipos de datos cuyas codificaciones son documentos infoset rápidos.

NOTA – En la cláusula 12 se especifica la aplicación de la Rec. UIT-T X.692 | ISO/CEI 8825-3 a las definiciones de tipo ASN.1, proporcionando la codificación a nivel de bit de un documento infoset rápido.

5.2 En esta Recomendación | Norma Internacional se utiliza el tipo de letra **bold Courier** para la notación ASN.1 y **bold Arial** para la sintaxis W3C XML y para los nombres de elementos de información del conjunto de información XML.

5.3 Los nombres de las propiedades de elementos de información se escriben con letra de tipo **bold Arial** y entre corchetes (por ejemplo, **[children]**).

5.4 Los nombres de categorías de cadenas de caracteres (véase 8.4.2) y los nombres de categorías de nombres cualificados (véase 8.5.4) se escriben en MAYÚSCULAS.

5.5 En esta Recomendación | Norma Internacional, las posiciones de los bits en un octeto se especifican utilizando la terminología primer bit, segundo bit, etc., hasta octavo bit, siendo el primer bit el bit más significativo del octeto y el octavo bit el bit menos significativo del mismo.

6 Principios de la construcción y utilización de tablas de vocabulario

6.1 Las tablas de vocabulario son tablas conceptuales que hacen corresponder un índice de tabla de vocabulario con una entrada de la tabla de vocabulario.

NOTA – No se define cuál es la representación de tablas de vocabulario en la memoria de una computadora, ni tampoco se considera la forma mediante la que en una implementación se hace corresponder un índice de tabla de vocabulario con una entrada de dicha tabla de vocabulario.

6.2 El creador de un documento infoset rápido a partir de un infoset XML determina el contenido de las tablas de vocabulario.

6.3 En el caso más general, la cabecera del documento infoset rápido puede hacer referencia a un conjunto de tablas de vocabulario (un vocabulario externo) así como a las adiciones a dicha tabla de vocabulario destinadas a conformar el vocabulario inicial para dicho documento infoset rápido. Durante la creación y procesamiento de un documento infoset rápido pueden producirse adiciones a las tablas de vocabulario, de forma que éstas crecen progresivamente para formar las tablas de vocabulario finales de dicho documento.

6.4 Algunas tablas de vocabulario crecen progresivamente desde un vocabulario inicial hasta un vocabulario final durante la creación y procesamiento de un documento infoset rápido, y por tanto, incluyen el término "dinámico" en su nombre. No existen mecanismos para la supresión de entradas de una tabla.

6.5 Los índices de una tabla de vocabulario se asignan implícitamente. La primera entrada a una tabla de vocabulario tiene un índice de tabla de vocabulario de uno, asignándose como valor del índice de la tabla de vocabulario de las entradas posteriores a dicha tabla el entero siguiente superior. Cuando en esta Recomendación | Norma Internacional se especifica que se añade algo a una tabla de vocabulario, ello implica que se asignará el siguiente índice de tabla de vocabulario.

NOTA – Los índices de tabla de vocabulario comienzan en uno, no en cero, debido a que el valor cero (cuando se permite) tiene el significado especial de "cadena de caracteres vacía" en un campo que en otro caso puede incluir un índice de tabla de vocabulario.

6.6 Para soportar esta asignación implícita de índices de tabla de vocabulario, se define completamente el orden conceptual de procesamiento de los componentes (a cualquier profundidad) de un documento infoset rápido (véase 8.1).

NOTA – Este orden es el mismo que el empleado en las codificaciones de los componentes en un documento infoset rápido. No implica necesariamente que la semántica del documento se procesa en dicho orden. El orden se define únicamente con el fin de asegurar que tanto el creador como el procesador de un documento infoset rápido asignan el mismo índice de tabla de vocabulario a una entrada dada de tabla de vocabulario.

6.7 Las tablas de vocabulario se utilizan para diversos fines (véase la cláusula 8), pero su función primaria es permitir la utilización de un índice de tabla de vocabulario en lugar de una entrada de tabla de vocabulario, siendo dichos índices más pequeños (pudiendo ser más rápidos de procesar) que una entrada de la tabla. En la cláusula 9 se especifican entradas que están integradas en el caso de determinadas tablas de vocabulario. Estas entradas siempre están implícitamente presentes en dichas tablas de vocabulario, con los índices de tabla de vocabulario especificados en dicha cláusula 9.

6.8 Para algunas categorías de cadenas de caracteres, el creador de un documento infoset rápido tiene la opción de añadir, o no, una cadena a una tabla de vocabulario, en función de número previsible (o conocido) de ocurrencias de dicha cadena de caracteres en el infoset XML.

6.9 La forma y significados precisos de las entradas de tabla de vocabulario se especifican en la cláusula 8, pero en la mayoría de los casos se trata de cadenas de caracteres de longitud variable, a menudo cortas, pero potencialmente con una longitud de hasta 2^{32} octetos.

6.10 Un creador que elabore un documento infoset rápido conforme a lo establecido, debe realizar todas las adiciones a las tablas de vocabulario tal como se especifica en 7.13.7, 7.14.6 a 7.14.7, y 7.16.7. Ello asegura que el número de entradas de tabla de vocabulario correspondientes a cada tabla de vocabulario nunca supera 2^{20} .

NOTA – Una entrada de tabla de vocabulario puede ser igual a una o más de las otras entradas de tabla de vocabulario. Esto es así a fin de permitir la creación eficaz de documentos infoset rápidos. No obstante, las entradas duplicadas reducirán la eficacia de la transferencia. Los procesadores no se ven afectados por las entradas duplicadas.

6.11 Un procesador de un documento infoset rápido que cumpla lo establecido debe realizar todas las adiciones a las tablas de vocabulario tal como se especifica en 7.13.8, 7.14.11 y 7.16.8. Ello garantiza que no se ha violado la restricción señalada en 6.10 a).

7 Definiciones de tipos ASN.1

7.1 Generalidades

7.1.1 En esta Recomendación | Norma Internacional se especifica un conjunto de tipos ASN.1 que soportan una representación del conjunto de información (infoset) XML. El tipo raíz de este conjunto de tipos es el tipo `Document`.

7.1.2 Se imponen algunas restricciones al contenido de los infoset XML y se hacen en la representación algunas simplificaciones (véase la cláusula 11) a fin de mejorar la usabilidad de la especificación y la eficiencia de las codificaciones realizadas.

NOTA – Un infoset XML que no satisfaga estas restricciones no puede ser representado como un documento infoset rápido, ni puede normalmente ser representado como un documento XML de nombres de espacio bien constituido.

7.1.3 Para cada tipo de elemento de información especificado en el conjunto de información W3C XML, se facilita la correspondiente definición de tipo ASN.1 de esta Recomendación | Norma Internacional. Esta definición de tipo es siempre un tipo secuencia, cuyos componentes se corresponden con las propiedades del elemento de información.

7.1.4 Algunas propiedades de los elementos de información no están incluidas en las definiciones de tipos ASN.1 (véase 11.4).

7.1.5 En algunos casos, el valor de una propiedad no incluida en las definiciones de tipos ASN.1 puede determinarse a partir del valor de las propiedades de los mismos o de otros elementos de información incluidos. En esos casos, la omisión de dicha propiedad simplifica la representación sin que se produzca pérdida de información. Sin embargo, existen algunos casos en los que el valor de una propiedad no incluida no puede determinarse a partir de otras propiedades. En tales casos, la omisión de dicha propiedad es una simplificación que no limita la utilidad de la especificación para la mayoría de los casos prácticos.

7.1.6 En la cláusula 12 se especifica la codificación del tipo `Document`.

7.2 El tipo `Document`

7.2.1 El tipo `Document` es el siguiente:

```
Document ::= SEQUENCE {
    additional-data          SEQUENCE (SIZE(1..one-meg)) OF
        additional-datum SEQUENCE {
            id                URI,
            data              NonEmptyOctetString } OPTIONAL,
    initial-vocabulary      SEQUENCE {
        external-vocabulary  URI OPTIONAL,
        restricted-alphabets SEQUENCE (SIZE(1..256)) OF
            NonEmptyOctetString OPTIONAL,
        encoding-algorithms SEQUENCE (SIZE(1..256)) OF
            NonEmptyOctetString OPTIONAL,
        prefixes             SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        namespace-names      SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        local-names          SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        other-ncnames        SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        other-uris           SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        attribute-values     SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
```

```

content-character-chunks SEQUENCE (SIZE(1..one-meg)) OF
    EncodedCharacterString OPTIONAL,
other-strings SEQUENCE (SIZE(1..one-meg)) OF
    EncodedCharacterString OPTIONAL,
element-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
    NameSurrogate OPTIONAL,
attribute-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
    NameSurrogate OPTIONAL }
(CONSTRAINED BY {
    -- Si el componente vocabulario inicial está presente,
    -- al menos uno de sus componentes estará presente -- })
OPTIONAL,
notations SEQUENCE (SIZE(1..MAX)) OF
    Notation OPTIONAL,
unparsed-entities SEQUENCE (SIZE(1..MAX)) OF
    UnparsedEntity OPTIONAL,
character-encoding-scheme NonEmptyOctetString OPTIONAL,
standalone BOOLEAN OPTIONAL,
version NonIdentifyingStringOrIndex OPTIONAL
    -- categoría OTHER STRING --,
children SEQUENCE (SIZE(0..MAX)) OF
    CHOICE {
        element Element,
        processing-instruction ProcessingInstruction,
        comment Comment,
        document-type-declaration DocumentTypeDeclaration }}

```

donde el valor **one-meg** es:

```
one-meg INTEGER ::= 1048576 -- Dos elevado a la potencia de 20
```

El tipo **NonEmptyOctetString** es:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

Donde el valor de **four-gig** es:

```
four-gig INTEGER ::= 4294967296 -- Dos elevado a la potencia de 32
```

El tipo **URI** es:

```
URI ::= NonEmptyOctetString
```

7.2.2 Los tipos **EncodedCharacterString**, **NameSurrogate**, **Notation**, **UnparsedEntity**, **NonIdentifyingStringOrIndex**, **Element**, **ProcessingInstruction**, **Comment**, y **DocumentTypeDeclaration** se definen en 7.17, 7.15, 7.11, 7.10, 7.14, 7.3, 7.5, 7.8 y 7.9 respectivamente.

7.2.3 El tipo **URI** será un **URI** según se especifica en IETF RFC 2396.

7.2.4 El componente **restricted-alphabets** de **initial-vocabulary** (si está presente) incluirá una o más cadenas de caracteres, cada una de las cuales tendrá caracteres de una alfabeto restringido. Cada cadena de caracteres contendrá al menos dos caracteres, y todos los caracteres de la cadena de caracteres será diferentes.

NOTA – La utilización de un alfabeto restringido para optimizar las codificaciones de las cadenas de caracteres se especifica en 7.17.6.

7.2.5 El componente **encoding-algorithms** de **initial-vocabulary** (si está presente) incluirá uno o más **URI**, cada uno de los cuales identifica un algoritmo de codificación.

NOTA – Existen algoritmos de codificación integrados definidos en esta Recomendación | Norma Internacional (véase la cláusula 10), para los que se especifican índices de tabla de vocabulario, pero que quedan fuera del alcance de esta Recomendación | Norma Internacional para la definición de algoritmos de codificación adicionales y sus **URI** asociados, ni tampoco se establecen aquí los medios necesarios para definir dichos algoritmos. En 8.3.3 se especifica la información necesaria para definir un algoritmo de codificación.

7.2.6 El tipo **Document** representa el elemento de información **document** de un infoset XML. Dado que los restantes elementos de información de un infoset XML son propiedades de dicho elemento de información o de un elemento que es un hijo o descendiente de dicho elemento (a cualquier profundidad), cada **Document** representa un infoset XML completo.

NOTA – Cada **Document** sin una referencia a un vocabulario externo (véase 7.2.13) también define un vocabulario final que puede utilizarse como vocabulario externo de algún otro documento infoset rápido.

7.2.7 El componente **additional-data** (si está presente) incluirá uno o más componentes **additional-datum** para permitir mecanismos adicionales de procesamiento de un documento infosef rápido.

NOTA 1 – Un ejemplo serían los datos que permiten que un procesador acceda a partes de un documento infosef rápido sin que sea necesario el procesamiento de todo el documento. La forma de dichos datos no ha sido normalizada.

NOTA 2 – El número de componentes **additional-datum** está restringido a 2^{20} (véase 7.2.1).

7.2.8 Cada **additional-datum** constará de:

- a) el componente **id** (un valor del tipo **URI**); el URI hará referencia a una especificación que defina la forma y semántica del componente **data**; y

NOTA – La forma de **additional-datum** puede especificarse como un tipo abstracto conjuntamente con una regla de codificación, o mediante cualquier otro medio que sea adecuado.

- b) el componente **data**, que es una cadena de octetos que contiene los datos de procesamiento adicionales.

7.2.9 La utilización de un componente **additional-data** está sujeta a lo siguiente:

- a) un procesador podrá ignorar un componente **additional-datum** salvo que reconozca el URI y considere que el procesamiento adicional es relevante para la actividad de dicho procesador;
- b) un procesador que ignore todos los componente **additional-datum** es, sin embargo, capaz de generar un infosef XML equivalente al infosef XML utilizado para generar el documento infosef rápido.

7.2.10 Pueden existir múltiples componentes **additional-datum** con el mismo URI, que serán procesados de acuerdo con la especificación asociada al URI.

7.2.11 El componente **initial-vocabulary** proporciona datos que (junto con algunas entradas de tabla integradas) determina íntegramente el contenido inicial de la tabla de alfabeto restringido (véase 8.2), la tabla del algoritmo de codificación (véase 8.3), las tablas dinámicas de cadenas (véase 8.4), y las tablas dinámicas de nombres (véase 8.5) de este documento infosef rápido (el vocabulario inicial del documento infosef rápido). Un vocabulario inicial consta de los datos siguientes:

- a) un conjunto ordenado de alfabetos restringidos (véase 8.2.2), que contiene al menos los alfabetos restringidos integrados (véase la cláusula 9);
- b) un conjunto ordenado de algoritmos de codificación (véase 8.3.2), que contiene al menos los algoritmos de codificación integrados (véase la cláusula 10);
- c) ocho conjuntos ordenados independientes de cadenas de caracteres, correspondientes a las ocho categorías de cadenas de caracteres especificadas en esta Recomendación | Norma Internacional (véase 8.4.2), conteniendo cada conjunto cero o más cadenas de caracteres de una única categoría; y
- d) dos conjuntos ordenados independientes de sustitutos de nombre (véase 8.5.2), correspondientes a las dos categorías de nombres cualificados especificados en esta Recomendación | Norma Internacional (véase 8.5.4), conteniendo cada conjunto cero o más sustitutos de nombres de una única categoría.

NOTA – Un vocabulario inicial no puede estar completamente vacío, debido a que siempre incluye (al menos) los alfabetos restringidos integrados y los algoritmos de codificación integrados. Sin embargo, no es extraño que un documento infosef rápido tenga un vocabulario inicial que solo contenga dichos datos, puesto que la decisión (que toma el creador de un documento infosef rápido) sobre cómo utilizar el componente **initial-vocabulary** es función de la implementación, y algunas implementaciones pueden optar por añadir dinámicamente todas las entradas de la tabla de vocabulario (en el cuerpo del documento infosef rápido).

7.2.12 El vocabulario inicial del documento infosef rápido se determinará tal como se indica a continuación:

- a) Si no existe el componente **initial-vocabulary**, el vocabulario inicial constará únicamente de las entradas de la tabla integrada especificadas en 7.2.21, 7.2.22, y las cláusulas 9 y 10.
- b) Si existe el componente **initial-vocabulary** y no existe el componente **external-vocabulary**, el vocabulario inicial constará de las entradas de tabla integradas especificadas en 7.2.21, 7.2.22, y las cláusulas 9 y 10, con las entradas adicionales de la tabla (si existen) determinadas en 7.2.16.
- c) Si existen tanto el componente **initial-vocabulary**, como el componente **external-vocabulary**, el vocabulario inicial constará del vocabulario final identificado mediante el componente **external-vocabulary**, tal como se especifica en 7.2.13 y 7.2.14, con las entradas adicionales de la tabla de vocabulario (si existen) determinadas en 7.2.16.

7.2.13 El componente **external-vocabulary** identifica un vocabulario final utilizando uno de los mecanismos especificados en 7.2.14. El tipo URI (véase 7.2.1) determina el vocabulario final que se debe utilizar como vocabulario externo con una de las tres formas posibles (véase 7.2.14).

NOTA – Esta Recomendación | Norma Internacional no especifica vocabularios externos ni ninguna URI que haga referencia a vocabularios externos. Dichos vocabularios externos y los URI pueden ser definidos por cualquier autoridad capaz de atribuir dichos URI, y pueden ser acordados de forma privada o estar sujetos a normalización.

7.2.14 Un vocabulario externo puede especificarse de una de las tres formas siguientes:

- a) como el vocabulario final de un documento infoset rápido, que no hará referencia a un vocabulario externo; o

NOTA 1 – La implementación determina si el vocabulario final se almacena localmente o si solamente se almacena el documento infoset rápido, y el vocabulario final se genera procesando éste.

NOTA 2 – La restricción de que el vocabulario final de un documento infoset rápido con una referencia a un vocabulario externo no puede ser utilizado como vocabulario externo se impone a fin de simplificar la implementación y evitar referencias circulares.

- b) como un documento XML de espacios de nombres bien constituido, que conceptualmente se procesa de la forma siguiente:

- 1) se determinará el infoset XML del documento XML de espacio de nombres bien constituido; y
- 2) el documento infoset rápido correspondiente a dicho infoset XML se creará tal como se especifica en esta Recomendación | Norma Internacional, pero no tendrá **initial-vocabulary**, el componente **add-to-table** del **NonIdentifyingStringOrIndex** (véase 7.14) tomará siempre el valor **VERDADERO**; y en ningún cuadro de cadenas habrá múltiples cadenas de caracteres idénticas; y
- 3) el vocabulario final de este documento infoset rápido se convierte en el vocabulario externo; o

NOTA 3 – La implementación determina si el vocabulario final se almacena localmente o si solamente se almacena el documento infoset rápido, y el vocabulario final se genera procesando éste.

- c) como un conjunto de tablas de vocabulario especificadas utilizando cualquier otro mecanismo o texto suficientemente preciso, que incluirá las entradas de tabla integradas de las cláusulas 9 y 10 (estando los índices de las tablas de vocabulario especificados en dichas cláusulas).

NOTA 4 – La especificación de una notación para la definición de tablas de vocabulario queda fuera del alcance de esta Recomendación | Norma Internacional.

NOTA 5 – El requisito de incluir las entradas de la tabla integradas cuando se utiliza este mecanismo asegura que todas las tablas de vocabulario incluyen las entradas de la tabla integradas.

7.2.15 Para un vocabulario externo especificado de conformidad con 7.2.14 c), se asignan índices consecutivos comenzando por 1 a todas las entradas de cadenas y nombres de la tabla, excepto a las de la tabla PREFIX y de la tabla NAMESPACE NAME. Las entradas de la tabla PREFIX y de la tabla NAMESPACE NAME recibirán índices consecutivos comenzando por 2. Todos los alfabetos restringidos distintos a los integrados recibirán índices consecutivos comenzando por 16. Todos los algoritmos de codificación distintos a los integrados recibirán índices consecutivos comenzando por 32.

7.2.16 Cada **NonEmptyOctetString**, **EncodedCharacterString** y **NameSurrogate** (si existe) que esté presente en cualquiera de los restantes componentes del **initial-vocabulary** se añadirá a una tabla de vocabulario de forma ordenada (véase 8.1), tal como se especifica en el cuadro 1 .

Cuadro 1 – Correspondencia entre identificadores de componentes y tablas de vocabulario

Identificador de componente	Tipo de entrada ASN.1	Tabla de vocabulario (véase cláusula 8)
restricted-alphabets	NonEmptyOctetString	La tabla de alfabeto restringido (véase 8.2)
encoding-algorithms	NonEmptyOctetString	La tabla del algoritmo de codificación (véase 8.3)
prefixes	NonEmptyOctetString	La tabla PREFIX (véase 8.4)
namespace-names	NonEmptyOctetString	La tabla NAMESPACE NAME (véase 8.4)
local-names	NonEmptyOctetString	La tabla LOCAL NAME (véase 8.4)
other-ncnames	NonEmptyOctetString	La tabla OTHER NCNAME (véase 8.4)
other-uris	NonEmptyOctetString	La tabla OTHER URI (véase 8.4)
attribute-values	EncodedCharacterString	La tabla ATTRIBUTE VALUE (véase 8.4)
content-character-chunks	EncodedCharacterString	La tabla CONTENT CHARACTER CHUNK (véase 8.4)
other-strings	EncodedCharacterString	La tabla OTHER STRING (véase 8.4)
element-name-surrogates	NameSurrogate	La tabla ELEMENT NAME (véase 8.5)
Attribute-name-surrogates	NameSurrogate	La tabla ATTRIBUTE NAME (véase 8.5)

7.2.17 Un valor del tipo `NonEmptyOctetString` incluirá la codificación UTF-8 (véase ISO/CEI 10646, anexo D) de una cadena de caracteres.

7.2.18 La tabla de alfabeto restringido y la tabla del algoritmo de codificación de un vocabulario inicial tendrá como máximo 256 entradas. Todas las tablas tendrán como máximo 2^{20} entradas.

NOTA – La restricción del número de entradas tiene por objetivo garantizar límites superiores comunes de los índices de las tablas. La restricción también se aplica si las entradas de la tabla se añaden dinámicamente (véase 7.13.7, 7.14.6, 7.14.7, y 7.16.7). Estas restricciones no impiden la codificación de cualquier infoset XML como documento infoset rápido.

7.2.19 Los índices de tablas de vocabulario de los alfabetos restringidos integrados tienen valores comprendidos entre 1 y 2 (véase la cláusula 9). Los índices de la tabla de vocabulario de los alfabetos restringidos en el componente `restricted-alphabets` de `initial-vocabulary` (si existe) se asignará de la forma siguiente:

- a) si no existe vocabulario externo, o si éste solamente contiene alfabetos integrados restringidos, los índices se asignarán comenzando por el valor 16;
- b) en cualquier otro caso, los índices se asignarán comenzando por el mayor valor del índice de alfabeto restringido del vocabulario externo más uno.

NOTA – Ello significa que no se utilizan los índices de tabla de vocabulario comprendidos entre 3 y 15. Estos valores están reservados para versiones futuras de esta Recomendación | Norma Internacional.

7.2.20 Los algoritmos de codificación integrados tienen índices de tablas de vocabulario comprendidos entre 1 y 10 (véase la cláusula 10). Los índices de tablas de vocabulario de los algoritmos de codificación presentes en el componente `encoding-algorithms` de `initial-vocabulary` (si existe) se asignarán de la forma siguiente:

- a) si no existe un vocabulario externo, o el vocabulario externo solo contiene algoritmos de codificación integrados, los índices se asignarán comenzando por el valor 32;
- b) en cualquier otro caso, los índices se asignarán comenzando por el mayor valor del índice de alfabeto restringido del vocabulario externo más uno.

NOTA – Esto significa que los índices de tabla de vocabulario de 11 a 31 no se utilizan. Estos valores están reservados para versiones futuras de esta Recomendación | Norma Internacional.

7.2.21 La tabla PREFIX tendrá una entrada de prefijo integrada de "xml", a la que se le asigna un índice de 1. Los índices de tablas de vocabulario de los prefijos del componente `prefixes` de `initial-vocabulary` (si existe) se asignarán de la forma siguiente:

- a) si no existe un vocabulario externo, o el vocabulario externo solo contiene la entrada de prefijo integrada, los índices se asignarán comenzando por el valor 2;
- b) en cualquier otro caso, los índices se asignarán comenzando por el mayor valor del índice de alfabeto restringido del vocabulario externo más uno.

7.2.22 La tabla NAMESPACE NAME tendrá una entrada integrada de nombre de espacio de nombres de:

<http://www.w3.org/XML/1998/namespace>

a la que se le asignará el índice 1.

7.2.23 Los índices de tabla de vocabulario de los nombre de espacio de nombres del componente `namespace-names` de `initial-vocabulary` (si existe) se asignarán de la forma siguiente:

- a) si no existe un vocabulario externo, o el vocabulario externo solo contiene la entrada de nombre de espacio de nombres integrada, los índices se asignarán comenzando por el valor 2;
- b) en cualquier otro caso, los índices se asignarán comenzando por el mayor valor del índice de nombre del espacio de nombres del vocabulario externo más uno.

7.2.24 El componente `notations` representa la propiedad `[notations]` del elemento de información `document`. Este componente es del tipo secuencia de, incluso aunque la propiedad `[notations]` se especifique en el conjunto de información W3C XML como un conjunto no ordenado (de elementos de información `notation`).

NOTA – En esta y en cualquier circunstancia, se utiliza un tipo secuencia de en lugar de un tipo conjunto de debido a que este último no satisface la necesidad de una ordenación estricta de todos los componentes de un documento infoset rápido (véase 8.1).

7.2.25 El componente `unparsed-entities` representa la propiedad `[unparsed entities]` del elemento de información `document`. Este componente es del tipo secuencia de, incluso aunque la propiedad `[unparsed entities]` se especifique en el conjunto de información W3C XML como un conjunto no ordenado (de elementos de información `unparsed entity`).

7.2.26 El componente `character-encoding-scheme` representa la propiedad `[character encoding scheme]` del elemento de información `document`. Este componente es del tipo `NonEmptyOctetString` y un valor de este componente incluirá la codificación UTF-8 (véase ISO/CEI 10646, anexo D) de la propiedad `[character encoding scheme]`. La ausencia de este componente en un valor abstracto del tipo `Document` indica que la propiedad `[character encoding scheme]` tiene un valor de "UTF-8".

NOTA – Soportar la propiedad `[character encoding scheme]` permite la transformación en ambos sentidos de documentos XML en documentos infoset rápidos sin modificar el esquema de codificación de caracteres. El creador de un documento infoset rápido a partir de un documento XML puede codificar la propiedad `[character encoding scheme]` obtenida de la declaración de codificación del documento XML (véase W3C XML 1.0, 4.3.1 y W3C XML 1.1, 4.3.1). Si el procesador de un documento infoset rápido desea producir la codificación original, puede utilizar el componente `character-encoding-scheme` (si existe).

7.2.27 El componente `standalone` representa la propiedad `[standalone]` del elemento de información `document`. El valor abstracto `TRUE` representa el valor `yes` de esta propiedad, y el valor abstracto `FALSE` representa el valor `no`. La ausencia de este componente en un valor abstracto del tipo `Document` indica que la propiedad `[standalone]` no tiene valor.

7.2.28 El componente `version` representa la propiedad `[version]` del elemento de información `document`. Este componente es del tipo `NonIdentifyingStringOrIndex` (véase 7.14), representando en este caso una cadena de caracteres de la categoría OTHER STRING. La ausencia de este componente en un valor abstracto del tipo `Document` indica que la propiedad `[standalone]` no tiene valor.

7.2.29 El componente `children` representa la propiedad `[children]` del elemento de información `document`. Exactamente uno de los elementos de secuencia de (en cualquier posición) utilizará la alternativa `element` del tipo elección, y como máximo uno de los elementos (en cualquier posición) utilizará la alternativa `document-type-declaration`. Cada uno de los restantes elementos (si los hay) utilizarán la alternativa `processing-instruction` o la alternativa `comment`.

7.2.30 La propiedad `[document element]` del elemento de información `document` no está incluida en el tipo `Document`. El valor de esta propiedad es siempre el único y exclusivo elemento de información `element` que es miembro de la propiedad `[children]` del elemento de información `document`.

7.2.31 La propiedad `[base URI]` del elemento de información `document` no está incluida en el tipo `Document` y no es soportada por esta Recomendación | Norma Internacional.

7.2.32 La propiedad `[all declarations processed]` del elemento de información `document` no está incluida en el tipo `Document`, y se asume que tiene el valor `true` (véase 11.3).

7.3 El tipo `Element`

7.3.1 El tipo `Element` es:

```

Element ::= SEQUENCE {
    namespace-attributes SEQUENCE (SIZE(1..MAX)) OF
        NamespaceAttribute OPTIONAL,
    qualified-name       QualifiedNameOrIndex
        -- categoría ELEMENT NAME --,
    attributes           SEQUENCE (SIZE(1..MAX)) OF
        Attribute OPTIONAL,
    children             SEQUENCE (SIZE(0..MAX)) OF
        CHOICE {
            element                Element,
            processing-instruction ProcessingInstruction,
            unexpanded-entity-reference UnexpandedEntityReference,
            character-chunk         CharacterChunk,
            comment                 Comment }}

```

7.3.2 Los tipos `NameSpaceAttribute`, `QualifiedNameOrIndex`, `Attribute`, `ProcessingInstruction`, `UnexpandedEntityReference`, `CharacterChunk` y `Comment` se definen en 7.12, 7.16, 7.4, 7.5, 7.6, 7.7, y 7.8 respectivamente.

7.3.3 El tipo `Element` representa al elemento de información `element` del conjunto de información XML.

7.3.4 El componente `namespace-attributes` representa la propiedad `[namespace attributes]` del elemento de información `element`. Este componente es del tipo secuencia de, aunque la propiedad `[namespace attributes]` se especifique en el conjunto de información W3C XML como un conjunto no ordenado (de los elementos de información `attribute`).

NOTA – El tipo del componente de secuencia de es `NamespaceAttribute` (en lugar de ser `Attribute`), incluso aunque la propiedad `[namespace attributes]` del elemento de información `element` se especifique en el conjunto de información W3C XML

como un conjunto de elementos de información **attribute**. En un infoset XML restringido (véase 11.3), las propiedades de un elemento de información **namespace** pueden determinarse a partir de las propiedades de un elemento de información **attribute** que represente un atributo espacio de nombres. La inversa solo es parcialmente cierta, pero esta limitación puede considerarse aceptable para las utilidades previsibles de esta Recomendación | Norma Internacional. (Véase también la nota incluida en 7.2.24.)

7.3.5 El componente **qualified-name** representa el nombre cualificado (véase 3.4.11) del elemento de información **element** (es decir, el conjunto que consta de las propiedades **[prefix]**, **[namespace name]** y **[local name]** de dicho elemento de información). Este componente es del tipo **QualifiedNameOrIndex** (véase 7.16), que en este caso representa un nombre cualificado de la categoría ELEMENT NAME.

7.3.6 El componente **attributes** representa la propiedad **[attributes]** del elemento de información **element**. Este componente es del tipo secuencia de, incluso aunque la propiedad **[attributes]** se especifique en el conjunto de información W3C XML como un conjunto no ordenado (de los elementos de información **attribute**).

7.3.7 El componente **children** representa la propiedad **[children]** del elemento de información **element**. Cuando dos o más hijos adyacentes son elementos de información **character**, puede utilizarse un único elemento **CharacterChunk** para representar dichos elementos de información **character** adyacentes.

NOTA – Si existe un secuencia de N caracteres adyacentes entre los hijos de un elemento de información **element**, se permite cualquier agrupación de dichos N caracteres en series de fragmentos de caracteres consecutivos. Sin embargo, es previsible que el creador de un documento infoset rápido haga cada fragmento de caracteres tan grande como sea posible para que las codificaciones sean eficientes.

7.3.8 La propiedad **[in-scope namespaces]** del elemento de información **element** no está incluida en el tipo **Element**.

NOTA – En un infoset XML restringido (véase 11.3), la propiedad **[in-scope namespaces]** de un elemento de información **element** puede determinarse a partir de la propiedad **[namespace attributes]** del elemento de información **element**, junto con la propiedad **[namespace attributes]** de todos los elementos de información **element** (si existen) que contengan (directa o indirectamente) dicho elemento de información **element**.

7.3.9 La propiedad **[base URI]** del elemento de información **element** no está incluida en el tipo **Element** y esta Recomendación | Norma Internacional no la soporta.

7.3.10 La propiedad **[parent]** del elemento de información **element** no está incluida en el tipo **Element**. El valor de esta propiedad, para cualquier elemento de información **element**, es el elemento de información **document** o **element** que contenga dicho elemento de información formando parte de su propiedad **[children]**.

7.4 El tipo **Attribute**

7.4.1 El tipo **Attribute** es:

```
Attribute ::= SEQUENCE {
    qualified-name      QualifiedNameOrIndex
                        -- categoría ATTRIBUTE NAME --,
    normalized-value   NonIdentifyingStringOrIndex
                        -- categoría ATTRIBUTE VALUE -- }
```

7.4.2 Los tipos **QualifiedNameOrIndex** y **NonIdentifyingStringOrIndex** se definen en 7.16 y 7.14 respectivamente.

7.4.3 El tipo **Attribute** representa el elemento de información **attribute** del conjunto de información XML.

7.4.4 El componente **qualified-name** representa el nombre cualificado (véase 3.4.11) del elemento de información **attribute** (es decir, el conjunto que consta de las propiedades **[prefix]**, **[namespace name]** y **[local name]** de este elemento de información). Este componente es del tipo **QualifiedNameOrIndex** (véase 7.16), que en este caso representa un nombre cualificado de la categoría ATTRIBUTE NAME.

7.4.5 El componente **normalized-value** representa la propiedad **[normalized value]** del elemento de información **attribute**. Este componente es del tipo **NonIdentifyingStringOrIndex** (véase 7.14), que en este caso representa una cadena de caracteres de la categoría ATTRIBUTE VALUE.

7.4.6 La longitud de la cadena de caracteres asignada a **normalized-value** no puede ser mayor de 2³².

NOTA – Esta restricción se deriva de la definición ASN.1, diseñada para optimizar las codificaciones y simplificar la implementación (véase también 11.3 j).

7.4.7 La propiedad **[specified]** del elemento de información **attribute** no está incluida en el tipo **Attribute**.

7.4.8 La propiedad **[attribute type]** del elemento de información **attribute** no está incluida en el tipo **Attribute**.

7.4.9 La propiedad **[references]** del elemento de información **attribute** no está incluida en el tipo **Attribute**.

NOTA – En un infoset XML restringido (véase 11.3), la propiedad **[references]** de un elemento de información **attribute** puede determinarse a partir de la propiedad **[normalized value]** del elemento de información **attribute**, junto con las propiedades de otros elementos de información del infoset XML.

7.4.10 La propiedad **[owner element]** del elemento de información **attribute** no está incluida en el tipo **Attribute**. El valor de esta propiedad, para cualquier elemento de información **attribute** dado, es el elemento de información **element** que contenga dicho elemento de información formando parte de su propiedad **[attributes]**.

7.5 El tipo **ProcessingInstruction**

7.5.1 El tipo **ProcessingInstruction** es:

```
ProcessingInstruction ::= SEQUENCE {
    target      IdentifyingStringOrIndex
                -- categoría OTHER NCNAME --,
    content     NonIdentifyingStringOrIndex
                -- categoría OTHER STRING -- }
```

7.5.2 Los tipos **IdentifyingStringOrIndex** y **NonIdentifyingStringOrIndex** se definen en 7.13 y 7.14 respectivamente.

7.5.3 El tipo **ProcessingInstruction** representa el elemento de información **processing instruction** del conjunto de información XML.

7.5.4 El componente **target** representa la propiedad **[target]** del elemento de información **processing instruction**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER NCNAME.

7.5.5 El componente **content** representa la propiedad **[content]** del elemento de información **processing instruction**. Este componente es del tipo **NonIdentifyingStringOrIndex** (véase 7.14), que en este caso representa una cadena de caracteres de la categoría OTHER STRING.

7.5.6 La longitud de la cadena de caracteres asignada a **content** no puede ser mayor de 2³².

NOTA – Esta restricción se deriva de la definición ASN.1, diseñada para optimizar las codificaciones y para simplificar la implementación (véase también 11.3 j).

7.5.7 La propiedad **[notation]** del elemento de información **processing instruction** no está incluida en el tipo **ProcessingInstruction**.

NOTA – En un infoset XML restringido (véase 11.3), la propiedad **[notation]** de un elemento de información **processing instruction** puede determinarse a partir de la propiedad **[target]** del elemento de información **processing instruction**, junto con la propiedad **[notations]** del elemento de información **document**.

7.5.8 La propiedad **[parent]** del elemento de información **processing instruction** no está incluida en el tipo **ProcessingInstruction**. El valor de esta propiedad es, para cualquier elemento de información **processing instruction** dado, es el elemento de información **document**, **element** o **document type definition** que contenga dicho elemento de información formando parte de su propiedad **[children]**.

7.6 El tipo **UnexpandedEntityReference**

7.6.1 El tipo **UnexpandedEntityReference** es:

```
UnexpandedEntityReference ::= SEQUENCE {
    name        IdentifyingStringOrIndex
                -- categoría OTHER NCNAME --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
                -- categoría OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
                -- categoría OTHER URI -- }
```

7.6.2 El tipo **IdentifyingStringOrIndex** se define en 7.13.

7.6.3 El tipo **UnexpandedEntityReference** representa el elemento de información **unexpanded entity reference** del conjunto de información XML.

7.6.4 El componente **name** representa la propiedad **[name]** del elemento de información **unexpanded entity reference**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER NCNAME.

7.6.5 El componente **system-identifier** representa la propiedad [**system identifier**] del elemento de información **unexpanded entity reference**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo **UnexpandedEntityReference** indica que la propiedad [**system identifier**] no tiene un valor.

7.6.6 El componente **public-identifier** representa la propiedad [**public identifier**] del elemento de información **unexpanded entity reference**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo **UnexpandedEntityReference** indica que la propiedad [**public identifier**] no tiene un valor.

7.6.7 La propiedad [**declaration base URI**] del elemento de información **unexpanded entity reference** no está incluida en el tipo **UnexpandedEntityReference** y no se soporta en esta Recomendación | Norma Internacional.

7.6.8 La propiedad [**parent**] del elemento de información **unexpanded entity reference** no está incluida en el tipo **UnexpandedEntityReference**. El valor de esta propiedad, para cualquier elemento de información **unexpanded entity reference** dado, es el elemento de información **element** que contenga dicho elemento de información formando parte de su propiedad [**children**].

7.7 El tipo **CharacterChunk**

7.7.1 El tipo **CharacterChunk** es:

```
CharacterChunk ::= SEQUENCE {
    character-codes          NonIdentifyingStringOrIndex
                           -- categoría CONTENT CHARACTER CHUNK -- }
```

7.7.2 El tipo **NonIdentifyingStringOrIndex** se define en 7.14.

7.7.3 El tipo **CharacterChunk** se corresponde con el elemento de información **character**, pero representa una serie de elementos de información **character** adyacentes (miembros del [**children**] del elemento de información **element** padre) en lugar de un único elemento de información **character**.

7.7.4 El número de elementos de información **character** representados por un valor del tipo **CharacterChunk** será distinto de cero.

7.7.5 El componente **character-codes** representa la propiedad [**character code**] del elemento o múltiples elementos de información **character** del fragmento (*chunk*). Este componente es del tipo **NonIdentifyingStringOrIndex** (véase 7.14), que en este caso representa una cadena de caracteres de la categoría CONTENT CHARACTER CHUNK.

7.7.6 La longitud de la cadena de caracteres asignada a **character-codes** no puede ser mayor de 2^{32} .

NOTA – Esta restricción se deriva de la definición ASN.1, diseñada para optimizar las codificaciones y para simplificar la implementación. Esta restricción no impide que la codificación de un elemento de información **element** contenga más de 2^{32} elementos de información **character**, en virtud de la utilización de múltiples fragmentos.

7.7.7 La propiedad [**element content whitespace**] del elemento o elementos de información **character** no está incluida en el tipo **CharacterChunk**.

7.7.8 La propiedad [**parent**] del elemento o elementos de información **character** no está incluida en el tipo **CharacterChunk**. El valor de esta propiedad, para cualquier elemento de información **character** dado, es el elemento de información **element** que contenga dicho elemento de información formando parte de su propiedad [**children**].

7.8 El tipo **Comment**

7.8.1 El tipo **Comment** es:

```
Comment ::= SEQUENCE {
    content          NonIdentifyingStringOrIndex -- categoría OTHER STRING -- }
```

7.8.2 El tipo **NonIdentifyingStringOrIndex** se define en 7.14.

7.8.3 El tipo **Comment** representa el elemento de información **comment** del conjunto de información XML.

7.8.4 El componente **content** representa la propiedad [**content**] del elemento de información **comment**. Este componente es del tipo **NonIdentifyingStringOrIndex** (véase 7.14), que en este caso representa una cadena de caracteres de la categoría OTHER STRING.

7.8.5 La longitud de la cadena de caracteres asignada a **content** no puede ser mayor de 2³².

NOTA – Esta restricción se deriva de la definición ASN.1, diseñada para optimizar las codificaciones y para simplificar la implementación (véase también 11.3 j).

7.8.6 La propiedad **[parent]** del elemento de información **comment** no está incluida en el tipo **Comment**. El valor de esta propiedad, para un elemento de información **comment** dado, es el elemento de información **document** o **element** que contenga dicho elemento de información formando parte de su propiedad **[children]**.

7.9 El tipo **DocumentTypeDeclaration**

7.9.1 El tipo **DocumentTypeDeclaration** es:

```
DocumentTypeDeclaration ::= SEQUENCE {
    system-identifier    IdentifyingStringOrIndex OPTIONAL
                        -- categoría OTHER URI --,
    public-identifier    IdentifyingStringOrIndex OPTIONAL
                        -- categoría OTHER URI --,
    children             SEQUENCE (SIZE(0..MAX)) OF
                        ProcessingInstruction }
```

7.9.2 El tipo **IdentifyingStringOrIndex** se define en 7.13.

7.9.3 El tipo **DocumentTypeDeclaration** representa el elemento de información **document type declaration** del conjunto de información XML.

7.9.4 El componente **system-identifier** representa la propiedad **[system identifier]** del elemento de información **document type declaration**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo **DocumentTypeDeclaration** indica que la propiedad **[system identifier]** no tiene valor.

7.9.5 El componente **public-identifier** representa la propiedad **[public identifier]** del elemento de información **document type declaration**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo **DocumentTypeDeclaration** indica que la propiedad **[public identifier]** no tiene un valor.

7.9.6 El componente **children** representa la propiedad **[children]** del elemento de información **document type declaration**.

7.9.7 La propiedad **[parent]** del elemento de información **document type declaration** no está incluida en el tipo **DocumentTypeDeclaration**. El valor de esta propiedad, para un elemento de información **document type declaration** dado, es el elemento de información **document** que contenga dicho elemento de información formando parte de su propiedad **[children]**.

7.10 El tipo **UnparsedEntity**

7.10.1 El tipo **UnparsedEntity** es:

```
UnparsedEntity ::= SEQUENCE {
    name                IdentifyingStringOrIndex
                        -- categoría OTHER NCNAME --,
    system-identifier    IdentifyingStringOrIndex
                        -- categoría OTHER URI --,
    public-identifier    IdentifyingStringOrIndex OPTIONAL
                        -- categoría OTHER URI --,
    notation-name        IdentifyingStringOrIndex
                        -- categoría OTHER NCNAME -- }
```

7.10.2 El tipo **IdentifyingStringOrIndex** se define en 7.13.

7.10.3 El tipo **UnparsedEntity** representa el elemento de información **unparsed entity** del conjunto de información XML.

7.10.4 El componente **name** representa la propiedad **[name]** del elemento de información **unparsed entity**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER NCNAME.

7.10.5 El componente **system-identifier** representa la propiedad **[system identifier]** del elemento de información **unparsed entity**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI.

7.10.6 El componente `public-identifier` representa la propiedad `[public identifier]` del elemento de información `unparsed entity`. Este componente es del tipo `IdentifyingStringOrIndex` (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo `UnparsedEntity` indica que la propiedad `[public identifier]` no tiene un valor.

7.10.7 El componente `notation-name` representa la propiedad `[notation name]` del elemento de información `unparsed entity`. Este componente es del tipo `IdentifyingStringOrIndex` (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER NCNAME.

7.10.8 La propiedad `[declaration base URI]` del elemento de información `unparsed entity` no está incluida en el tipo `UnparsedEntity` y no se soporta en esta Recomendación | Norma Internacional.

7.10.9 La propiedad `[notation]` del elemento de información `unparsed entity` no está incluida en el tipo `UnparsedEntity`.

NOTA – En un infoset XML restringido (véase 11.3), la propiedad `[notation]` de un elemento de información `unparsed entity` puede determinarse a partir de la propiedad `[notation name]` del elemento de información `unparsed entity` conjuntamente con la propiedad `[notations]` del elemento de información `document`.

7.11 El tipo `Notation`

7.11.1 El tipo `Notation`

```
Notation ::= SEQUENCE {
    name                IdentifyingStringOrIndex
                       -- categoría OTHER NCNAME --,
    system-identifier   IdentifyingStringOrIndex OPTIONAL
                       -- categoría OTHER URI --,
    public-identifier   IdentifyingStringOrIndex OPTIONAL
                       -- categoría OTHER URI -- }
```

7.11.2 El tipo `IdentifyingStringOrIndex` se define en 7.13.

7.11.3 El tipo `Notation` representa el elemento de información `notation` del conjunto de información XML.

7.11.4 El componente `name` representa la propiedad `[name]` del elemento de información `notation`. Este componente es del tipo `IdentifyingStringOrIndex` (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER NCNAME.

7.11.5 El componente `system-identifier` representa la propiedad `[system identifier]` del elemento de información `notation`. Este componente es del tipo `IdentifyingStringOrIndex` (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo `Notation` indica que la propiedad `[system identifier]` no tiene un valor.

7.11.6 El componente `public-identifier` representa la propiedad `[public identifier]` del elemento de información `notation`. Este componente es del tipo `IdentifyingStringOrIndex` (véase 7.13), que en este caso representa una cadena de caracteres de la categoría OTHER URI. La ausencia de este componente en un valor abstracto del tipo `Notation` indica que la propiedad `[public identifier]` no tiene un valor.

7.11.7 La propiedad `[declaration base URI]` del elemento de información `notation` no está incluida en el tipo `Notation` y no se soporta en esta Recomendación | Norma Internacional.

7.12 El tipo `NamespaceAttribute`

7.12.1 El tipos `NamespaceAttribute` es:

```
NamespaceAttribute ::= SEQUENCE {
    prefix                IdentifyingStringOrIndex OPTIONAL
                       -- categoría PREFIX --,
    namespace-name        IdentifyingStringOrIndex OPTIONAL
                       -- categoría NAMESPACE NAME -- }
```

7.12.2 El tipo `IdentifyingStringOrIndex` se define en 7.13.

7.12.3 El tipo `NamespaceAttribute` representa un elemento de información `attribute` que es miembro de la propiedad `[namespace attributes]` de un elemento de información `element` del conjunto de información XML.

NOTA – En el conjunto de información XML, tanto los atributos como los atributos de espacio de nombres son elementos de información `attribute`. En esta Recomendación | Norma Internacional, se utilizan distintos tipos con fines de optimización.

7.12.4 Existen dos tipos de atributos de espacio de nombres en el conjunto de información XML:

- a) declaraciones de espacio de nombres por defecto: la propiedad **[prefix]** del elemento de información **attribute** no tiene valor y la propiedad **[local name]** es "xmlns" ;
- b) declaraciones de espacio de nombres que no vienen dadas por defecto: la propiedad **[prefix]** del elemento de información **attribute** es "xmlns", y la propiedad **[local name]** proporciona el prefijo de la declaración de espacio de nombres.

En ambos casos, la propiedad **[normalized value]** del elemento de información **attribute** proporciona el nombre del espacio de nombres de la declaración de espacio de nombres.

7.12.5 Si el atributo espacio de nombres es una declaración de espacio de nombres por defecto (caso a) de 7.12.4), entonces estará ausente el componente **prefix**, en cualquier otro caso (caso b de 7.12.4) estará presente, representando la propiedad **[local name]** del elemento de información **attribute**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría PREFIX.

7.12.6 Si la propiedad **[normalized value]** del elemento de información **attribute** es una cadena vacía, entonces estará ausente el componente **namespace-name**, en cualquier otro caso estará presente, representando la propiedad **[normalized value]** del elemento de información **attribute**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría NAMESPACE NAME.

7.12.7 La propiedad **[namespace name]** del elemento de información **attribute** es siempre "http://www.w3.org/2000/xmlns/" (véase el Infoset W3C XML) y no está incluida en el tipo **NamespaceAttribute**.

7.13 El tipo **IdentifyingStringOrIndex**

7.13.1 El tipo **IdentifyingStringOrIndex** es:

```
IdentifyingStringOrIndex ::= CHOICE {
    literal-character-string    NonEmptyOctetString,
    string-index                INTEGER (1..one-meg) }
```

7.13.2 El tipo **NonEmptyOctetString** y el valor **one-meg** se definen en 7.2.1.

7.13.3 El tipo **IdentifyingStringOrIndex** representa una cadena de caracteres que incluye información de identificación.

NOTA – Son ejemplos de dichas cadenas de caracteres los prefijos, los nombres de espacios de nombres y los nombres locales de elementos y de atributos.

7.13.4 Un valor abstracto de este tipo ASN.1 tiene una cadena de caracteres (de una categoría dada) como un valor del tipo **NonEmptyOctetString**, o el índice de tabla de vocabulario de una cadena de caracteres (de una categoría dada) en la tabla de vocabulario para dicha categoría de cadena (véase 8.4.2), denominada tabla de cadena aplicable.

NOTA 1 – La categoría de cadena siempre se especifica en el texto asociado a subcláusulas anteriores (véase 7.5 a 7.12) cuando utilice este tipo.

NOTA 2 – Las cadenas de caracteres "identificadoras" se tratan de forma distinta a las cadenas de caracteres "no identificadoras" (véase 7.14). Mientras que una cadena de caracteres no identificadora puede ser codificada en uno de entre muchos formatos de codificación, todas las cadenas de caracteres identificadoras se codifican en UTF-8. Asimismo, mientras que una cadena de caracteres no identificadora puede ser añadida o no (a criterio del creador) a la tabla dinámica de cadenas (véase 7.14.6), las cadenas de caracteres identificadoras siempre se añaden a la tabla dinámica de cadena (véase 7.13.7).

7.13.5 Si **literal-character-string** está presente, incluirá la codificación UTF-8 (véase ISO/CEI 10646, anexo D) de la cadena de caracteres (véase 7.13.4).

7.13.6 Si **string-index** está presente, incluirá el índice de la tabla de vocabulario de cualquiera de las entradas de la tabla de cadenas aplicable que son idénticas a la cadena de caracteres.

7.13.7 Cuando se crea un valor abstracto de este tipo ASN.1 (que representa una cadena de caracteres dada de una categoría dada), si existe una cadena de caracteres idéntica en el contenido actual de la tabla de cadenas aplicable, el creador realizará la acción a) o la acción b) siguientes como opción de implementación (pero debería seleccionarse, si es posible, la primera opción, ya que produce el número más bajo de índices dirigidos a la misma cadena de caracteres), o (si no hay una cadena de caracteres idéntica) el creador realizará la acción b). Las acciones a) y b) son:

- a) seleccionar la alternativa **string-index**, y asignar a **string-index** el índice de tabla de vocabulario de cualquiera de las entradas existentes idénticas a la cadena de caracteres;

- b) seleccionar la alternativa **literal-character-string**, asignar la cadena de caracteres dada a **literal-character-string**, y añadir a la tabla de cadenas aplicable una cadena de caracteres idéntica, salvo que dicha tabla contenga 2^{20} entradas.

NOTA – Si se escoge la acción b), habrá más de una cadena de caracteres idéntica en la tabla de cadenas (si no contiene todavía 2^{20} entradas). Esto no afecta al procesamiento posterior de las cadenas de caracteres (véase 7.13.8).

7.13.8 Cuando se procesa un valor abstracto de este tipo ASN.1 que represente una cadena de caracteres (de una categoría dada), el procesador determinará la cadena de caracteres representada por el valor abstracto de la forma siguiente:

- a) Si existe la alternativa **string-index**, la cadena de caracteres representada por el valor abstracto será la cadena de caracteres del contenido actual de la tabla de cadenas aplicable cuyo índice de tabla de vocabulario es el valor de **string-index**.
- b) Si existe la alternativa **literal-character-string**, la cadena de caracteres representada por el valor abstracto será el valor de **literal-character-string** y se añadirá una cadena de caracteres idéntica a la tabla de cadenas aplicable (son obstante, véase 7.13.9), salvo que la tabla ya contenga 2^{20} entradas.

7.13.9 Si un procesador no puede (por cualquier razón, incluidos las limitaciones propias de la implementación) añadir una cadena a una tabla de vocabulario que contenga menos de 2^{20} entradas cuando dicha adición sea requerida según 7.13.8 b), detendrá el procesamiento del documento infosec rápido y generará un error.

7.14 El tipo **NonIdentifyingStringOrIndex**

7.14.1 El tipo **NonIdentifyingStringOrIndex** es:

```
NonIdentifyingStringOrIndex ::= CHOICE {
    literal-character-string SEQUENCE {
        add-to-table          BOOLEAN,
        character-string      EncodedCharacterString },
    string-index             INTEGER (0..one-meg) }
```

7.14.2 El tipo **EncodedCharacterString** y el valor **one-meg** se definen en 7.17 y en 7.2.1 respectivamente.

7.14.3 El tipo **NonIdentifyingStringOrIndex** representa una cadena de caracteres que no transporta información de identificación.

NOTA – Un ejemplo de dicha cadena de caracteres es el valor de un atributo.

7.14.4 Un valor abstracto de **NonIdentifyingStringOrIndex** tiene una cadena de caracteres (de una categoría dada) como valor del tipo **EncodedCharacterString** (véase 7.17), o bien, el índice de tabla de vocabulario de una cadena de caracteres de una categoría dada en la tabla de vocabulario para dicha categoría de cadena (véase 8.4.2), denominada tabla de cadena aplicable.

NOTA – Cuando se utiliza ese tipo, la categoría de la cadena siempre se especifica en el texto asociado de cláusulas anteriores.

7.14.5 Si **string-index** existe, podrá ser cero (lo que denota una cadena de caracteres de longitud cero – véase 7.14.6) o incluir el índice de tabla de vocabulario de cualquiera de las entradas de la tabla de cadenas aplicable que son idénticas a la cadena de caracteres.

7.14.6 Para una cadena de caracteres de longitud cero, el creador siempre utilizará el **string-index** alternativo con el valor entero cero. Un procesador tratará dicho valor como si representara una cadena de caracteres de longitud cero.

7.14.7 Cuando se crea un valor abstracto del tipo **NonIdentifyingStringOrIndex** (que representa una cadena de caracteres dada o una categoría dada) de longitud distinta de cero, si existe una cadena de caracteres idéntica en el contenido actual de la tabla de cadenas aplicable, el creador realizará la acción a) o la acción b) siguientes como opción de implementación (pero debería seleccionarse, si es posible, la primera opción, ya que produce el número más bajo de índices dirigidos a la misma cadena de caracteres), o (si no hay una cadena de caracteres idéntica) el creador realizará la acción b). Las acciones a) y b) son:

- a) seleccionar la alternativa **string-index**, y asignar a **string-index** el índice de tabla de vocabulario de cualquiera de las entradas existentes idénticas a la cadena de caracteres;
- b) seleccionar la alternativa **literal-character-string**, asignar la cadena de caracteres dada al componente **character-string**; y
- 1) añadir a la tabla de cadenas aplicable una cadena de caracteres idéntica y poner el componente **add-to-table** a VERDADERO (esta acción b1 no se realizará si la tabla de cadenas aplicable ya contiene 2^{20} entradas); o

NOTA 1 – Si la tabla de caracteres aplicable ya contiene 2^{20} entradas, sólo se puede realizar la acción a) o b2.

- 2) poner el componente **add-to-table** a **FALSO**.

NOTA 2 – Si se elige la acción b1 habrá más de una cadena de caracteres idéntica en el contenido actual. Esto no afecta el procesamiento posterior de las cadenas de caracteres (véase 7.14.8).

7.14.8 Cuando se procesa un valor abstracto de este tipo ASN.1 que representa una cadena de caracteres de una categoría dada, el procesador determinará la cadena de caracteres que representa el valor abstracto de la forma siguiente:

- a) Si la alternativa **string-index** está presente, la cadena de caracteres representada por el valor abstracto será la cadena de caracteres de la tabla de cadenas aplicable cuyo índice de tabla de vocabulario sea el valor de **string-index**.
- NOTA 1 – Si **string-index** supera el tamaño actual de dicha tabla de vocabulario, el documento infoset rápido es erróneo.
- b) Si la alternativa **literal-character-string** está presente y el componente **add-to-table** tiene el valor **TRUE**, la cadena de caracteres representada por el valor abstracto será el valor del componente **character-string**. El procesador añadirá a la tabla de cadenas aplicable una cadena de caracteres idéntica (no obstante, véase 7.14.12).
- NOTA 2 – Si la tabla de cadena aplicable ya tiene 2²⁰ cadenas, el documento infoset rápido es erróneo.
- c) Si la alternativa **literal-character-string** está presente y el componente **add-to-table** tiene el valor **FALSE**, la cadena de caracteres representada por el valor abstracto será el valor de componente **character-string**.

7.14.9 Si un procesador es incapaz (por cualquier razón, incluidas las limitaciones propias de la implementación) de añadir una cadena a una tabla de vocabulario cuando dicha adición se requiere de conformidad con 7.14.8 b), detendrá el procesamiento del documento infoset rápido y generará un error.

7.15 El tipo NameSurrogate

7.15.1 El tipo **NameSurrogate** es:

```
NameSurrogate ::= SEQUENCE {
    prefix-string-index          INTEGER(1..one-meg) OPTIONAL,
    namespace-name-string-index INTEGER(1..one-meg) OPTIONAL,
    local-name-string-index     INTEGER(1..one-meg) }
(CONSTRAINED BY {-- prefix-string-index sólo estará presente si
-- existe namespace-name-string-index --})
```

7.15.2 El valor de **one-meg** se define en 7.2.1.

7.15.3 El tipo **NameSurrogate** tiene tres enteros positivos (los dos primeros son opcionales) conformando un sustituto de nombre (véase 8.5.2).

NOTA – Este tipo solo existe en el componente **initial-vocabulary** del tipo **Document**.

7.15.4 El **prefix-string-index** (si existe), **namespace-name-string-index** (si existe), y **local-name-string-index** serán mayores que cero pero no superiores al número de entradas de las tablas **PREFIX**, **NAMESPACE NAME**, y **LOCAL-NAME** del vocabulario inicial, respectivamente.

NOTA – Si el procesador de un documento infoset rápido opta por no verificar que se cumple dicha restricción, los procesamientos posteriores pueden producir situaciones de vulnerabilidad de la seguridad.

7.15.5 El **prefix-string-index** no estará presente salvo que **namespace-name-string-index** esté presente.

7.16 El tipo QualifiedNameOrIndex

7.16.1 El tipo **QualifiedNameOrIndex** es:

```
QualifiedNameOrIndex ::= CHOICE {
    literal-qualified-name SEQUENCE {
        prefix          IdentifyingStringOrIndex OPTIONAL
                        -- categoría PREFIX --,
        namespace-name  IdentifyingStringOrIndex OPTIONAL
                        -- categoría NAMESPACE NAME --,
        local-name     IdentifyingStringOrIndex
                        -- categoría LOCAL NAME -- },
    name-surrogate-index INTEGER (1..one-meg) }
```

7.16.2 El tipo **IdentifyingStringOrIndex** y el valor **one-meg** se definen en 7.13 y 7.2.1 respectivamente.

7.16.3 El tipo **QualifiedNameOrIndex** representa un nombre cualificado (véase 3.4.11).

7.16.4 Un valor abstracto de este tipo ASN.1 incluye tres componentes correspondientes al prefijo, nombre de espacio de nombres y nombre local de un nombre cualificado (de una categoría dada), o bien, el índice de la tabla de vocabulario de un sustituto de nombre de una categoría dada en la tabla de vocabulario para dicha categoría de nombre cualificado (véase 8.5.2), denominada tabla de nombres aplicable.

NOTA – Cuando se utiliza este tipo la categoría siempre se especifica en el texto asociado de cláusulas anteriores.

7.16.5 Si el **name-surrogate-index** está presente, incluirá en la tabla de nombres aplicable el índice de tabla de vocabulario de un sustituto de nombre.

7.16.6 Si el **namespace-name** no está presente, el **prefix** tampoco estará presente.

7.16.7 Cuando se crea un valor abstracto de este tipo ASN.1 representando un nombre cualificado dado (o una categoría dada), el creador tomará las acciones especificadas en las subcláusulas siguientes.

7.16.7.1 Se evaluarán las condiciones siguientes en el orden expuesto:

- a) que el nombre cualificado, o bien no tiene prefijo, o su prefijo existe en el contenido vigente de la tabla **PREFIX**;
- b) que el nombre cualificado, o bien no tenga nombre de espacio de nombres, o que el mismo está incluido en la actual tabla **NAMESPACE NAME**;
- c) que el nombre local del nombre cualificado existe en la actual tabla **LOCAL NAME**;
- d) que se satisfacen las tres primeras condiciones y en el contenido actual de la tabla de nombres aplicable existe un sustituto de nombre (véase 8.5), que consta de índice o índices de tabla de vocabulario del prefijo (si existe), nombre de espacio de nombres (si existe) y nombre local.

7.16.7.2 Si se cumplen todas las condiciones anteriores, se seleccionará la alternativa **name-surrogate-index**, al que se asignará el valor del índice de la tabla de vocabulario del sustituto de nombre determinado en 7.16.7.1 d) en la tabla de nombres aplicable, completando los procedimientos de 7.16.7.

7.16.7.3 En cualquier otro caso, se seleccionará la alternativa **literal-qualified-name**, y sus componentes se asignarán de la forma siguiente:

- a) si el nombre cualificado no tiene un prefijo, el componente **prefix** no estará presente, en cualquier otro caso se asignará al componente **prefix** un prefijo aplicando 7.13.7 con la restricción de que la acción 7.13.7 b) no se realizará si hay una cadena de caracteres idéntica en el contenido actual de la tabla de cadenas aplicable. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría **PREFIX**;
- b) si el nombre cualificado no tiene un nombre de espacio de nombres, el componente **namespace-name** no estará presente, en cualquier otro caso, se asignará al componente **namespace-name** un nombre de espacio de nombres aplicando 7.13.7 con la restricción de que la acción 7.13.7 b) no se realizará si hay una cadena de caracteres idéntica en el contenido actual de la tabla de cadenas aplicable. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría **NAMESPACE NAME** (véase 8.4.2);
- c) el nombre local del nombre cualificado se asignará (aplicando 7.13.7) al componente **local-name**. Este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría **LOCAL NAME**.

NOTA – La aplicación de 7.13.7 en esta subcláusula puede causar la adición del nombre local en la tabla **LOCAL NAME**.

7.16.7.4 Si la aplicación de 7.13.7 en la subcláusula 7.16.7.3 a una o más de las tres cadenas de caracteres anteriores no ha añadido la cadena a una tabla de vocabulario porque dicha tabla ya contenía ²⁰ entradas es (véase 7.13.7 b), entonces no puede crearse un sustituto de nombre para este nombre cualificado.

7.16.7.5 En cualquier otro caso, se crea un sustituto de nombre (véase 8.5), que consta de uno o varios índices de tabla de vocabulario del prefijo (si existe), nombre de espacio de nombres (si existe) y nombre local. Si este sustituto de nombre no existe en el contenido vigente de la tabla de nombres aplicable, se añadirá a dicha tabla, salvo que la tabla ya contenga ²⁰ entradas.

7.16.8 Cuando se procesa un valor abstracto del tipo **QualifiedNameOrIndex** que representa un nombre cualificado de una categoría dada, el procesador determinará el nombre cualificado representado por el valor abstracto de conformidad con las subcláusulas siguientes.

7.16.8.1 Si la alternativa **name-surrogate-index** está presente, el nombre cualificado representado por el valor abstracto será el representado por el sustituto de nombre de la categoría dada (véase 8.5.2) de la tabla de nombres aplicables cuyo índice de tabla de vocabulario sea el valor de **name-surrogate-index**.

7.16.8.2 Si la alternativa **literal-qualified-name** está presente, entonces:

- a) El nombre cualificado representado por el valor abstracto se determinará de la forma siguiente:
 - 1) el prefijo del nombre cualificado se determinará (aplicando 7.13.8) a partir del componente **prefix** (si existe); este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría PREFIX;
 - 2) el nombre del espacio de nombres del nombre cualificado se determinará (aplicando 7.13.8) a partir del componente **namespace-name** (si existe); este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría NAMESPACE NAME (véase 8.4.2);
 - 3) el nombre local del nombre cualificado se determinará (aplicando 7.13.8) a partir del componente **local-name** (si existe); este componente es del tipo **IdentifyingStringOrIndex** (véase 7.13), que en este caso representa una cadena de caracteres de la categoría LOCAL NAME.

NOTA – Estas actuaciones pueden requerir una adición a las correspondientes tablas de vocabulario, tal como se especifica en 7.13.8 b).
- b) Si tras posibles adiciones resultado del procesamiento de los componentes de los **literal-qualified-name**, existen índices de tablas de vocabulario para todos los componentes presentes, se añadirá a la tabla de nombres aplicable un sustituto de nombre que conste de dichos índices de vocabulario (no obstante, véase 7.16.9), salvo que la tabla de vocabulario contenga ya 2^{20} sustitutos de nombres.

7.16.9 Si un procesador no puede (por cualquier motivo, incluidos límites específicos de la implementación) añadir un sustituto de nombre a una tabla de vocabulario que contenga menos de 2^{20} entradas cuando dicha adición sea necesaria conforme a 7.16.8.2 b), detendrá el procesamiento del documento infosef rápido y generará un error.

7.17 El tipo **EncodedCharacterString**

7.17.1 El tipo **EncodedCharacterString** es:

```

EncodedCharacterString ::= SEQUENCE {
    encoding-format      CHOICE {
        utf-8            NULL,
        utf-16           NULL,
        restricted-alphabet  INTEGER(1..256),
        encoding-algorithm  INTEGER(1..256) },
    octets              NonEmptyOctetString }

```

7.17.2 El tipo **EncodedCharacterString** contiene una codificación de una cadena de caracteres. Especifica una cadena de octetos que constituye una correspondencia reversible entre una cadena de caracteres y una cadena de octetos. El creador de un documento infosef rápido especifica en **encoding-format** la codificación utilizada, y el procesador utiliza esta información para decodificar los octetos del componente **octets** en una cadena de caracteres.

7.17.3 El componente **octets** incluirá un valor de cadena de octetos que sea la codificación de la cadena de caracteres especificada por el componente **encoding-format**.

NOTA – En general, pueden utilizarse diversas codificaciones para una cadena de caracteres dada. El creador de un documento infosef rápido puede elegir en base a determinados criterios (por ejemplo, puede tratar de optimizar el tamaño o la velocidad de procesamiento), pero también puede preferir la conveniencia y aplicabilidad universal de UTF-8 o de UTF-16BE. Algunas codificaciones sólo pueden codificar cadenas de caracteres que contengan exclusivamente un subconjunto de los caracteres ISO/CEI 10646.

7.17.4 El formato de codificación **utf-8** puede utilizarse para cualquier cadena de caracteres. Este formato de codificación se aplicará generando una codificación UTF-8 (véase ISO/CEI 10646) de la cadena de caracteres y asignando dicha codificación al componente **octets**.

NOTA – Este formato de codificación es el más adecuado para cadenas de caracteres en las que los caracteres ISO/CEI 10646 de la primera parte del plano multilingüe básico ISO/CEI 10646 sean los más comunes, y cuando ninguno de los restantes formatos de codificación sea aplicable o de mayor utilidad.

7.17.5 El formato de codificación **utf-16** también puede utilizarse para cualquier cadena de caracteres. Este formato de codificación se aplicará generando una codificación UTF-16BE (véase Unicode, 2.6) de la cadena de caracteres y asignando dicha codificación al componente **octets**.

NOTA 1 – Este formato de codificación es el más adecuado para cadenas de caracteres en las que haya presentes una amplia gama de caracteres ISO/CEI 10646, y ninguno de los restantes formatos de codificación se aplicable o de más utilidad.

NOTA 2 – El orden de los octetos de la codificación UTF-16BE se especifica en Unicode, 2.6, siendo el octeto más significativo el situado en primer lugar (el primer octeto de la cadena de octetos).

7.17.6 El formato de codificación **restricted-alphabet** está basado en utilizar un alfabeto restringido, seleccionado de entre los presentes en la tabla de alfabetos restringidos. El componente **restricted-alphabet** incluirá el índice de tabla de vocabulario del alfabeto restringido. Este formato de codificación sólo puede utilizarse para una cadena de caracteres que conste exclusivamente de caracteres del alfabeto restringido de la entrada de tabla de alfabetos restringidos indexada según el componente **restricted-alphabet**. Se aplicará un formato de codificación **restricted-alphabet** tal como se especifica en 7.17.6.1 a 7.17.6.6.

7.17.6.1 Se asignará, en orden, un valor entero (comenzando desde cero) a cada carácter del alfabeto restringido.

7.17.6.2 Cada carácter de la cadena de caracteres se convertirá en un valor entero, que es el entero asignado a dicho carácter en el alfabeto restringido.

7.17.6.3 Cada entero se representará como un número binario entero sin signo en un campo de bits. El tamaño del campo de bits se determinará mediante un valor entero asignado al último carácter del alfabeto restringido. Dicho valor entero se incrementará en 1 para producir un valor entero (por ejemplo, N). El tamaño del campo de bits será el número mínimo de bits necesarios para codificar N como entero sin signo.

NOTA 1 – El incremento es necesario porque un valor con todos unos en el campo de bits se interpreta como el final de la cadena de caracteres, y no puede utilizarse para representar un carácter. Ello significa que si el alfabeto restringido contiene un número de caracteres que sea una potencia exacta de dos, el campo de bits tendrá un bit más de lo esperado.

NOTA 2 – Por ejemplo, si existen 24 caracteres en el alfabeto restringido, cada carácter se codificará en cinco bits, pero si existen 32 caracteres en el alfabeto restringido, cada carácter se codificará con seis bits.

7.17.6.4 Todos estos campos de bits se concatenarán (en orden) en una cadena de bits.

7.17.6.5 Si la longitud de la cadena de bits resultante no es múltiplo de 8 bits, se añadirán bits con valor '1' que hagan que la longitud de la cadena de bits sea múltiplo de 8 bits.

7.17.6.6 La cadena de bits resultante (que ahora sí es múltiplo de 8 bits), reinterpretada como una cadena de octetos, será asignada al componente **octets**.

7.17.7 El formato de codificación **encoding-algorithm** se especifica mediante el algoritmo de codificación (véase 8.3) que constituye la entrada de la tabla en la tabla del algoritmo de codificación cuyo índice de tabla de vocabulario sea el valor del componente **encoding-algorithm**. El índice de la tabla de vocabulario del algoritmo de codificación se asignará al componente **encoding-algorithm**, y la cadena de octetos resultante de la codificación se colocará en el componente **octets**.

8 Construcción y procesamiento de un documento infoset rápido

Un documento infoset rápido hace un uso intensivo de índices de tablas de vocabulario en una serie de tablas que se construyen en diversas etapas de la elaboración y procesamiento de dicho documento. En la subcláusula 8.1 se especifica una ordenación conceptual de los componentes de un valor abstracto del tipo **Document** para asegurar que el creador y los procesadores de documentos infoset rápidos creen tablas de vocabulario idénticas. En las subcláusulas siguientes se especifican las tablas de vocabulario que se construyen y utilizan en la creación y procesamiento de un documento infoset rápido. La representación de dichas tablas en un sistema computacional es función de la implementación y no está normalizado. Una tabla de vocabulario proporciona una correspondencia entre un índice de tabla de vocabulario y la información de un infoset XML (posiblemente de forma indirecta).

NOTA – Las tablas de vocabulario de un documento infoset rápido se crean dinámicamente durante la construcción del propio documento infoset rápido. Se crean de forma dinámica a partir del contenido del documento infoset rápido durante el procesamiento de dicho documento. Nunca se intercambian de otra forma.

8.1 Ordenación conceptual de componentes de un valor abstracto del tipo **Document**

8.1.1 Para asegurar que diferentes implementaciones asignan índices de tablas de vocabulario de la misma forma cuando se construye y se procesa un documento infoset rápido, se especifica un orden conceptual para los componentes de un valor abstracto de tipo **Document**. La construcción y procesamiento de dichas tablas abstractas utilizará este orden conceptual cuando se añadan cadenas (véanse 7.13.7 y 7.14.6) y sustitutos de nombres (véase 7.16.7) a las tablas de vocabulario.

NOTA – Este orden es el mismo que el de las codificaciones de los componentes en un documento infoset rápido. No implica necesariamente que la semántica del documento se procese en dicho orden. El orden se define exclusivamente con el fin de asegurar que para una entrada de tabla de vocabulario dada, el creador y el procesador de un documento infoset rápido asignan el mismo índice de tabla de vocabulario.

8.1.2 El orden conceptual de la construcción y procesamiento de un documento infosef rápido se especifica tal como se indica en los párrafos siguientes: los componentes de un valor abstracto del tipo **Document** se analizan de conformidad con el algoritmo especificado en 8.1.2.1 a 8.1.2.5. El orden en el que se analizan los componentes define el orden conceptual.

8.1.2.1 En primer lugar se analiza el componente de máximo nivel del valor abstracto (correspondiente al tipo **Document**).

8.1.2.2 Si el componente analizado es del tipo secuencia, los componentes de tipo secuencia presentes en el valor abstracto se analizarán en el orden de su definición textual, desde el primer al último componente presentes, aplicando recursivamente las subcláusulas 8.1.2.1 a 8.1.2.5 a cada componente analizado.

8.1.2.3 Si el componente analizado es del tipo secuencia de, todas las ocurrencias del componente secuencia se analizarán en el orden de "secuencia de", de la primera a la última, aplicando recursivamente las subcláusulas 8.1.2.1 a 8.1.2.5 a cada componente analizado.

8.1.2.4 Si el componente analizado es del tipo elección, se analizará la alternativa que esté presente en el valor abstracto, aplicando recursivamente a dicha alternativa las subcláusulas 8.1.2.1 a 8.1.2.5.

8.1.2.5 Si el componente analizado es de cualquier otro tipo ASN.1, no se necesitan actuaciones ulteriores para dicho componente.

8.2 Tabla de alfabetos restringidos

8.2.1 Cada documento infosef rápido tiene una tabla de alfabeto restringido asociada al mismo. La tabla de alfabetos restringidos contiene alfabetos restringidos que pueden referenciarse mediante un índice de tabla de vocabulario.

8.2.2 Cada entrada de la tabla de alfabetos restringidos será un conjunto ordenado de diferentes caracteres ISO/CEI 10646 de cualquier tamaño entre 2 y 2^{20} caracteres.

NOTA – Un alfabeto restringido permite la codificación compacta de cualquier cadena de caracteres que conste exclusivamente de caracteres de dicho conjunto, mediante la asignación progresiva de valores enteros a los caracteres del conjunto y la utilización de dichos enteros para codificar los caracteres de la cadena (véase 7.17.6).

8.3 Tabla de algoritmos de codificación

8.3.1 Cada documento infosef rápido tiene una tabla de algoritmos de codificación asociada. La tabla de algoritmos de codificación contiene definiciones de algoritmos de codificación a los que se puede hacer referencia mediante un índice de tabla de vocabulario.

8.3.2 Cada entrada en dicha tabla especifica la codificación de una cadena de caracteres con algunas características definidas en una cadena de octetos (véase 7.17.7).

NOTA – Las características definidas pueden hacer referencia a la longitud de la cadena, a los caracteres que aparecen en la misma, o a un modelo arbitrariamente complejo de caracteres. En general, un algoritmo de codificación dado solo se aplica a un subconjunto especial y definido de cadena de caracteres ISO/CEI 10646.

8.3.3 Los algoritmos de codificación están sujetos a las restricciones siguientes:

- a) el algoritmo de codificación tendrá un URI asociado, salvo que sea un algoritmo de codificación integrado, de forma que pueda hacerse referencia al mismo en las adiciones a la tabla;
- b) el algoritmo de codificación especificará precisamente a qué tipos de cadena de caracteres se puede aplicar; dicha especificación incluirá un alfabeto restringido (si existiera), una gama de longitudes (si existiera), y cualquier limitación adicional relativa a la longitud y contenido de la cadena de caracteres (tales como un patrón);
- c) para cualquier cadena de caracteres a la que pueda aplicarse, el algoritmo de codificación proporcionará una correspondencia reversible entre dicha cadena de caracteres y una cadena de octetos.

NOTA 1 – Lo anterior implica que no puede haber ninguna cadena de caracteres S para las que un paso de codificación, desde S a E, seguido de un paso de codificación de E a S', resulte en $S' \neq S$, aunque la diferencia entre S y S' sea pequeña (por ejemplo, un carácter ESPACIO extra). Por otro lado, no se requiere que cualquier cadena de caracteres S sea codificable, ni que las codificaciones sean canónicas.

NOTA 2 – No se requiere que una aplicación que cree un documento infosef rápido a partir de datos contenidos en memoria, tales como números en coma flotante, deba generar una representación léxica de dichos datos y que se aplique un algoritmo de codificación a dicha representación. En su lugar, la aplicación puede crear una cadena de octetos directamente a partir de dichos datos, siempre que la cadena de octetos sea tal que pueda haber sido producida aplicando dicho algoritmo de codificación a una cadena de caracteres que represente los datos contenidos en memoria, y sea una cadena de caracteres a la que pueda aplicarse dicho algoritmo de codificación.

NOTA 3 – En otras normas pueden especificarse algoritmos de codificación (distintos a los que están integrados) o bien los algoritmos pueden ser objeto de un acuerdo entre el creador y los procesadores de un documento infosep rápido.

8.4 Tablas de cadenas dinámicas

8.4.1 Cada documento infosep rápido tiene ocho tablas de cadenas dinámicas asociada al mismo. Cada tabla de cadenas dinámica contiene cadenas de caracteres a las que puede hacerse referencia mediante un índice de tabla de vocabulario.

8.4.2 En esta Recomendación | Norma Internacional se clasifican todas las cadenas de caracteres que pueden existir en un documento infosep rápido en las ocho categorías siguientes, cada una de las cuales tiene una tabla de cadenas dinámica:

- a) PREFIX: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[prefix]** de un elemento de información **element**, **attribute** o **namespace**.
- b) NAMESPACE NAME: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[namespace name]** de un elemento de información **element**, **attribute** o **namespace**.
- c) LOCAL NAME: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[local name]** de un elemento de información **element** o **attribute**.
- d) OTHER NCNAME: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[target]** de un elemento de información **processing instruction** o la propiedad **[name]** de un elemento de información **unexpanded entity reference**, **unparsed entity** o **notation** o la propiedad **[notation name]** de un elemento de información **unparsed entity**.
- e) OTHER URI: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[system identifier]** o la propiedad **[public identifier]** de un elemento de información **unexpanded entity reference**, **document type declaration**, **unparsed entity** o **notation**.
- f) ATTRIBUTE VALUE: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[normalized value]** de un elemento de información **attribute**.
- g) CONTENT CHARACTER CHUNK: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[character code]** de un fragmento (*chunk*) de elementos de información **character** que son hijos consecutivos de un elemento de información **element** dado.
- h) OTHER STRING: esta categoría incluye cadenas de caracteres que constituyen la propiedad **[version]** de un elemento de información **document**; o la propiedad **[content]** de un elemento de información **processing instruction** o **comment**.

8.5 Tablas de nombres dinámicas y sustitutos de nombres

8.5.1 Cada documento infosep rápido tiene dos tablas de nombres dinámicas asociadas. Cada tabla dinámica de nombres contiene sustitutos de nombres a los que puede hacerse referencia mediante un índice de tabla de vocabulario y que se utilizan para identificar un nombre cualificado, que puede tener prefijo o no tenerlo (y que puede tener o no un nombre de espacio de nombres).

8.5.2 Un sustituto de nombre es un conjunto de hasta tres índices de tabla de vocabulario ordenados:

- a) (facultativamente) el índice de una cadena en la tabla PREFIX;
- b) (facultativamente) el índice de una cadena en la tabla NAMESPACE NAME; y
- c) el índice de una cadena en la tabla LOCAL NAME.

El primer índice de tabla de vocabulario no estará presente salvo que el segundo lo esté.

8.5.3 Puede darse tres casos:

- a) existen los tres índices, en cuyo caso el sustituto de nombre representa un nombre cualificado con prefijo;
- b) sólo existen el segundo y tercer índice, en cuyo caso el sustituto de nombre representa un nombre cualificado sin prefijo que tiene un nombre de espacio de nombres;
- c) sólo existe el tercer índice, en cuyo caso el sustituto de nombre representa un nombre cualificado sin prefijo que no tiene nombre de espacio de nombres.

8.5.4 En esta Recomendación | Norma Internacional se clasifican todos los nombres cualificados que pueden existir en un documento infosep rápido (y por tanto, también los sustitutos de nombres que les representan) en las dos categorías siguientes, cada una de las cuales tiene una tabla de nombres dinámica:

- a) ELEMENT NAME: esta categoría incluye los sustitutos de nombres que representan al nombre cualificado de un elemento de información **element**.
- b) ATTRIBUTE NAME: esta categoría incluye los sustitutos de nombres que representan al nombre cualificado de un elemento de información **attribute**.

8.5.5 Dada una tabla de cadenas dinámica, el nombre cualificado representado por un sustituto de nombre dado se determina de la forma siguiente:

- a) el primer índice de tabla de vocabulario (si existe) se interpretará como el índice de tabla de vocabulario de una cadena de caracteres de la tabla PREFIX; y
- b) el segundo índice de tabla de vocabulario del (si existe) se interpretará como el índice de tabla de vocabulario de una cadena de caracteres de la tabla NAMESPACE NAME; y
- c) el tercer entero se interpretará como el índice de tabla de vocabulario de una cadena de caracteres de la tabla LOCAL NAME.

9 Alfabetos restringidos integrados

9.1 Alfabeto restringido "numérico"

9.1.1 Este alfabeto restringido tiene un índice de tabla de vocabulario de 1, y consta de los siguientes quince caracteres ISO/CEI 10646 (en este orden):

DÍGITO CERO a DÍGITO NUEVE
 GUIÓN – MENOS
 SIGNO MÁS
 PUNTO
 LETRA MINÚSCULA CURSIVA E
 ESPACIO

9.1.2 Este alfabeto restringido es adecuado para la codificación de cadenas de caracteres que representen varios tipos de números, incluidos números en coma flotante en notación científica. Una única cadena de caracteres puede contener múltiples números separados por espacios.

9.2 Alfabeto restringido "fecha y hora"

9.2.1 Este alfabeto restringido tiene un índice de tabla de vocabulario de 2 y consta de los siguientes quince caracteres ISO/CEI 10646 (en este orden):

DÍGITO CERO a DÍGITO NUEVE
 GUIÓN – MENOS
 DOS PUNTOS
 LETRA MAYÚSCULA CURSIVA T
 LETRA MAYÚSCULA CURSIVA Z
 ESPACIO

9.2.2 Este alfabeto restringido es adecuado para la codificación de cadenas de caracteres que representen las expresiones más comunes de fecha y hora basadas en ISO 8601. Una única cadena de caracteres puede contener varias de dichas expresiones separadas por espacios.

10 Algoritmos de codificación integrados

10.1 Generalidades

10.1.1 En esta cláusula se especifican algoritmos de codificación integrados. Los algoritmos de codificación integrados no tienen un URI asociado.

NOTA – Los URI son necesarios para algoritmos de codificación que han de ser explícitamente identificados en un vocabulario inicial, sin embargo, los algoritmos de codificación integrados siempre se añaden implícitamente a la tabla del algoritmo de codificación y por tanto no necesitan URIs.

ISO/CEI 24824-1:2005 (S)

10.1.2 En esta cláusula el término "palabra" indica cualquier grupo de caracteres consecutivos de una cadena de caracteres dada, que:

- a) no contengan ningún ESPACIO; y
- b) está al comienzo de la cadena de caracteres, o bien viene precedida por un ESPACIO; y
- c) está al final de la cadena de caracteres, o bien está seguida de un ESPACIO.

NOTA – Una "palabra" no está restringida a estar formada por caracteres alfabéticos.

10.2 Algoritmo de codificación "hexadecimal"

10.2.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 1, y sólo puede aplicarse a una cadena de caracteres que conste de los siguientes dieciséis caracteres ISO/CEI 10646:

de DÍGITO CERO hasta DÍGITO NUEVE

de LETRA MAYÚSCULA CURSIVA A hasta LETRA MAYÚSCULA CURSIVA F

que contiene un número par de caracteres (incluyendo cero).

NOTA – No se permite incluir ningún espacio en blanco XML.

10.2.2 La cadena de caracteres se interpretará como la codificación hexadecimal de una cadena de octetos, donde el primer carácter de la cadena se corresponde con los cuatro bits más significativos del primer octeto, y así sucesivamente.

10.3 Algoritmo de codificación "base64"

10.3.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 2, y sólo puede aplicarse a una cadena de caracteres que:

- a) consta exclusivamente de caracteres desde la LETRA CURSIVA MAYÚSCULA A hasta la LETRA CURSIVA MAYÚSCULA Z, LETRA CURSIVA MINÚSCULA A hasta LETRA CURSIVA MINÚSCULA Z, del DÍGITO CERO al DÍGITO NUEVE, SIGNO MÁS, BARRA OBLÍCUA y el SIGNO IGUAL; y

NOTA – No permite espacios en blanco XML en la cadena de caracteres.

- b) sea un ejemplar válido de la codificación de transferencia de contenido especificado en IETF RFC 2045, 6.8, o que pueda convertirse en un ejemplar válido de dicha codificación mediante la inserción de un espacio en blanco XML siempre que lo requiera IETF RFC 2045.

10.3.2 La cadena de caracteres se interpretará como codificación Base64 (véase IETF RFC 2045) de una cadena de octetos (asumiendo que existe un espacio en blanco adicional XML siempre que lo requiera IETF RFC 2045). La cadena de octetos resultante es la especificada por dicha codificación Base64.

10.3.3 Este algoritmo de codificación es adecuado para cadenas de caracteres de codificación que no contengan espacios en blanco XML, y son cadenas Base64 (de cualquier longitud) o pueden convertirse en cadenas Base64 mediante la adición de un espacio en blanco XML.

10.4 Algoritmo de codificación "corto" ("short")

10.4.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 3, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres consta exclusivamente de caracteres del DÍGITO CERO al DÍGITO NUEVE, GUIÓN-MENOS, y ESPACIO; y
- b) ni el primer ni el último carácter de la cadena de caracteres es un ESPACIO, y no existen dos caracteres ESPACIO consecutivos; y
- c) la cadena de caracteres incluye al menos una palabra (véase 10.1.2); y
- d) cada GUIÓN-MENOS presente es el primer carácter de una palabra; y
- e) cada GUIÓN-MENOS está seguido de, al menos, un carácter DÍGITO CERO a DÍGITO NUEVE; y
- f) cada DÍGITO CERO es el único carácter de una palabra, o bien, está precedido de un carácter de entre DÍGITO CERO a DÍGITO NUEVE; y
- g) si cada palabra de la cadena de caracteres se interpreta como una cadena de caracteres numéricos en base 10 enteros y con signo, representa un valor comprendido entre -32768 y 32767.

10.4.2 Cada palabra (véase 10.1.2) de la cadena de caracteres se interpretará como una cadena de caracteres numéricos en base 10 enteros y con signo, y se representará como un entero complemento a dos de 16 bits.

10.4.3 Cada grupo de 8 bits del entero complemento a dos de 16 bits de una palabra, produce un octeto de la cadena de octetos, comenzando por el grupo de 8 bits de orden superior. El bit de orden superior de cada grupo de 8 bits se convertirá en el bit más significativo del octeto correspondiente. Si existen varias palabras en la cadena de caracteres, se codificarán en orden, y los octetos producidos por los varios enteros complemento a dos de 16 bits se concatenarán en dicho orden.

10.4.4 Este algoritmo de codificación es adecuado para la codificación de cadenas de caracteres que representen un único entero del rango -32768 a 32767 (representable como un entero complemento a dos de 16 bits) o una lista de dichos valores enteros.

10.5 Algoritmo de codificación "int"

10.5.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 4, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres cumple las condiciones especificadas en 10.4.1 a) hasta f); y
- b) cada palabra (véase 10.1.2) de la cadena de caracteres, se interpreta como una cadena de caracteres numéricos en base 10 enteros y con signo, representa un valor comprendido entre -2147483648 y 2147483647 .

10.5.2 Cada palabra (véase 10.1.2) de la cadena de caracteres se interpretará como una cadena de caracteres numéricos en base 10 enteros y con signo, y se representará como un entero complemento a dos de 32 bits.

10.5.3 Cada grupo de 8 bits del entero complemento a dos de 32 bits de una palabra producirá un octeto de la cadena de octetos, comenzando por el grupo de 8 bits de orden superior. El bit de orden superior de cada grupo de 8 bits se convierte en el bit más significativo del correspondiente octeto. Si existen varias palabras en la cadena de caracteres, se codificarán en orden, y los octetos producidos por varios enteros complemento a dos de 32 bits se concatenarán en dicho orden.

10.5.4 Este algoritmo de codificación es adecuado para codificar cadenas de caracteres que representen un único entero del rango -2147483648 a 2147483647 (representable como un entero complemento a dos de 32 bits) o una lista de dichos enteros.

10.6 Algoritmo de codificación "largo" ("long")

10.6.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 5, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres cumple las condiciones especificadas en 10.4.1 a) hasta f); y
- b) cada palabra (véase 10.1.2) de la cadena de caracteres, interpretada como una cadena de caracteres numéricos en base 10 enteros y con signo, representa un valor comprendido entre -9223372036854775808 y 9223372036854775807 .

10.6.2 Cada palabra (véase 10.1.2) de la cadena de caracteres se interpretará como una cadena de caracteres numéricos en base 10 enteros y con signo, y se representará como un entero complemento a dos de 64 bits.

10.6.3 Cada grupo de 8 bits del entero complemento a dos de 64 bits de una palabra producirá un octeto de la cadena de octetos, comenzando por el grupo de 8 bits de orden superior. El bit de orden superior de cada grupo de 8 bits se convertirá en el bit más significativo del correspondiente octeto. Si existen varias palabras en la cadena de caracteres, se codificarán en orden, y los octetos producidos por múltiples enteros complemento a dos de 64 bits se concatenarán en dicho orden.

10.6.4 Este algoritmo de codificación es adecuado para codificar cadenas de caracteres que representen a un entero del rango -9223372036854775808 a 9223372036854775807 (representable como un entero complemento a dos de 64 bits) o una lista de dichos valores enteros.

10.7 Algoritmo de codificación "booleano" ("boolean")

10.7.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 6, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres consta exclusivamente de una o más palabras "false" (falso) o de la palabra "true" (verdadero), y del carácter ESPACIO; y

- b) ni el primer ni último carácter de la cadena de caracteres es un ESPACIO, y no existen dos caracteres ESPACIO consecutivos; y
- c) la cadena de caracteres contiene al menos una palabra (véase 10.1.2).

10.7.2 Cada palabra "false" (falso) o "true" (verdadero) de la cadena de caracteres se codificará como un único bit (puesto a cero o a uno, respectivamente) de la cadena de octetos producida, comenzando por el quinto bit del primer octeto hasta el octavo bit del primer octeto. Los bits siguientes se sitúan en octetos subsiguientes desde el primer bit de cada octeto hasta el octavo bit del mismo, utilizando solamente los octetos que sean necesarios. Cualquier bit no utilizado del último octeto se pondrá a cero.

10.7.3 Los primeros cuatro bits del primer octeto contendrán el número de bits no utilizados del último octeto, codificado como un entero sin signo de 4 bits.

NOTA – El primer octeto puede también ser el último octeto y puede contener hasta tres bits no utilizados. Si existe más de un octeto, el último octeto puede contener hasta siete bits no utilizados.

10.8 Algoritmo de codificación "flotante" ("float")

10.8.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 7, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres consta exclusivamente de caracteres DÍGITO CERO a DÍGITO NUEVE, GUIÓN-MENOS, PUNTO, LETRA MAYÚSCULA CURSIVA E, y ESPACIO; y
NOTA – La LETRA MINÚSCULA CURSIVA E no está permitida, pues en ese caso la codificación no sería reversible.
- b) ni el primero ni el último de los caracteres de la cadena de caracteres es un ESPACIO, y no existen dos caracteres ESPACIO consecutivos; y
- c) la cadena de caracteres contiene al menos una palabra (véase 10.1.2); y
- d) cada palabra de la cadena de caracteres concuerda con la representación léxica canónica flotante tal como se especifica en un esquema W3C XML, Parte 2, 3.2.4; y
- e) cada palabra de la cadena de caracteres, si se representa como una cadena de caracteres numéricos base 10 en coma flotante, genera un valor que puede representarse como un valor flotante de 32 bits IEEE 754.

10.8.2 Cada palabra (véase 10.1.2) de la cadena de caracteres se interpretará como una cadena de caracteres numéricos base 10 en coma flotante y se representará como un valor flotante de 32 bits IEEE 754.

10.8.3 Cada grupo de 8 bits del valor flotante de 32 bits IEEE 754 de una palabra generará un octeto de la cadena de octetos, comenzando por el grupo inicial de 8 bits. El bit inicial de cada grupo de 8 bits será el bit más significativo del correspondiente octeto. Si la cadena de caracteres consta de varias palabras, se codificarán en orden, y los octetos producidos por los múltiples valores flotante de 32 bits IEEE 754 se concatenarán en dicho orden.

10.8.4 Este algoritmo de codificación es adecuado para la codificación de cadenas de caracteres que representan un único número en coma flotante representable como un valor flotante de 32 bits IEEE 754 o como una lista de dichos números en coma flotante.

10.9 Algoritmo de codificación "doble" ("double")

10.9.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 8, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres satisface las condiciones especificadas en 10.8.1 a) hasta c);
- b) cada palabra de la cadena de caracteres concuerda con la representación canónica léxica de un doble, tal como se especifica en un esquema W3C XML, Parte 2, 3.2.5; y
- c) cada palabra (véase 10.1.2) de la cadena de caracteres, si se representa como una cadena de caracteres numéricos base 10 en coma flotante, genera un valor que puede representarse como un valor doble de 64 bits IEEE 754.

10.9.2 Cada palabra (véase 10.1.2) de la cadena de caracteres se interpretará como una cadena de caracteres numéricos base 10 en coma flotante y se representará como un valor doble de 64 bits IEEE 754.

10.9.3 Cada grupo de 8 bits del valor flotante de 64 bits IEEE 754 de una palabra generará un octeto de la cadena de octetos, comenzando por el grupo inicial de 8 bits. El bit inicial de cada grupo de 8 bits será el bit más significativo del correspondiente octeto. Si la cadena de caracteres consta de varias palabras, se codificarán en orden, y se concatenarán los octetos producidos por los múltiples valores flotante de 64 bits IEEE 754.

10.9.4 Este algoritmo de codificación es adecuado para la codificación de cadenas de caracteres que representan un único número en coma flotante representable como un valor doble de 64 bits IEEE 754 o como una lista de dichos números en coma flotante.

10.10 Algoritmo de codificación "uuid"

10.10.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 9, y solo puede aplicarse a una cadena de caracteres que cumpla todas las condiciones siguientes:

- a) la cadena de caracteres consta exclusivamente de los caracteres DÍGITO CERO a DÍGITO NUEVE, LETRA MINÚSCULA CURSIVA A hasta LETRA MINÚSCULA CURSIVA F, GUIÓN-MENOS y ESPACIO; y
- b) ni el primero ni el último de los caracteres de la cadena de caracteres es un ESPACIO, y no existen dos caracteres ESPACIO consecutivos; y
- c) la cadena de caracteres contiene al menos una palabra (véase 10.1.2);
- d) cada palabra contiene exactamente 36 caracteres; y
- e) en cada palabra existen exactamente cuatro caracteres GUIÓN-MENOS, que ocupan las posiciones 9, 14, 19 y 24 (contando desde uno).

10.10.2 Cada palabra (véase 10.1.2) de la cadena de caracteres se interpretará como la representación hexadecimal de un UUID (véase Rec. UIT-T X.667 | ISO/CEI 9834-8, 6.4), y se representará como un entero sin signo de 16 octetos, tal como se especifica en la Rec. UIT-T X.667 | ISO/CEI 9834-8, 6.3. Si existen varias palabras, los enteros sin signo de 16 octetos estarán concatenados.

10.10.3 Este algoritmo de codificación es adecuado para la codificación de cadenas de caracteres que representan un único UUID o una lista de UUIDs.

10.11 Algoritmo de codificación "cdata"

10.11.1 Este algoritmo de codificación tiene un índice de tabla de vocabulario de 10, y puede aplicarse a cualquier cadena de caracteres.

10.11.2 La cadena de octetos producida será la codificación UTF-8 (véase ISO/CEI 10646) de la cadena de caracteres.

10.11.3 Este algoritmo solo se utilizará con infosets XML creados analizando sintácticamente un documento XML, en el que la información adicional identifica que la cadena de caracteres corresponde a una sección CDATA completa (véase W3C XML 1.0 y W3C XML 1.1). Si este algoritmo de codificación se utiliza en un documento infoset rápido, a todas las cadenas de caracteres correspondientes a secciones CDATA se les aplicará dicho algoritmo.

11 Restricciones de los infoset XML soportados y otras simplificaciones

11.1 Esta Recomendación | Norma Internacional soporta la mayoría de los infoset XML que es previsible encontrar en la práctica, pero no soporta algunos infoset XML que son teóricamente posibles, aunque rara vez aparecen.

11.2 En esta cláusula se utiliza el término "auto consistente XML" ("*XML-self-consistent*") con el significado siguiente: un conjunto de propiedades de uno o más elementos de información es "auto consistente XML" si dicho conjunto de propiedades pueden haberse obtenido mediante el análisis semántico de un documento XML bien constituido de espacios de nombres.

11.3 Los infoset XML soportados cumplirán todas las condiciones siguientes:

- a) la propiedad **[all declarations processed]** del elemento de información **document** tiene el valor **true**;
- b) la propiedad **[in-scope namespaces]** de cada elemento de información **element** junto con la propiedad **[namespace attributes]** de todos los elementos de información **element** forman un conjunto auto consistente XML;
- c) la propiedad **[namespace name]** de cada elemento de información **element** junto con la propiedad **[namespace attributes]** de todos los elementos de información **element** y la propiedad **[prefix]** de dicho elemento de información **element** forman un conjunto auto consistente XML;
- d) la propiedad **[namespace name]** de cada elemento de información **attribute** junto con la propiedad **[namespace attributes]** de todos los elementos de información **element** y la propiedad **[prefix]** de dicho elemento de información **attribute** forman un conjunto auto consistente XML;

- e) la propiedad **[references]** de cada elemento de información **attribute** junto con la propiedad **[normalized value]** del elemento de información **attribute** forman un conjunto auto consistente XML;
- f) la propiedad **[notation]** de cada elemento de información **processing instruction** junto con la propiedad **[target]** del elemento de información **processing instruction** y la propiedad **[notations]** del elemento de información **document** forman un conjunto auto consistente XML;
- g) la propiedad **[notation]** de cada elemento de información **unparsed entity** junto con la propiedad **[notation name]** del elemento de información **unparsed entity** y la propiedad **[notations]** del elemento de información **document** forman un conjunto auto consistente XML;
- h) la propiedad **[element content whitespace]** de todos los elementos de información **character** que no representan espacios en blanco tiene el valor **false**;
- i) la propiedad **[element content whitespace]** de cada elemento de información **character** junto con la propiedad **[character code]** del elemento de información **character** forman un conjunto auto consistente XML;
- j) la propiedad **[normalized value]** de todos los elementos de información **attribute** y la propiedad **[content]** de todos los elementos de información instrucción **comment** y **processing** contienen como máximo 2³² caracteres.

11.4 Las propiedades siguientes de los elementos de información del conjunto de información XML no están incluidas en los tipos ASN.1 que representan a dichos elementos de información:

- a) las propiedades **[document element]**, **[base URI]** y **[all declarations processed]** del elemento de información **document** (véanse 7.2.30, 7.2.31 y 7.2.32);
- b) las propiedades **[in-scope namespaces]**, **[base URI]** y **[parent]** del elemento de información **element** (véanse 7.3.8, 7.3.9 y 7.3.10);
- c) las propiedades **[specified]**, **[attribute type]**, **[references]** y **[owner element]** del elemento de información **attribute** (véanse 7.4.7, 7.4.8, 7.4.9 y 7.4.10);
- d) las propiedades **[notation]** y **[parent]** del elemento de información **processing instruction** (véanse 7.5.7 y 7.5.8);
- e) las propiedades **[declaration base URI]** y **[parent]** del elemento de información **unexpanded entity reference** (véanse 7.6.7 y 7.6.8);
- f) la propiedad **[element content whitespace]** del elemento de información **character** (véase 7.7.7);
- g) la propiedad **[parent]** del elemento de información **character** (véase 7.7.8);
- h) la propiedad **[parent]** del elemento de información **comment** (véase 7.8.6);
- i) la propiedad **[parent]** del elemento de información **document type declaration** (véase 7.9.7);
- j) las propiedades **[declaration base URI]** y **[notation]** del elemento de información **unparsed entity** (véanse 7.10.8 y 7.10.9);
- k) la propiedad **[declaration base URI]** del elemento de información **notation** (véase 7.11.7).

12 Codificación a nivel de bit del tipo Document

12.1 En esta cláusula se especifican codificaciones especiales del tipo **Document** que permitan conformar un documento infoset rápido.

NOTA – Estas codificaciones especiales están diseñadas para optimizar la velocidad de procesamiento y la compactibilidad, que se consideran críticos en muchos de los usos previsible de esta Recomendación | Norma Internacional.

12.2 Las codificaciones se especifican en términos de acciones que debe realizar un codificador, con el resultado de la adición de bits a un flujo de bits. El flujo de bits inicial está vacío o es una declaración XML (véase 12.3).

12.3 Una declaración XML (véase W3C XML 1.1, 2.8) puede (a voluntad del creador) estar incluida al comienzo del flujo de bits. La declaración XML (si existe) será una de las cadenas de caracteres siguiente, codificada en UTF-8:

- 1) `<?xml encoding='finf'?>`
- 2) `<?xml encoding='finf' standalone='yes'?>`
- 3) `<?xml encoding='finf' standalone='no'?>`
- 4) `<?xml version='1.0' encoding='finf'?>`
- 5) `<?xml version='1.0' encoding='finf' standalone='yes'?>`
- 6) `<?xml version='1.0' encoding='finf' standalone='no'?>`

- 7) `<?xml version='1.1' encoding='finf'?>`
- 8) `<?xml version='1.1' encoding='finf' standalone='yes'?>`
- 9) `<?xml version='1.1' encoding='finf' standalone='no'?>`

12.4 El número de versión (si existe) de la declaración XML tomará el valor de la propiedad **[version]** correspondiente del elemento de información **document**. La declaración XML no incluirá un número de versión si la propiedad **[version]** no tienen un valor.

12.5 La declaración autónoma (si existe) incluida en la declaración XML tomará el valor de la propiedad **[standalone]** correspondiente del elemento de información **document**. La declaración XML no incluirá una declaración autónoma si la propiedad **[standalone]** no tienen un valor.

12.6 Se añaden al flujo de los bits dieciséis bits '1110000000000000'.

NOTA – Estos bits están presentes al comienzo del documento infoset rápido o tras la declaración XML. Si no existe declaración XML, un analizador sintáctico puede distinguir analizando los primeros 16 bits de una codificación, un potencial documento infoset rápido de cualquier documento W3C XML 1.0 o W3C XML 1.1 bien constituido, porque dichos 16 bits nunca deben estar presentes al comienzo de un documento XML bien constituido.

12.7 Se añade al flujo de bits un campo con bits de dieciséis bits que incluye el número de versión de esta Recomendación | Norma Internacional (véase 12.9) codificado como un entero sin signo de 16 bits.

12.8 Se añade el bit '0' (relleno) al flujo de bits.

NOTA – Esto se hace para asegurar la alineación de bits en las partes finales de la codificación.

12.9 El número de versión de la presente edición de esta Recomendación | Norma Internacional es 1.

NOTA – Las ediciones ulteriores de esta Recomendación | Norma Internacional incrementarán el número de versión si fuera previsible la existencia de problemas de interfuncionamiento entre la nueva edición y las ediciones previas.

12.10 La codificación ECN (véase Rec. UIT-T X.692 | ISO/CEI 8825-3) del valor abstracto del tipo **Document**, especificado mediante el módulo de enlace de codificación en A.2, se añadirá al flujo de bits.

NOTA – El anexo C proporciona una descripción informal de las codificaciones especificadas en A.2.

12.11 Si la codificación del valor abstracto del tipo **Document** no termina en el último bit de un octeto, se añadirán cuatro bits '0000' (relleno) al flujo de bits, completando así el último octeto.

12.12 Una vez realizados los pasos descritos, el contenido del flujo de bits es un documento infoset rápido.

Anexo A

Módulo ASN.1 y módulos ECN para documentos infoset rápidos
(Este anexo es parte integrante de la presente Recomendación | Norma Internacional)

A.1 Definición del módulo ASN.1

```

FastInfoset {joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoset(0)
modules(0) fast-infoset(0)}

DEFINITIONS AUTOMATIC TAGS ::= BEGIN

finf-doc-opt-decl OBJECT IDENTIFIER ::= {joint-iso-itu-t(2) asn1(1)
generic-applications(10) fast-infoset(0) encodings(1)
optional-xml-declaration(0)}

finf-doc-no-decl OBJECT IDENTIFIER ::= {joint-iso-itu-t(2) asn1(1)
generic-applications(10) fast-infoset(0) encodings(1)
no-xml-declaration(1)}

Document ::= SEQUENCE {
    additional-data          SEQUENCE (SIZE(1..one-meg)) OF
        additional-datum SEQUENCE {
            id                URI,
            data              NonEmptyOctetString } OPTIONAL,
    initial-vocabulary      SEQUENCE {
        external-vocabulary  URI OPTIONAL,
        restricted-alphabets SEQUENCE (SIZE(1..256)) OF
            NonEmptyOctetString OPTIONAL,
        encoding-algorithms SEQUENCE (SIZE(1..256)) OF
            NonEmptyOctetString OPTIONAL,
        prefixes             SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        namespace-names     SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        local-names         SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        other-ncnames       SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        other-uris          SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        attribute-values    SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
        content-character-chunks SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
        other-strings       SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
        element-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
            NameSurrogate OPTIONAL,
        attribute-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
            NameSurrogate OPTIONAL }
        (CONSTRAINED BY {
            -- If the initial-vocabulary component is present, at least
            -- one of its components shall be present -- }) OPTIONAL,
    notations               SEQUENCE (SIZE(1..MAX)) OF
        Notation OPTIONAL,
    unparsed-entities       SEQUENCE (SIZE(1..MAX)) OF
        UnparsedEntity OPTIONAL,
    character-encoding-scheme NonEmptyOctetString OPTIONAL,
    standalone              BOOLEAN OPTIONAL,
    version                 NonIdentifyingStringOrIndex OPTIONAL
        -- OTHER STRING category --,
    children                SEQUENCE (SIZE(0..MAX)) OF
        CHOICE {
            element                Element,
            processing-instruction ProcessingInstruction,
            comment                 Comment,
            document-type-declaration DocumentTypeDeclaration }}

```

```

one-meg INTEGER ::= 1048576 -- Two to the power 20
four-gig INTEGER ::= 4294967296 -- Two to the power 32
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
URI ::= NonEmptyOctetString

Element ::= SEQUENCE {
    namespace-attributes SEQUENCE (SIZE(1..MAX)) OF
        NamespaceAttribute OPTIONAL,
    qualified-name QualifiedNameOrIndex
        -- ELEMENT NAME category --,
    attributes SEQUENCE (SIZE(1..MAX)) OF
        Attribute OPTIONAL,
    children SEQUENCE (SIZE(0..MAX)) OF
        CHOICE {
            element Element,
            processing-instruction ProcessingInstruction,
            unexpanded-entity-reference UnexpandedEntityReference,
            character-chunk CharacterChunk,
            comment Comment }}

Attribute ::= SEQUENCE {
    qualified-name QualifiedNameOrIndex
        -- ATTRIBUTE NAME category --,
    normalized-value NonIdentifyingStringOrIndex
        -- ATTRIBUTE VALUE category -- }

ProcessingInstruction ::= SEQUENCE {
    target IdentifyingStringOrIndex
        -- OTHER NCNAME category --,
    content NonIdentifyingStringOrIndex
        -- OTHER STRING category -- }

UnexpandedEntityReference ::= SEQUENCE {
    name IdentifyingStringOrIndex
        -- OTHER NCNAME category --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category -- }

CharacterChunk ::= SEQUENCE {
    character-codes NonIdentifyingStringOrIndex
        -- CONTENT CHARACTER CHUNK category -- }

Comment ::= SEQUENCE {
    content NonIdentifyingStringOrIndex -- OTHER STRING category --}

DocumentTypeDeclaration ::= SEQUENCE {
    system-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category --,
    children SEQUENCE (SIZE(0..MAX)) OF
        ProcessingInstruction }

UnparsedEntity ::= SEQUENCE {
    name IdentifyingStringOrIndex
        -- OTHER NCNAME category --,
    system-identifier IdentifyingStringOrIndex
        -- OTHER URI category --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category --,
    notation-name IdentifyingStringOrIndex
        -- OTHER NCNAME category -- }

Notation ::= SEQUENCE {
    name IdentifyingStringOrIndex
        -- OTHER NCNAME category --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
        -- OTHER URI category -- }

```

```

NamespaceAttribute ::= SEQUENCE {
    prefix          IdentifyingStringOrIndex OPTIONAL
                    -- PREFIX category --,
    namespace-name  IdentifyingStringOrIndex OPTIONAL
                    -- NAMESPACE NAME category -- }

IdentifyingStringOrIndex ::= CHOICE {
    literal-character-string  NonEmptyOctetString,
    string-index              INTEGER (1..one-meg) }

NonIdentifyingStringOrIndex ::= CHOICE {
    literal-character-string  SEQUENCE {
        add-to-table          BOOLEAN,
        character-string      EncodedCharacterString },
    string-index              INTEGER (0..one-meg) }

NameSurrogate ::= SEQUENCE {
    prefix-string-index      INTEGER(1..one-meg) OPTIONAL,
    namespace-name-string-index  INTEGER(1..one-meg) OPTIONAL,
    local-name-string-index  INTEGER(1..one-meg) }
    (CONSTRAINED BY {-- prefix-string-index shall only be present if
                    -- namespace-name-string-index is present --})

QualifiedNameOrIndex ::= CHOICE {
    literal-qualified-name SEQUENCE {
        prefix          IdentifyingStringOrIndex OPTIONAL
                        -- PREFIX category --,
        namespace-name  IdentifyingStringOrIndex OPTIONAL
                        -- NAMESPACE NAME category --,
        local-name      IdentifyingStringOrIndex
                        -- LOCAL NAME category -- },
    name-surrogate-index  INTEGER (1..one-meg) }

EncodedCharacterString ::= SEQUENCE {
    encoding-format  CHOICE {
        utf-8          NULL,
        utf-16         NULL,
        restricted-alphabet  INTEGER(1..256),
        encoding-algorithm  INTEGER(1..256) },
    octets           NonEmptyOctetString }

END

```

A.2 Definiciones del módulo ECN

```

FastInfoSetEDM {joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoSet(0)
modules(0) fast-infoSet-edm(1)}
ENCODING-DEFINITIONS ::= BEGIN
EXPORTS FastInfoSetEncodingSet;
RENAMES
    #INTEGER AS #PositiveOrNonNegativeInteger
    IN #IdentifyingStringOrIndex.string-index,
    #NonIdentifyingStringOrIndex.string-index,
    #QualifiedNameOrIndex.name-surrogate-index,
    #NameSurrogate.namespace-name-string-index,
    #NameSurrogate.prefix-string-index,
    #NameSurrogate.local-name-string-index
    FROM FastInfoSet;

/* RENAMES automatically imports:
#Document, #NonEmptyOctetString, #NameSurrogate, #ProcessingInstruction,
#UnexpandedEntityReference, #Comment, #DocumentTypeDeclaration,
#UnparsedEntity, #Notation, #Element, #Attribute, #CharacterChunk,
#NamespaceAttribute, #IdentifyingStringOrIndex, #NonIdentifyingStringOrIndex,
#QualifiedNameOrIndex, #EncodedCharacterString FROM FastInfoSet;
*/

-- Useful encoding classes
#PositiveOrNonNegativeInteger ::= #INTEGER

#NonEmptySequenceOfLength ::= #INT(1..1048576)

#NonEmptyOctetStringLength ::= #INT(1..4294967296)

```

```

#TwoAlternativeDiscriminant ::= #INT(0..1)
#ThreeAlternativeDiscriminant ::= #INT(0..2)
#FourAlternativeDiscriminant ::= #INT(0..3)
#FiveAlternativeDiscriminant ::= #INT(0..4)
-- Used when encoding the length of a SEQUENCE OF (see C.21)
#NonEmptySequenceOfLengthAlternatives1 ::= #ALTERNATIVES {
    small      #INT(1..128),
    large      #INT(129..1048576) }
-- Used when encoding the length of a NonEmptyOctetString (see C.22)
#NonEmptyOctetStringLengthAlternatives2 ::= #ALTERNATIVES {
    small      #INT(1..64),
    medium     #INT(65..320),
    large      #INT(321..4294967296) }
-- Used when encoding the length of a NonEmptyOctetString (see C.23)
#NonEmptyOctetStringLengthAlternatives5 ::= #ALTERNATIVES {
    small      #INT(1..8),
    medium     #INT(9..264),
    large      #INT(265..4294967296) }
-- Used when encoding the length of a NonEmptyOctetString (see C.24)
#NonEmptyOctetStringLengthAlternatives7 ::= #ALTERNATIVES {
    small      #INT(1..2),
    medium     #INT(3..258),
    large      #INT(259..4294967296) }
-- Used when encoding a positive integer (see C.25)
#PositiveIntegerAlternatives2 ::= #ALTERNATIVES {
    small      #INT(1..64),
    medium     #INT(65..8256),
    large      #INT(8257..1048576) }
-- Used when encoding a positive integer (see C.27)
#PositiveIntegerAlternatives3 ::= #ALTERNATIVES {
    small      #INT(1..32),
    medium     #INT(33..2080),
    medium-large #INT(2081..526368),
    large      #INT(526369..1048576) }
-- Used when encoding a positive integer (see C.28)
#PositiveIntegerAlternatives4 ::= #ALTERNATIVES {
    small      #INT(1..16),
    medium     #INT(17..1040),
    medium-large #INT(1041..263184),
    large      #INT(263185..1048576) }
-- Used when encoding a non-negative integer (see C.26)
#NonNegativeIntegerAlternatives2 ::= #ALTERNATIVES {
    zero       #INT(0),
    small      #INT(1..64),
    medium     #INT(65..8256),
    large      #INT(8257..1048576) }
-- Used to insert pre-padding before an encoding in many cases
#PrecededByPrepadding{<#C>} ::= #CONCATENATION {
    prepadding  #PAD,
    original   #C }
-- Used to insert a two-alternative discriminant before an encoding
#PrecededByTwoAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    discriminant #TwoAlternativeDiscriminant,
    original     #C }

```

```

-- Used to insert a three-alternative discriminant before an encoding
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    discriminant    #ThreeAlternativeDiscriminant,
    original        #C }

-- Used to insert a four-alternative discriminant before an encoding
#PrecededByFourAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    prepadding      #PAD,
    discriminant    #FourAlternativeDiscriminant,
    original        #C }

-- Used to insert a five-alternative discriminant before an encoding
#PrecededByFiveAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    prepadding      #PAD,
    discriminant    #FiveAlternativeDiscriminant,
    original        #C }

-- Used to insert a length field before the encoding of a SEQUENCE OF
#PrecededByNonEmptySequenceOfLength{<#C>} ::= #CONCATENATION {
    length          #NonEmptySequenceOfLength,
    original        #C }

-- Used to insert a length field before the encoding of a NonEmptyOctetString
#PrecededByNonEmptyOctetStringLength{<#C>} ::= #CONCATENATION {
    length          #NonEmptyOctetStringLength,
    original        #C }

-- Encodes an item of the notations component of the Document
-- type (see C.2.6.1)
eNotationDriver1 #Notation ::= {
    ENCODE STRUCTURE {
        STRUCTURED WITH eNotationPrepaddingAdder1 }
    WITH FastInfoSetEncodingSet }

-- Encodes an item of the unparsed-entities component of the Document
-- type (see C.2.7.1)
eUnparsedEntityDriver1 #UnparsedEntity ::= {
    ENCODE STRUCTURE {
        STRUCTURED WITH eUnparsedEntityPrepaddingAdder1 }
    WITH FastInfoSetEncodingSet }

-- Encodes an item of the namespace-attributes component of the Element
-- type (see C.3.4.2)
eNamespaceAttributeDriver1 #NamespaceAttribute ::= {
    ENCODE STRUCTURE {
        STRUCTURED WITH eNamespaceAttributePrepaddingAdder1 }
    WITH FastInfoSetEncodingSet }

-- Encodes an item of the attributes component of the Element
-- type (see C.3.6.1)
eAttributeDriver1 #Attribute ::= {
    ENCODE STRUCTURE {
        STRUCTURED WITH eAttributePrepaddingAdder1 }
    WITH FastInfoSetEncodingSet }

-- Encodes an item of the components attribute-values,
-- content-character-chunks, and other-strings of the Document
-- type (see C.2.5.4)
eEncodedCharacterStringDriver1 #EncodedCharacterString ::= {
    ENCODE STRUCTURE {
        STRUCTURED WITH eEncodedCharacterStringPrepaddingAdder1 }
    WITH FastInfoSetEncodingSet }

-- Encodes an item of the components element-name-surrogates and
-- attribute-name-surrogates of the Document type (see C.2.5.5)

```

```

eNameSurrogateDriver1 #NameSurrogate ::= {
    ENCODE STRUCTURE {
        STRUCTURED WITH eNameSurrogatePrepaddingAdder1 }
    WITH FastInfoSetEncodingSet }

-- Encodes the initial-vocabulary component of the Document
-- type (see C.2.5)

eInitialVocabularyPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eInitialVocabularyWithPrepadding1 }

-- Inserts pre-padding before the encoding of an item of the
-- notations component of the Document type (see C.2.6.1)

eNotationPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eNotationWithPrepadding1 }

-- Inserts pre-padding before the encoding of an item of the
-- unparsed-entities component of the Document type (see C.2.7.1)

eUnparsedEntityPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eUnparsedEntityWithPrepadding1 }

-- Inserts pre-padding before the encoding of the standalone
-- component of the Document type (see C.2.9)

eStandalonePrepaddingAdder1 #BOOL ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eStandaloneWithPrepadding1 }

-- Inserts pre-padding before the encoding of an item of the
-- children component of the DocumentTypeDeclaration type (see C.9.6)

eDocTypeDeclChildPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eDocTypeDeclChildWithPrepadding1 }

-- Inserts pre-padding before the encoding of an item of the
-- namespace-attributes component of the Element type (see C.3.4.2)

eNamespaceAttributePrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eNamespaceAttributeWithPrepadding1 }

-- Inserts pre-padding before the encoding of an item of the
-- attributes component of the Element type (see C.3.6.1)

eAttributePrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eAttributeWithPrepadding1 }

-- Inserts pre-padding before the encoding of the literal-qualified-name
-- component of the QualifiedNameOrIndex type. Used when the encoding
-- starts on the second bit of an octet (see C.17.3)

eLiteralQualifiedNamePrepaddingAdder2 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eLiteralQualifiedNameWithPrepadding2 }

-- Inserts pre-padding before the encoding of the literal-qualified-name
-- component of the QualifiedNameOrIndex type. Used when the encoding
-- starts on the third bit of an octet (see C.18.3)

eLiteralQualifiedNamePrepaddingAdder3 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eLiteralQualifiedNameWithPrepadding3 }

-- Inserts pre-padding before the encoding of an item of the components
-- restricted-alphabets, encoding-algorithms, prefixes, namespace-names,
-- local-names, other-ncnames, and other-uris of the Document
-- type (see C.2.5.3)

```

```

eNonEmptyOctetStringPrepaddingAdder1 #OCTET-STRING ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByPrepadding
        ENCODED BY eNonEmptyOctetStringWithPrepadding1 }}

-- Inserts pre-padding before the encoding of an item of the components
-- attribute-values, content-character-chunks, and other-strings of the
-- Document type (see C.2.5.4)

eEncodedCharacterStringPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eEncodedCharacterStringWithPrepadding1 }

-- Inserts pre-padding before the encoding of an item of the components
-- element-name-surrogates and attribute-name-surrogates of the Document
-- type (see C.2.5.5)

eNameSurrogatePrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eNameSurrogateWithPrepadding1 }

-- Inserts a discriminant before the encoding of an item of the
-- children component of the Document type (see C.2.11.2 to C.2.11.5)

eDocumentChildDiscriminantAdder1or5 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY eDocumentChildWithDiscriminant1or5 }

-- Inserts a discriminant before the encoding of an item of the
-- children component of the Element type (see C.3.7.2 to C.3.7.6)

eElementChildDiscriminantAdder1or5 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFiveAlternativeDiscriminant
    ENCODED BY eElementChildWithDiscriminant1or5 }

-- Inserts a discriminant before the encoding of the length of a SEQUENCE OF,
-- identifying one of the two ways of encoding the length (see C.21)

eNonEmptySequenceOfLengthDiscriminantAdder1 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByTwoAlternativeDiscriminant
    ENCODED BY eNonEmptySequenceOfLengthWithDiscriminant1 }

-- Inserts a discriminant before the encoding of the length of a
-- NonEmptyOctetString, identifying one of the three ways of encoding the
-- length. Used when the encoding starts on the second bit of an octet (see C.22)

eNonEmptyOctetStringLengthDiscriminantAdder2 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant2 }

-- Inserts a discriminant before the encoding of the length of a
-- NonEmptyOctetString, identifying one of the three ways of encoding the
-- length. Used when the encoding starts on the fifth bit of an octet (see C.23)

eNonEmptyOctetStringLengthDiscriminantAdder5 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant5 }

-- Inserts a discriminant before the encoding of the length of a
-- NonEmptyOctetString, identifying one of the three ways of encoding the
-- length. Used when the encoding starts on the seventh bit of an octet
-- (see C.24)

eNonEmptyOctetStringLengthDiscriminantAdder7 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant7 }

-- Inserts a discriminant before the encoding of a positive integer,
-- identifying one of the three ways of encoding it. Used when the encoding
-- starts on the second bit of an octet (see C.25)

ePositiveIntegerDiscriminantAdder2 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY ePositiveIntegerWithDiscriminant2 }

```

```

-- Inserts a discriminant before the encoding of a positive integer,
-- identifying one of the four ways of encoding it. Used when the encoding
-- starts on the third bit of an octet (see C.27)

ePositiveIntegerDiscriminantAdder3 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY ePositiveIntegerWithDiscriminant3 }

-- Inserts a discriminant before the encoding of a positive integer,
-- identifying one of the four ways of encoding it. Used when the encoding
-- starts on the fourth bit of an octet (see C.28)

ePositiveIntegerDiscriminantAdder4 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY ePositiveIntegerWithDiscriminant4 }

-- Inserts a discriminant before the encoding of a non-negative integer,
-- identifying one of the three ways of encoding it (see C.26)

eNonNegativeIntegerDiscriminantAdder2 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY eNonNegativeIntegerWithDiscriminant2 }

-- Sets the prepadding that has been added before the initial-vocabulary
-- component of the Document type and encodes the component (see C.2.5)

eInitialVocabularyWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 3
            PAD-PATTERN bits:'000'B },
        original {
            ENCODE STRUCTURE {
                restricted-alphabets {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }

                    WITH FastInfoSetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
                encoding-algorithms {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }

                    WITH FastInfoSetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
                prefixes {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }

                    WITH FastInfoSetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
                namespace-names {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }

                    WITH FastInfoSetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
                local-names {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }

                    WITH FastInfoSetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
                other-ncnames {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }

                    WITH FastInfoSetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
                other-uris {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eRepetitionWithLengthNonEmptyOctetString1
                    }
                }
            }
        }
    }
}

```

```

        WITH FastInfosetErrorEncodingSet }
        OPTIONAL-ENCODING USE-SET,
    attribute-values {
        ENCODE STRUCTURE {
            STRUCTURED WITH
                eRepetitionWithLengthEncodedCharacterString1 }
        WITH FastInfosetErrorEncodingSet }
        OPTIONAL-ENCODING USE-SET,
    content-character-chunks {
        ENCODE STRUCTURE {
            STRUCTURED WITH
                eRepetitionWithLengthEncodedCharacterString1 }
        WITH FastInfosetErrorEncodingSet }
        OPTIONAL-ENCODING USE-SET,
    other-strings {
        ENCODE STRUCTURE {
            STRUCTURED WITH
                eRepetitionWithLengthEncodedCharacterString1 }
        WITH FastInfosetErrorEncodingSet }
        OPTIONAL-ENCODING USE-SET,
    element-name-surrogates {
        ENCODE STRUCTURE {
            STRUCTURED WITH eRepetitionWithLengthNameSurrogate1 }
        WITH FastInfosetErrorEncodingSet }
        OPTIONAL-ENCODING USE-SET,
    attribute-name-surrogates {
        ENCODE STRUCTURE {
            STRUCTURED WITH eRepetitionWithLengthNameSurrogate1 }
        WITH FastInfosetErrorEncodingSet }
        OPTIONAL-ENCODING USE-SET }
    WITH FastInfosetErrorEncodingSet }}
    WITH FastInfosetErrorEncodingSet }

-- Sets the prepadding that has been added before each item of the notations
-- component of the Document type and encodes the item (see C.2.6.1)
eNotationWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 6
            PAD-PATTERN bits:'110000'B },
        original eNotation7 }
    WITH FastInfosetErrorEncodingSet }

-- Sets the prepadding that has been added before each item of the
-- unparsed-entities component of the Document type and encodes the item
-- (see C.2.7.1)
eUnparsedEntityWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 7
            PAD-PATTERN bits:'1101000'B },
        original eUnparsedEntity8 }
    WITH FastInfosetErrorEncodingSet }

-- Sets the prepadding that has been added before the standalone component of
-- the Document type and encodes the component (see C.2.9)
eStandaloneWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 7
            PAD-PATTERN bits:'0000000'B },
        original USE-SET }
    WITH FastInfosetErrorEncodingSet }

-- Sets the prepadding that has been added before each item of the
-- namespace-attributes component of the Element type and encodes the item
-- (see C.3.4.2)
eNamespaceAttributeWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {

```

```

        ENCODING-SPACE SIZE 6
        PAD-PATTERN bits:'110011'B
        EXHIBITS HANDLE "nsa" AT { 0 | 1 | 2 | 3 | 4 | 5}
        AS bits:'110011'B },
    original eNamespaceAttribute7
    STRUCTURED WITH {
        ENCODING-SPACE
        EXHIBITS HANDLE "nsa" AT { 0 | 1 | 2 | 3 | 4 | 5} AS bits:'110011'B
    }
    WITH FastInfoSetEncodingSet }

-- Sets the prepadding that has been added before each item of the attributes
-- component of the Element type and encodes the item (see C.3.6.1)
eAttributeWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 1
            PAD-PATTERN bits:'0'B },
        original eAttribute2 }
    WITH FastInfoSetEncodingSet }

-- Sets the prepadding that has been added before each item of the
-- children component of the DocumentTypeDeclaration type and encodes the
-- item (see C.9.6)
eDocTypeDeclChildWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 8
            PAD-PATTERN bits:'11100001'B },
        original eProcessingInstruction1 }
    WITH FastInfoSetEncodingSet }

-- Sets the prepadding that has been added before each item of the components
-- restricted-alphabets, encoding-algorithms, prefixes, namespace-names,
-- local-names, other-ncnames, and other-uris of the Document type and encodes
-- the item (see C.2.5.3)
eNonEmptyOctetStringWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 1
            PAD-PATTERN bits:'0'B },
        original USE-SET }
    WITH FastInfoSetEncodingSet }

-- Sets the prepadding that has been added before each item of the components
-- attribute-values, content-character-chunks, and other-strings of the
-- Document type and encodes the item (see C.2.5.4)
eEncodedCharacterStringWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 2
            PAD-PATTERN bits:'00'B },
        original USE-SET }
    WITH FastInfoSetEncodingSet }

eNameSurrogateWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 6
            PAD-PATTERN bits:'000000'B },
        original eNameSurrogate7 }
    WITH FastInfoSetEncodingSet }

-- Sets the prepadding that has been added before the literal-qualified-name
-- component of the QualifiedNameOrIndex type and encodes the component.
-- Used when the encoding starts on the second bit of an octet (see C.17.3)
eLiteralQualifiedNameWithPrepadding2{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {

```

```

        ENCODING-SPACE SIZE 5
        PAD-PATTERN bits:'11110'B },
    original {
        ENCODE STRUCTURE {
            prefix eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET,
            namespace-name eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET,
            local-name eIdentifyingStringOrIndex1 }
        WITH FastInfoSetEncodingSet }
    STRUCTURED WITH {
        ENCODING-SPACE
        EXHIBITS HANDLE "qn" AT { 0 | 1 | 2 | 3 } AS bits:'1111'B }}
    WITH FastInfoSetEncodingSet }

-- Sets the prepadding that has been added before the literal-qualified-name
-- component of the QualifiedNameOrIndex type and encodes the component. Used
-- when the encoding starts on the third bit of an octet (see C.18.3)

eLiteralQualifiedNameWithPrepadding3{<#C>}
#PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 4
            PAD-PATTERN bits:'1111'B
            EXHIBITS HANDLE "qn" AT { 0 | 1 | 2 | 3 } AS bits:'1111'B },
        original {
            ENCODE STRUCTURE {
                prefix eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
                namespace-name eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
                local-name eIdentifyingStringOrIndex1 }
            WITH FastInfoSetEncodingSet }
        STRUCTURED WITH {
            ENCODING-SPACE
            EXHIBITS HANDLE "qn" AT { 0 | 1 | 2 | 3 } AS bits:'1111'B }}
        WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a SEQUENCE OF and encodes
-- the SEQUENCE OF NonEmptyOctetString (see C.21)

eNonEmptySequenceOfWithLengthNonEmptyOctetString1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
            ENCODE STRUCTURE {
                eNonEmptyOctetStringPrepaddingAdder1
                STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
        WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a SEQUENCE OF and encodes
-- the SEQUENCE OF EncodedCharacterString (see C.21)

eNonEmptySequenceOfWithLengthEncodedCharacterString1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
            ENCODE STRUCTURE {
                eEncodedCharacterStringDriver1
                STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
        WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a SEQUENCE OF and encodes
-- the SEQUENCE OF NameSurrogate (see C.21)

eNonEmptySequenceOfWithLengthNameSurrogate1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,

```

```

original {
    ENCODE STRUCTURE {
        eNameSurrogateDriver1
        STRUCTURED WITH eRepetitionItems1{<length>}
    } WITH PER-BASIC-UNALIGNED }}
WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a SEQUENCE OF and encodes
-- the SEQUENCE OF additional-datum (see C.21)

eNonEmptySequenceOfWithLengthAdditionalDatum1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
            ENCODE STRUCTURE {
                additional-datum {
                    ENCODE STRUCTURE {
                        id eNonEmptyOctetStringPrepaddingAdder1,
                        data eNonEmptyOctetStringPrepaddingAdder1 }
                    WITH FastInfoSetEncodingSet}
                STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
        WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a NonEmptyOctetString
-- and encodes the NonEmptyOctetString. Used when the encoding starts on the
-- second bit of an octet (see C.22)

eNonEmptyOctetStringWithLength2{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptyOctetStringLength2,
        original eOctetStringOctets1{<length>} }
    WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a NonEmptyOctetString
-- and encodes the NonEmptyOctetString. Used when the encoding starts on the
-- fifth bit of an octet (see C.23)

eNonEmptyOctetStringWithLength5{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptyOctetStringLength5,
        original eOctetStringOctets1{<length>} }
    WITH FastInfoSetEncodingSet }

-- Encodes the length field that has been added before a NonEmptyOctetString
-- and encodes the NonEmptyOctetString. Used when the encoding starts on the
-- seventh bit of an octet (see C.24)

eNonEmptyOctetStringWithLength7{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptyOctetStringLength7,
        original eOctetStringOctets1{<length>} }
    WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before an item of the children
-- component of the Document type and encodes the item (see C.2.11.2 to C.2.11.5)

eDocumentChildWithDiscriminant1or5{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ALIGNED TO NEXT octet
            ENCODING-SPACE SIZE 0 },
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '11100001'B,
                2 TO '11100010'B,
                3 TO '110001'B }
        }
    }
}

```

```

        WITH FastInfoSetEncodingSet },
    original {
        ENCODE STRUCTURE {
            element eElement2,
            processing-instruction eProcessingInstruction1,
            comment eComment1,
            document-type-declaration eDocumentTypeDeclaration7
        STRUCTURED WITH {
            ALTERNATIVE DETERMINED BY field-to-be-set
            USING discriminant }}
        WITH FastInfoSetEncodingSet }}
    WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before an item of the children
-- component of the Element type and encodes the item (see C.3.7.2 to C.3.7.6)

eElementChildWithDiscriminant1or5{<#C>}
#PrecededByFiveAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ALIGNED TO NEXT octet
            ENCODING-SPACE SIZE 0 },
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '11100001'B,
                2 TO '110010'B,
                3 TO '10'B,
                4 TO '11100010'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                element eElement2,
                processing-instruction eProcessingInstruction1,
                unexpanded-entity-reference eUnexpandedEntityReference7,
                character-chunk eCharacterChunk3,
                comment eComment1
            STRUCTURED WITH {
                ALTERNATIVE DETERMINED BY field-to-be-set
                USING discriminant }}
            WITH FastInfoSetEncodingSet }}
        WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before the length of a
-- SEQUENCE OF (identifying one of the two ways of encoding the length)
-- and encodes the length (see C.21)

eNonEmptySequenceOfLengthWithDiscriminant1{<#C>}
#PrecededByTwoAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '1000'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
                WITH FastInfoSetEncodingSet }}
        WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before the length of a
-- NonEmptyOctetString (identifying one of the three ways of encoding the
-- length) and encodes the length. Used when the encoding starts on the
-- second bit of an octet (see C.22)

eNonEmptyOctetStringLengthWithDiscriminant2{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {

```

```

discriminant {
    USE #BIT-STRING
    MAPPING TO BITS {
        0 TO '0'B,
        1 TO '1000000'B,
        2 TO '1100000'B }
    WITH FastInfoSetEncodingSet },
original {
    ENCODE STRUCTURE {
        STRUCTURED WITH {
            ALTERNATIVE DETERMINED BY field-to-be-set
            USING discriminant }}
    WITH FastInfoSetEncodingSet }}
WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before the length of a
-- NonEmptyOctetString (identifying one of the three ways of encoding the
-- length) and encodes the length. Used when the encoding starts on the fifth
-- bit of an octet (see C.23)

eNonEmptyOctetStringLengthWithDiscriminant5{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '1000'B,
                2 TO '1100'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfoSetEncodingSet }}
    WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before the length of
-- a NonEmptyOctetString (identifying one of the three ways of encoding
-- the length) and encodes the length. Used when the encoding starts on
-- the seventh bit of an octet (see C.24)

eNonEmptyOctetStringLengthWithDiscriminant7{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '10'B,
                2 TO '11'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfoSetEncodingSet }}
    WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before a positive integer
-- (identifying one of the three ways of encoding the integer) and encodes
-- the integer. Used when the encoding starts on the second bit of an octet
-- (see C.25)

ePositiveIntegerWithDiscriminant2{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,

```

```

        1 TO '10'B,
        2 TO '110'B }
    WITH FastInfoSetEncodingSet },
original {
    ENCODE STRUCTURE {
        STRUCTURED WITH {
            ALTERNATIVE DETERMINED BY field-to-be-set
            USING discriminant }}
    WITH FastInfoSetEncodingSet }
STRUCTURED WITH {
    ENCODING-SPACE SIZE self-delimiting-values
    EXHIBITS HANDLE "qn" AT { 0 | 1 | 2 | 3 }
    AS range:{low 0, high 12}} -- Less than '1110'B
WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before a positive integer
-- (identifying one of the four ways of encoding the integer) and encodes
-- the integer. Used when the encoding starts on the third bit of an octet
-- (see C.27)

ePositiveIntegerWithDiscriminant3{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '100'B,
                2 TO '101'B,
                3 TO '110000000'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfoSetEncodingSet }
        STRUCTURED WITH {
            ENCODING-SPACE SIZE self-delimiting-values
            EXHIBITS HANDLE "qn" AT { 0 | 1 | 2 | 3 }
            AS range:{low 0, high 14}} -- Less than '1111'B
        WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before a positive integer
-- (identifying one of the four ways of encoding the integer) and encodes
-- the integer. Used when the encoding starts on the fourth bit of an octet
-- (see C.28)

ePositiveIntegerWithDiscriminant4{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '0'B,
                1 TO '100'B,
                2 TO '101'B,
                3 TO '110000000'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfoSetEncodingSet }
        WITH FastInfoSetEncodingSet }

-- Encodes the discriminant that has been added before a non-negative integer
-- (identifying one of the three ways of encoding the integer) and encodes
-- the integer (see C.26)

```

```

eNonNegativeIntegerWithDiscriminant2{<#C>
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
            MAPPING TO BITS {
                0 TO '1111111'B,
                1 TO '0'B,
                2 TO '10'B,
                3 TO '110'B }
            WITH FastInfoSetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
                WITH FastInfoSetEncodingSet }}}
        WITH FastInfoSetEncodingSet }

-- Encodes the Document type (see C.2)
eDocument2 #Document ::= {
    ENCODE STRUCTURE {
        additional-data {
            ENCODE STRUCTURE {
                STRUCTURED WITH eRepetitionWithLengthAdditionalDatum1 }
            WITH FastInfoSetEncodingSet }
            OPTIONAL-ENCODING USE-SET,
        initial-vocabulary {
            ENCODE STRUCTURE {
                STRUCTURED WITH eInitialVocabularyPrepaddingAdder1 }
            WITH FastInfoSetEncodingSet }
            OPTIONAL-ENCODING USE-SET,
        notations {
            ENCODE STRUCTURE {
                eNotationDriver1
                STRUCTURED WITH eRepetitionWithTerminator8bit1 }
            WITH FastInfoSetEncodingSet }
            OPTIONAL-ENCODING USE-SET,
        unparsed-entities {
            ENCODE STRUCTURE {
                eUnparsedEntityDriver1
                STRUCTURED WITH eRepetitionWithTerminator8bit1 }
            WITH FastInfoSetEncodingSet }
            OPTIONAL-ENCODING USE-SET,
        character-encoding-scheme eNonEmptyOctetStringPrepaddingAdder1
            OPTIONAL-ENCODING USE-SET,
        standalone eStandalonePrepaddingAdder1
            OPTIONAL-ENCODING USE-SET,
        version eNonIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET,
        children {
            ENCODE STRUCTURE {
                {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH eDocumentChildDiscriminantAdder1or5 }
                        WITH FastInfoSetEncodingSet }
                    STRUCTURED WITH eRepetitionWithTerminator4bit1 }
                WITH FastInfoSetEncodingSet }}}
            WITH FastInfoSetEncodingSet }

-- Encodes the Element type (see C.3)
eElement2 #Element ::= {
    ENCODE STRUCTURE {
        namespace-attributes {
            ENCODE STRUCTURE {
                eNamespaceAttributeDriver1
                STRUCTURED WITH eRepetitionWithTerminator10bit1 }
            WITH FastInfoSetEncodingSet }
            OPTIONAL-ENCODING eNamespaceAttributesOptionality3,
        qualified-name eQualifiedNameOrIndex3,

```

```

attributes {
  ENCODE STRUCTURE {
    eAttributeDriver1
    STRUCTURED WITH eRepetitionWithTerminator4bit1 }
  WITH FastInfoSetEncodingSet }
  OPTIONAL-ENCODING USE-SET,
children {
  ENCODE STRUCTURE {
    {
      ENCODE STRUCTURE {
        STRUCTURED WITH eElementChildDiscriminantAdder1or5 }
        WITH FastInfoSetEncodingSet }
      STRUCTURED WITH eRepetitionWithTerminator4bit1 }
      WITH FastInfoSetEncodingSet }}}
  WITH FastInfoSetEncodingSet }
-- Encodes the Attribute type (see C.4)
eAttribute2 #Attribute ::= {
  ENCODE STRUCTURE {
    qualified-name eQualifiedNameOrIndex2,
    normalized-value eNonIdentifyingStringOrIndex1 }
  WITH FastInfoSetEncodingSet }
-- Encodes the ProcessingInstruction type (see C.5)
eProcessingInstruction1 #ProcessingInstruction ::= {
  ENCODE STRUCTURE {
    target eIdentifyingStringOrIndex1,
    content eNonIdentifyingStringOrIndex1 }
  WITH FastInfoSetEncodingSet }
-- Encodes the UnexpandedEntityReference type (see C.6)
eUnexpandedEntityReference7 #UnexpandedEntityReference ::= {
  ENCODE STRUCTURE {
    name eIdentifyingStringOrIndex1,
    system-identifier eIdentifyingStringOrIndex1
    OPTIONAL-ENCODING USE-SET,
    public-identifier eIdentifyingStringOrIndex1
    OPTIONAL-ENCODING USE-SET }
  WITH FastInfoSetEncodingSet }
-- Encodes the CharacterChunk type (see C.7)
eCharacterChunk3 #CharacterChunk ::= {
  ENCODE STRUCTURE {
    character-codes eNonIdentifyingStringOrIndex3 }
  WITH FastInfoSetEncodingSet }
-- Encodes the Comment type (see C.8)
eComment1 #Comment ::= {
  ENCODE STRUCTURE {
    content eNonIdentifyingStringOrIndex1 }
  WITH FastInfoSetEncodingSet }
-- Encodes the DocumentTypeDeclaration type (see C.9)
eDocumentTypeDeclaration7 #DocumentTypeDeclaration ::= {
  ENCODE STRUCTURE {
    system-identifier eIdentifyingStringOrIndex1
    OPTIONAL-ENCODING USE-SET,
    public-identifier eIdentifyingStringOrIndex1
    OPTIONAL-ENCODING USE-SET,
    children {
      ENCODE STRUCTURE {
        {
          ENCODE STRUCTURE {
            STRUCTURED WITH eDocTypeDeclChildPrepaddingAdder1 }
            WITH FastInfoSetEncodingSet }
          STRUCTURED WITH eRepetitionWithTerminator4bit1 }
          WITH FastInfoSetEncodingSet }}}
    WITH FastInfoSetEncodingSet }

```

```

-- Encodes the UnparsedEntity type (see C.10)
eUnparsedEntity8 #UnparsedEntity ::= {
    ENCODE STRUCTURE {
        name eIdentifyingStringOrIndex1,
        system-identifier eIdentifyingStringOrIndex1,
        public-identifier eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET,
        notation-name eIdentifyingStringOrIndex1 }
    WITH FastInfoSetEncodingSet }

-- Encodes the Notation type (see C.11)
eNotation7 #Notation ::= {
    ENCODE STRUCTURE {
        name eIdentifyingStringOrIndex1,
        system-identifier eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET,
        public-identifier eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET }
    WITH FastInfoSetEncodingSet }

-- Encodes the NamespaceAttribute type (see C.12)
eNamespaceAttribute7 #NamespaceAttribute ::= {
    ENCODE STRUCTURE {
        prefix eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET,
        namespace-name eIdentifyingStringOrIndex1
            OPTIONAL-ENCODING USE-SET }
    WITH FastInfoSetEncodingSet }

-- Encodes the IdentifyingStringOrIndex type (see C.13)
eIdentifyingStringOrIndex1 #IdentifyingStringOrIndex ::= {
    ENCODE STRUCTURE {
        literal-character-string eNonEmptyOctetString2,
        string-index ePositiveInteger2 }
    WITH FastInfoSetEncodingSet }

-- Encodes the NonIdentifyingStringOrIndex type. Used when the encoding starts
-- on the first bit of an octet (see C.14)
eNonIdentifyingStringOrIndex1 #NonIdentifyingStringOrIndex ::= {
    ENCODE STRUCTURE {
        literal-character-string {
            ENCODE STRUCTURE {
                add-to-table USE-SET,
                character-string eEncodedCharacterString3 }
            WITH FastInfoSetEncodingSet },
        string-index eNonNegativeInteger2 }
    WITH FastInfoSetEncodingSet }

-- Encodes the NonIdentifyingStringOrIndex type. Used when the encoding starts
-- on the third bit of an octet (see C.15)
eNonIdentifyingStringOrIndex3 #NonIdentifyingStringOrIndex ::= {
    ENCODE STRUCTURE {
        literal-character-string {
            ENCODE STRUCTURE {
                add-to-table USE-SET,
                character-string eEncodedCharacterString5 }
            WITH FastInfoSetEncodingSet },
        string-index ePositiveInteger4 }
    WITH FastInfoSetEncodingSet }

-- Encodes the NameSurrogate type (see C.16)
eNameSurrogate7 #NameSurrogate ::= {
    ENCODE STRUCTURE {
        prefix-string-index ePositiveInteger2
            OPTIONAL-ENCODING USE-SET,
        namespace-name-string-index ePositiveInteger2

```

```

        OPTIONAL-ENCODING USE-SET,
        local-name-string-index ePositiveInteger2 }
    WITH FastInfoSetEncodingSet }

-- Encodes the QualifiedNameOrIndex type. Used when the encoding starts
-- on the second bit of an octet (see C.17)

eQualifiedNameOrIndex2 #QualifiedNameOrIndex ::= {
    ENCODE STRUCTURE {
        literal-qualified-name {
            ENCODE STRUCTURE {
                STRUCTURED WITH eLiteralQualifiedNamePrepaddingAdder2 }
            WITH FastInfoSetEncodingSet },
        name-surrogate-index ePositiveInteger2
        STRUCTURED WITH eQualifiedNameAlternatives3 }
    WITH FastInfoSetEncodingSet }

-- Encodes the QualifiedNameOrIndex type. Used when the encoding starts
-- on the third bit of an octet (see C.18)

eQualifiedNameOrIndex3 #QualifiedNameOrIndex ::= {
    ENCODE STRUCTURE {
        literal-qualified-name {
            ENCODE STRUCTURE {
                STRUCTURED WITH eLiteralQualifiedNamePrepaddingAdder3 }
            WITH FastInfoSetEncodingSet },
        name-surrogate-index ePositiveInteger3
        STRUCTURED WITH eQualifiedNameAlternatives3 }
    WITH FastInfoSetEncodingSet }

-- Encodes the EncodedCharacterString type. Used when the encoding starts
-- on the third bit of an octet (see C.19)

eEncodedCharacterString3 #EncodedCharacterString ::= {
    ENCODE STRUCTURE {
        encoding-format USE-SET,
        octets eNonEmptyOctetString5 }
    WITH FastInfoSetEncodingSet }

-- Encodes the EncodedCharacterString type. Used when the encoding starts
-- on the fifth bit of an octet (see C.20)

eEncodedCharacterString5 #EncodedCharacterString ::= {
    ENCODE STRUCTURE {
        encoding-format USE-SET,
        octets eNonEmptyOctetString7 }
    WITH FastInfoSetEncodingSet }

-- Encodes a repetition (SEQUENCE OF NonEmptyOctetString) by inserting a length
-- field before it (see C.2.5.3 to C.2.5.5)

eRepetitionWithLengthNonEmptyOctetString1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthNonEmptyOctetString1 }}

-- Encodes a repetition (SEQUENCE OF EncodedCharacterString) by inserting a length
-- field before it (see C.2.5.3 to C.2.5.5)

eRepetitionWithLengthEncodedCharacterString1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthEncodedCharacterString1 }}

-- Encodes a repetition (SEQUENCE OF NameSurrogate) by inserting a length
-- field before it (see C.2.5.3 to C.2.5.5)

eRepetitionWithLengthNameSurrogate1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthNameSurrogate1 }}

-- Encodes a repetition (SEQUENCE OF additional-datum) by inserting a length
-- field before it (see C.2.5.3 to C.2.5.5)

```

```

eRepetitionWithLengthAdditionalDatum1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthAdditionalDatum1 }}

-- Encodes the NonEmptyOctetString type. Used when the encoding starts
-- on the second bit of an octet (see C.22)

eNonEmptyOctetString2 #NonEmptyOctetString ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
        ENCODED BY eNonEmptyOctetStringWithLength2 }}

-- Encodes the NonEmptyOctetString type. Used when the encoding starts
-- on the fifth bit of an octet (see C.23)

eNonEmptyOctetString5 #NonEmptyOctetString ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
        ENCODED BY eNonEmptyOctetStringWithLength5 }}

-- Encodes the NonEmptyOctetString type. Used when the encoding starts
-- on the seventh bit of an octet (see C.24)

eNonEmptyOctetString7 #NonEmptyOctetString ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
        ENCODED BY eNonEmptyOctetStringWithLength7 }}

-- Encodes the length field that has been inserted before the encoding of
-- a SEQUENCE OF (see C.21)

eNonEmptySequenceOfLength1 #NonEmptySequenceOfLength ::= {
    USE #NonEmptySequenceOfLengthAlternatives1
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNonEmptySequenceOfLengthDiscriminantAdder1 }
        WITH FastInfoSetEncodingSet }}

-- Encodes the length field that has been inserted before the encoding of
-- a NonEmptyOctetString. Used when the encoding starts on the second bit
-- of an octet (see C.22)

eNonEmptyOctetStringLength2 #NonEmptyOctetStringLength ::= {
    USE #NonEmptyOctetStringLengthAlternatives2
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNonEmptyOctetStringLengthDiscriminantAdder2 }
        WITH FastInfoSetEncodingSet }}

-- Encodes the length field that has been inserted before the encoding of a
-- NonEmptyOctetString. Used when the encoding starts on the fifth bit of
-- an octet (see C.23)

eNonEmptyOctetStringLength5 #NonEmptyOctetStringLength ::= {
    USE #NonEmptyOctetStringLengthAlternatives5
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNonEmptyOctetStringLengthDiscriminantAdder5 }
        WITH FastInfoSetEncodingSet }}

-- Encodes the length field that has been inserted before the encoding of
-- a NonEmptyOctetString. Used when the encoding starts on the seventh bit
-- of an octet (see C.24)

eNonEmptyOctetStringLength7 #NonEmptyOctetStringLength ::= {
    USE #NonEmptyOctetStringLengthAlternatives7
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNonEmptyOctetStringLengthDiscriminantAdder7 }
        WITH FastInfoSetEncodingSet }}

```

```

-- Encodes a positive integer. Used when the encoding starts on the second bit
-- of an octet (see C.25)

ePositiveInteger2 #PositiveOrNonNegativeInteger ::= {
    USE #PositiveIntegerAlternatives2
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH ePositiveIntegerDiscriminantAdder2 }
        WITH FastInfoSetEncodingSet }}

-- Encodes a positive integer. Used when the encoding starts on the third bit
-- of an octet (see C.27)

ePositiveInteger3 #PositiveOrNonNegativeInteger ::= {
    USE #PositiveIntegerAlternatives3
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH ePositiveIntegerDiscriminantAdder3 }
        WITH FastInfoSetEncodingSet }}

-- Encodes a positive integer. Used when the encoding starts on the fourth bit
-- of an octet (see C.28)

ePositiveInteger4 #PositiveOrNonNegativeInteger ::= {
    USE #PositiveIntegerAlternatives4
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH ePositiveIntegerDiscriminantAdder4 }
        WITH FastInfoSetEncodingSet }}

-- Encodes a non-negative integer (see C.26)

eNonNegativeInteger2 #PositiveOrNonNegativeInteger ::= {
    USE #NonNegativeIntegerAlternatives2
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNonNegativeIntegerDiscriminantAdder2 }
        WITH FastInfoSetEncodingSet }}

-- Specifies how to determine the presence of the namespace-attributes component
-- of the Element type (see C.3.4.2)

eNamespaceAttributesOptionality3 #OPTIONAL ::= {
    PRESENCE DETERMINED BY handle
    HANDLE "nsa" }

-- Specifies how to determine the alternative of the QualifiedNameOrIndex type
-- (see C.17.3 and C.18.3)

eQualifiedNameAlternatives3 #ALTERNATIVES ::= {
    ALTERNATIVE DETERMINED BY handle
    HANDLE "qn"
    EXHIBITS HANDLE "nsa" AT { 0 | 1 | 2 | 3 | 4 | 5 }
    AS range:{low 0, high 50}} -- Less than '110011'B

-- Specifies how to determine the termination of a repetition using a 4-bit
-- terminator '1111'(see C.2.12, C.3.6.2, C.3.8, and C.9.7)

eRepetitionWithTerminator4bit1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        DETERMINED BY pattern PATTERN bits:'1111'B }}

-- Specifies how to determine the termination of a repetition using an 8-bit
-- terminator '11110000'(see C.2.6.2 and C.2.7.2)

eRepetitionWithTerminator8bit1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        DETERMINED BY pattern PATTERN bits:'11110000'B }}

```

```

-- Specifies how to determine the termination of a repetition using a 10-bit
-- terminator '1111000000'(see C.3.4.3)

eRepetitionWithTerminator10bit1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        DETERMINED BY pattern PATTERN bits:'1111000000'B
        EXHIBITS HANDLE "nsa" AT { 0 | 1 | 2 | 3 | 4 | 5 } AS bits:'110011'B }}

-- Encodes the items of a SEQUENCE OF, following the length field that has been
-- added (see C.2.5.3 to C.2.5.5)

eRepetitionItems1{<REFERENCE:len>} #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant MULTIPLE OF bit
        DETERMINED BY field-to-be-set USING len }}

-- Encodes the octets of a NonEmptyOctetString, following the length field that
-- has been added (see C.22, C.23, and C.24)

eOctetStringOctets1{<REFERENCE:len>} #OCTETS ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant MULTIPLE OF bit
        DETERMINED BY field-to-be-set USING len }}
empty-padding #PAD ::= {
    ENCODING-SPACE SIZE 0
}

FastInfoSetEncodingSet #ENCODINGS ::= { eDocument2 | empty-padding }
    COMPLETED BY PER-BASIC-UNALIGNED

END

FastInfoSetELM
    {joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoSet(0)
    modules(0) fast-infoSet-elm(2)}
    LINK-DEFINITIONS ::= BEGIN
    IMPORTS FastInfoSetEncodingSet, Document FROM FastInfoSetEDM;
    ENCODE #Document WITH FastInfoSetEncodingSet

END

```

Anexo B

Tipo de medios MIME para documentos infoset rápidos

(Este anexo es parte integrante de la presente Recomendación | Norma Internacional)

Este anexo define el tipo de medios "application/ fastinfoset" que describe documentos infoset rápidos.

El tipo de medios MIME se especifica a continuación utilizando la plantilla de registro IETF MIME y se ha registrado de conformidad con los procedimientos IETF.

Nombre del tipo de medios MIME:
Aplicación

Nombre del subtipo MIME:
fastinfoset

Parámetros requeridos:
Ninguno.

Parámetros opcionales:
Ninguno.

Consideraciones de codificación:

Los infosets XML codificados como documentos infoset rápidos darán como resultado la generación de datos binarios. Este tipo de medios MIME puede precisar codificación ulterior en medios de transporte que no sean capaces de manejar datos binarios.

Consideraciones de seguridad:

Dado que los infosets XML codificados como documentos infoset rápidos pueden transportar datos definidos por una aplicación cuya semántica sea independiente de cualquier entorno MIME (o del contexto en el que se utiliza el entorno MIME), se debería poder comprender la semántica del documento infoset rápido basándose únicamente en la semántica del entorno MIME. Por lo tanto, siempre que se utilice el tipo de medios "application/fastinfoset", se recomienda encarecidamente que las implicaciones de seguridad del contexto en el que se utiliza el documento infoset rápido sean totalmente comprensibles.

Consideraciones sobre interfuncionamiento:

No se conoce ningún asunto de interfuncionamiento.

Especificación publicada:

Rec. UIT-T X.891 | ISO/CEI 24824-1

Aplicaciones que utilizan este tipo de medios:

Ninguna aplicación actualmente conocida utiliza este tipo de medios.

Información adicional:

Números mágicos:

Un documento infoset rápido puede comenzar con una declaración XML opcional que debe ser una de las cadenas siguientes codificada en UTF-8:

```
<?xml encoding='finf'?>
<?xml encoding='finf' standalone='yes'?>
<?xml encoding='finf' standalone='no'?>
<?xml version='1.0' encoding='finf'?>
<?xml version='1.0' encoding='finf' standalone='yes'?>
<?xml version='1.0' encoding='finf' standalone='no'?>
<?xml version='1.1' encoding='finf'?>
<?xml version='1.1' encoding='finf' standalone='yes'?>
<?xml version='1.1' encoding='finf' standalone='no'?>
```

Los primeros cinco octetos de la declaración XML codificada en UTF-8 son hexadecimales 3C 3F 78 6D 6C. Los cuatro octetos que identifican un documento infoset rápido correspondiente a la subcadena "finf" codificada en UTF-8 son hexadecimales 66 69 6E 66.

Un documento infoset rápido comenzará con una secuencia de octetos E0 00 00 01 hexadecimal, si está ausente la declaración XML opcional.

Ampliaciones de fichero:
*.finf

Persona y dirección de correo electrónico para más información:

Relator ASN.1 del UIT-T (tsbmail@itu.int)

Relator ASN.1 de ISO/CEI JTC1/SC6 (ittf@iso.org)

Utilización que se pretende:

COMÚN

Autor/controlador de cambios:

Procedimientos conjuntos UIT-T | ISO/CEI de conformidad con la Rec. UIT-T A.23 *Colaboración con la Organización Internacional para la Normalización (ISO) y la Comisión Electrotécnica Internacional (CEI) sobre tecnologías de la información*, anexo A y directrices del JTC1 de ISO/CEI, anexo K.

Anexo C

Descripción de la codificación de un documento infoset rápido

(Este anexo no es parte integrante de la presente Recomendación | Norma Internacional)

C.1 Documento infoset rápido

C.1.1 Este anexo describe informalmente (pero de forma precisa y completa) las codificaciones que se especifican en la cláusula 12 y en el anexo A. Para conveniencia de los implementadores, todas las definiciones de tipo ASN.1 en el texto normativo se copian en este anexo en lugar de sencillamente referenciarlo.

C.1.2 Las codificaciones se describen en términos de acciones a realizar por el codificador, dando lugar a bits que se añaden al tren de bits. Las acciones a realizar por el decodificador no se describen explícitamente en el presente anexo, pero se pueden inferir de las acciones del codificador descritas en este anexo.

C.1.3 Un documento infoset rápido puede empezar con una declaración XML (véase 12.3) seguida de:

- a) los dieciséis bits '1110000000000000' (identificación); seguidos de
- b) los dieciséis bits '0000000000000001' (número de versión); seguidos de
- c) el bit '0' (relleno),

o con los mismos treinta y tres bits sin declaración XML. Los treinta y tres bits vienen seguidos inmediatamente por la codificación de un valor abstracto del tipo `Document` como se describe en C.2. Esta codificación finaliza en el bit octavo o cuarto de un octeto, dependiendo del contenido del documento infoset rápido. En el último caso, se añaden al tren de bits los cuatro bits '0000' (relleno).

C.2 Codificación del tipo Document

C.2.1 El tipo `Document` se define en 7.2 de la forma siguiente:

```
Document ::= SEQUENCE {
    additional-data          SEQUENCE (SIZE(1..one-meg)) OF
        additional-datum SEQUENCE {
            id                URI,
            data              NonEmptyOctetString } OPTIONAL,
    initial-vocabulary      SEQUENCE {
        external-vocabulary  URI OPTIONAL,
        restricted-alphabets SEQUENCE (SIZE(1..256)) OF
            NonEmptyOctetString OPTIONAL,
        encoding-algorithms  SEQUENCE (SIZE(1..256)) OF
            NonEmptyOctetString OPTIONAL,
        prefixes             SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        namespace-names     SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        local-names         SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        other-ncnames       SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        other-uris          SEQUENCE (SIZE(1..one-meg)) OF
            NonEmptyOctetString OPTIONAL,
        attribute-values    SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
        content-character-chunks SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
        other-strings       SEQUENCE (SIZE(1..one-meg)) OF
            EncodedCharacterString OPTIONAL,
        element-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
            NameSurrogate OPTIONAL,
        attribute-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
            NameSurrogate OPTIONAL }
        (CONSTRAINED BY {
            -- If the initial-vocabulary component is present, at least
            -- one of its components shall be present -- }) OPTIONAL,
    notations              SEQUENCE (SIZE(1..MAX)) OF
        Notation OPTIONAL,
    unparsed-entities     SEQUENCE (SIZE(1..MAX)) OF
        UnparsedEntity OPTIONAL,
```

```

character-encoding-scheme NonEmptyOctetString OPTIONAL,
standalone                BOOLEAN OPTIONAL,
version                   NonIdentifyingStringOrIndex OPTIONAL
                        -- OTHER STRING category --,
children                  SEQUENCE (SIZE(0..MAX)) OF
                        CHOICE {
                            element           Element,
                            processing-instruction ProcessingInstruction,
                            comment          Comment,
                            document-type-declaration DocumentTypeDeclaration }}

```

C.2.2 Un valor del tipo **Document** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre comienza en el segundo bit de un octeto y finaliza en el cuarto o en el octavo bit de otro octeto (que es el último bit del terminador '1111' descrito en C.2.12).

C.2.3 Para cada uno de los siete componentes opcionales **additional-data**, **initial-vocabulary**, **notations**, **unparsed-entities**, **character-encoding-scheme**, **standalone** y **version** (en este orden), si el componente está presente, se añade el bit '1' (presencia) al tren de bits, en otro caso, se añade el bit '0' (ausencia).

C.2.4 Si está presente el componente opcional **additional-data**, el número de componentes **additional-datum** se codifica como se describe en C.21 y cada uno de los componentes **additional-datum** se codifica como se describe en las dos subcláusulas siguientes.

C.2.4.1 El bit '0' (relleno) se añade al tren de bits y el componente **id** se codifica como se describe en C.22.

C.2.4.2 El bit '0' (relleno) se añade al tren de bits y el componente **data** se codifica como se describe en C.22.

C.2.5 Si está presente el componente opcional **initial-vocabulary**, se añaden los tres bits '000' (relleno) al tren de bits y se codifica el componente como se describe en las cinco subcláusulas siguientes.

C.2.5.1 Para cada uno de los trece componentes opcionales de **initial-vocabulary** (en orden textual), si está presente el componente, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.2.5.2 Si está presente el componente opcional **external-vocabulary** de **inicial-vocabulary**, se añade el bit '0' (relleno) al tren de bits y se codifica el componente como se describe en C.22.

C.2.5.3 Para cada uno de los componentes **restricted-alphabets**, **encoding-algorithms**, **prefixes**, **namespace-names**, **local-names**, **other-ncnames** y **other-uris** (en este orden) que esté presente, se codifica el número de elementos **NonEmptyOctetString** en el componente como se describe en C.21 y cada elemento se codifica (en orden) de la forma siguiente: el bit '0' (relleno) se añade al tren de bits y **NonEmptyOctetString** se codifica como se describe en C.22.

C.2.5.4 Para cada uno de los componentes **attribute-values**, **content-character-chunks** y **other-strings** (en este orden) que esté presente, se codifica el número de elementos **EncodedCharacterString** en el componente como se describe en C.21, y entonces se codifica cada elemento (en orden) como sigue: los dos bits '00' (relleno) se añaden al tren de bits y se codifica **EncodedCharacterString** como se describe en C.19.

C.2.5.5 Para cada uno de los componentes **element-name-surrogates** y **attribute-name-surrogates** (en este orden) que esté presente, se codifica el número de elementos **NameSurrogate** en el componente como se describe en C.21, y se codifica cada elemento (en orden) como sigue: los seis bits '000000' (relleno) se añaden al tren de bits y se codifica **NameSurrogate** como se describe en C.16.

C.2.6 Si está presente el componente opcional **notations**, se codifica el componente como se describe en las dos subcláusulas siguientes.

C.2.6.1 Cada elemento de **notations** (en orden) se codifica como sigue: se añaden los seis bits '110000' (identificación) al tren de bits, y se codifica **Notation** como se describe en C.11.

C.2.6.2 Se añaden los cuatro bits '1111' (terminación) y los cuatro bits '0000' (relleno) al tren de bits.

NOTA – Estos bits no se añaden si está ausente el componente **notations**.

C.2.7 Si está presente el componente opcional **unparsed-entities**, se codifica como se describe en las dos subcláusulas siguientes.

C.2.7.1 Cada elemento de **unparsed-entities** (en orden) se codifica como sigue: se añaden los siete bits '1101000' (identificación) al tren de bits y se codifica **UnparsedEntity** como se describe en C.10.

C.2.7.2 Se añaden los cuatro bits '1111' (terminación) y los cuatro bits '0000' (relleno) al tren de bits.

NOTA – Estos bits no se añaden si está ausente el componente **unparsed-entities**.

ISO/CEI 24824-1:2005 (S)

C.2.8 Si está presente el componente opcional **character-encoding-scheme**, se añade el bit "0" (relleno) al tren de bits y se codifica **NonEmptyOctetString** como se describe en C.22.

C.2.9 Si está presente el componente opcional **standalone**, se codifica como sigue: se añaden los siete bits '000000' (relleno) al tren de bits. Si el valor de **standalone** es **VERDADERO**, se añade el bit '1' al tren de bits, en otro caso, se añade el bit '0'.

C.2.10 Si está presente el componente opcional **version**, su valor se codifica como se describe en C.14.

C.2.11 Si el componente **children** tiene uno o más elementos, se codifica cada elemento (en orden) como se describe en las cinco subcláusulas siguientes.

C.2.11.1 Se requiere que la codificación de cada elemento se inicie en el primer bit de un octeto. Sin embargo, el último bit añadido puede haber sido el octavo o el cuarto bit de un octeto. Si ha sido el cuarto bit de un octeto, se añaden los bits '0000' (relleno) al tren de bits de forma que la codificación del elemento se inicie en el primer bit del octeto siguiente.

C.2.11.2 Si está presente la alternativa **element**, se añade el bit '0' (identificación) al tren de bits y se codifica **element** como se describe en C.3.

C.2.11.3 Si está presente la alternativa **processing-instruction**, se añaden los ocho bits '11110001' (identificación) al tren de bits y se codifica **processing-instruction** como se describe en C.5.

C.2.11.4 Si está presente la alternativa **comment**, se añaden los ocho bits '11100010' (identificación) al tren de bits y se codifica **comment** como se describe en C.8.

C.2.11.5 Si está presente la alternativa **document-type-declaration**, se añaden los seis bits '110001' (identificación) al tren de bits y se codifica **document-type-declaration** como se describe en C.9.

C.2.12 Se añaden los cuatro bits '1111' (terminación).

NOTA – Estos bits se añaden incluso cuando el componente **children** no tiene elementos.

C.3 Codificación del tipo Element

C.3.1 El tipo **Element** se define en 7.3 de la forma siguiente:

```
Element ::= SEQUENCE {
    namespace-attributes SEQUENCE (SIZE(1..MAX)) OF
        NamespaceAttribute OPTIONAL,
    qualified-name        QualifiedNameOrIndex
        -- categoría ELEMENT NAME --,
    attributes            SEQUENCE (SIZE(1..MAX)) OF
        Attribute OPTIONAL,
    children              SEQUENCE (SIZE(0..MAX)) OF
        CHOICE {
            element                Element,
            processing-instruction ProcessingInstruction,
            unexpanded-entity-reference UnexpandedEntityReference,
            character-chunk         CharacterChunk,
            comment                 Comment }}}
```

C.3.2 Se codifica un valor del tipo **Element** realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el segundo bit de un octeto y finaliza en el cuarto o en el octavo bit de un octeto (que es el último bit del terminador '1111' descrito en C.3.8)

C.3.3 Si está presente el componente opcional **attributes**, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.3.4 Si está presente el componente opcional **namespace-attributes**, se codifica como se describe en las tres subcláusulas siguientes.

C.3.4.1 Se añaden los cuatro bits '1110' (presencia) y los dos bits '00' (relleno) al tren de bits.

C.3.4.2 Cada elemento de **namespace-attributes** (en orden) se codifica como sigue: se añaden los seis bits '110011' (identificación) al tren de bits, y se codifica **NamespaceAttribute** como se describe en C.12.

C.3.4.3 Se añaden los cuatro bits '1111' (terminación) y los seis bits '000000' (relleno).

NOTA – Estos bits no se añaden si está ausente el componente **namespace-attributes**.

C.3.5 El valor del componente **qualified-name** se codifica como se describe en C.18.

C.3.6 Si está presente el componente opcional **attributes**, se codifica como se describe en las dos subcláusulas siguientes.

C.3.6.1 Cada elemento de **attributes** (en orden) se codifica como sigue: se añade el bit '0' (identificación) al tren de bits y se codifica **Attribute** como se describe en C.4.

C.3.6.2 Se añaden los cuatro bits '1111' (terminación).

NOTA – Estos bits no se añaden si está ausente el componente **attributes**.

C.3.7 Si el componente **children** tiene uno o mas elementos, se codifica cada elemento (en orden) como se describe en las seis subcláusulas siguientes.

C.3.7.1 Se requiere que la codificación de cada elemento se inicie en el primer bit de un octeto. Sin embargo, el último bit añadido puede haber sido el octavo o el cuarto bit de un octeto. Si ha sido el cuarto bit de un octeto, se añaden los bits '0000' (relleno) de forma que la codificación del elemento se inicie en el primer bit del octeto siguiente.

C.3.7.2 Si está presente la alternativa **element**, se añade el bit '0' (identificación) al tren de bits y se codifica **element** como se describe en esta subcláusula C.3.

C.3.7.3 Si está presente la alternativa **processing-instruction**, se añaden los ocho bits '11100001' (identificación) al tren de bits y se codifica **processing-instruction** como se describe en C.5.

C.3.7.4 Si está presente la alternativa **unexpanded-entity-reference**, se añaden los seis bits '110010' (identificación) al tren de bits y se codifica **unexpanded-entity-reference** como se describe en C.6.

C.3.7.5 Si está presente la alternativa **character-chunk**, se añaden los dos bits '10' (identificación) al tren de bits y se codifica **character-chunk** como se describe en C.7.

C.3.7.6 Si está presente la alternativa **comment**, se añaden los ocho bits '11100010' (identificación) al tren de bits y se codifica **comment** como se describe en C.8.

C.3.8 Se añaden los cuatro bits '1111' (terminación).

NOTA – Estos bits se añaden incluso cuando el componente **children** no tiene elementos.

C.4 Codificación del tipo **Attribute**

C.4.1 El tipo **attribute** se define en 7.4 como sigue:

```
Attribute ::= SEQUENCE {
    qualified-name      QualifiedNameOrIndex
                        -- categoría ATTRIBUTE NAME --,
    normalized-value   NonIdentifyingStringOrIndex
                        -- categoría ATTRIBUTE VALUE -- }
```

C.4.2 Se codifica un valor del tipo **Attribute** realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el segundo bit de un octeto y finaliza en el octavo bit de otro octeto.

C.4.3 El valor de **qualified-name** se codifica como se describe en C.17.

C.4.4 El valor de **normalized-value** se codifica como se describe en C.14.

C.5 Codificación del tipo **ProcessingInstruction**

C.5.1 El tipo **ProcessingInstruction** se define en 7.5 como sigue:

```
ProcessingInstruction ::= SEQUENCE {
    target      IdentifyingStringOrIndex
                -- categoría OTHER NCNAME --,
    content     NonIdentifyingStringOrIndex
                -- categoría OTHER STRING -- }
```

C.5.2 Un valor del tipo **ProcessingInstruction** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el primer bit de un octeto y finaliza en el octavo bit de otro octeto.

C.5.3 El valor de **target** se codifica como se describe en C.13.

C.5.4 El valor de **content** se codifica como se describe en C.14.

C.6 Codificación del tipo UnexpandedEntityReference

C.6.1 El tipo `UnexpandedEntityReference` se define en 7.6 como sigue:

```
UnexpandedEntityReference ::= SEQUENCE {
    name IdentifyingStringOrIndex
        -- categoría OTHER NCNAME --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
        -- categoría OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
        -- categoría OTHER URI -- }
```

C.6.2 Un valor del tipo `UnexpandedEntityReference` se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el séptimo bit de un octeto y finaliza en el octavo bit de otro octeto.

C.6.3 Para cada uno de los componentes opcionales `system-identifier` y `public-identifier` (en este orden), si el componente está presente, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.6.4 El valor de `name` se codifica como se describe en C.13.

C.6.5 Si está presente el componente opcional `system-identifier`, se codifica como se describe en C.13.

C.6.6 Si está presente el componente opcional `public-identifier`, se codifica como se describe en C.13.

C.7 Codificación del tipo CharacterChunk

C.7.1 El tipo `CharacterChunk` se define en 7.7 como sigue:

```
CharacterChunk ::= SEQUENCE {
    character-codes NonIdentifyingStringOrIndex
        -- categoría CONTENT CHARACTER CHUNK -- }
```

C.7.2 Un valor del tipo `CharacterChunk` se codifica realizando las acciones siguientes.

NOTA – Una codificación de este tipo siempre se inicia en el tercer bit de un octeto y finaliza en el octavo bit del mismo o de otro octeto.

C.7.3 El valor de `character-codes` se codifica como se describe en C.15.

C.8 Codificación del tipo Comment

C.8.1 El tipo `Comment` se define en 7.8 como sigue:

```
Comment ::= SEQUENCE {
    content NonIdentifyingStringOrIndex -- categoría OTHER STRING --}
```

C.8.2 Un valor del tipo `Comment` se codifica realizando la acción siguiente.

NOTA – Una codificación de este tipo siempre se inicia en el primer bit de un octeto y finaliza en el octavo bit del mismo o de otro octeto.

C.8.3 El valor de `content` se codifica como se describe en C.14.

C.9 Codificación del tipo DocumentTypeDeclaration

C.9.1 El tipo `DocumentTypeDeclaration` se define en 7.9 como sigue:

```
DocumentTypeDeclaration ::= SEQUENCE {
    system-identifier IdentifyingStringOrIndex OPTIONAL
        -- categoría OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
        -- categoría OTHER URI --,
    children SEQUENCE (SIZE(0..MAX)) OF
        ProcessingInstruction }
```

C.9.2 Un valor del tipo `DocumentTypeDeclaration` se codifica realizando las acciones siguientes (en su orden).

NOTA – Una codificación de este tipo siempre se inicia en el séptimo bit de un octeto y finaliza en el cuarto bit de otro octeto (que es el último bit de la terminación '1111' descrita en C.9.7).

C.9.3 Para cada uno de los componentes opcionales `system-identifier` y `public-identifier` (en este orden), si está presente el componente, se añade el bit '1' (presencia) al tren de bits, en otro caso, se añade el bit '0' (ausencia).

- C.9.4** Si está presente el componente opcional **system-identifier**, se codifica como se describe en C.13.
- C.9.5** Si está presente el componente opcional **public-identifier**, se codifica como se describe en C.13.
- C.9.6** Si el componente **children** tiene uno o mas elementos, cada elemento se codifica como sigue: se añaden los ocho bits '1110001' (identificación) al tren de bits y se codifica **ProcessingInstruction** como se describe en C.5.
- C.9.7** Se añaden los cuatro bits '1111' (terminación).

NOTA – Estos bits se añaden incluso cuando el componente **children** no tiene elementos.

C.10 Codificación del tipo **UnparsedEntity**

- C.10.1** El tipo **UnparsedEntity** se define en 7.10 como sigue:

```
UnparsedEntity ::= SEQUENCE {
    name                IdentifyingStringOrIndex
                        -- categoría OTHER NCNAME --,
    system-identifier   IdentifyingStringOrIndex
                        -- categoría OTHER URI --,
    public-identifier   IdentifyingStringOrIndex OPTIONAL
                        -- categoría OTHER URI --,
    notation-name       IdentifyingStringOrIndex
                        -- categoría OTHER NCNAME -- }
```

- C.10.2** Un valor del tipo **UnparsedEntity** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el octavo bit de un octeto y finaliza en el octavo bit de otro octeto.

- C.10.3** Si está presente el componente opcional **public-identifier**, se añade el bit '1' (presencia) al tren de bits, en otro caso, se añade el bit '0' (ausencia).
- C.10.4** El valor de **name** se codifica como se describe en C.13.
- C.10.5** El valor de **system-identifier** se codifica como se describe en C.13.
- C.10.6** Si está presente el componente opcional **public-identifier**, se codifica como se describe en C.13.
- C.10.7** El valor de **name** se codifica como se describe en C.13.

C.11 Codificación del tipo **Notation**

- C.11.1** El tipo **Notation** se define en 7.11 como sigue:

```
Notation ::= SEQUENCE {
    name                IdentifyingStringOrIndex
                        -- categoría OTHER NCNAME --,
    system-identifier   IdentifyingStringOrIndex OPTIONAL
                        -- categoría OTHER URI --,
    public-identifier   IdentifyingStringOrIndex OPTIONAL
                        -- categoría OTHER URI -- }
```

- C.11.2** Un valor del tipo **Notation** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el séptimo bit de un octeto y finaliza en el octavo bit de otro octeto.

- C.11.3** Para cada uno de los componentes opcionales **system-identifier** y **public-identifier** (en este orden), si está presente el componente, se añade el bit '1' (presencia) al tren de bits, en otro caso, se añade el bit '0' (ausencia).
- C.11.4** El valor de **name** se codifica como se describe en C.13.
- C.11.5** Si está presente el componente opcional **system-identifier**, se codifica como se describe en C.13.
- C.11.6** Si está presente el componente opcional **public-identifier**, se codifica como se describe en C.13.

C.12 Codificación del tipo **NamespaceAttribute**

- C.12.1** El tipo **NamespaceAttribute** se define en 7.12 como sigue:

```
NamespaceAttribute ::= SEQUENCE {
    prefix                IdentifyingStringOrIndex OPTIONAL
                        -- categoría PREFIX --,
```

```

namespace-name      IdentifyingStringOrIndex OPTIONAL
                    -- categoría NAMESPACE NAME -- }

```

C.12.2 Un valor del tipo **NamespaceAttribute** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el octavo bit de un octeto y finaliza en el octavo bit de otro octeto.

C.12.3 Si está presente el componente opcional **prefix**, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.12.4 Si está presente el componente opcional **namespace-name**, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.12.5 Si está presente el componente opcional **prefix**, se codifica como se describe en C.13.

C.12.6 Si está presente el componente opcional **namespace-name**, se codifica como se describe en C.13.

C.13 Codificación del tipo **IdentifyingStringOrIndex**

C.13.1 El tipo **IdentifyingStringOrIndex** se define en 7.13 como sigue:

```

IdentifyingStringOrIndex ::= CHOICE {
    literal-character-string  NonEmptyOctetString,
    string-index              INTEGER (1..one-meg) }

```

C.13.2 Un valor del tipo **IdentifyingStringOrIndex** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el primer bit de un octeto y finaliza en el octavo bit del mismo o de otro octeto.

C.13.3 Si está presente la alternativa **literal-character-string**, se añade el bit '0' (discriminador) al tren de bits y se codifica **literal-character-string** como se describe en C.22.

C.13.4 Si está presente la alternativa **string-index**, se añade el bit '1' (discriminador) al tren de bits y se codifica **string-index** como se describe en C.25.

C.14 Codificación del tipo **NonIdentifyingStringOrIndex** comenzando por el primer bit de un octeto

C.14.1 El tipo **NonIdentifyingStringOrIndex** se define en 7.14 como sigue:

```

NonIdentifyingStringOrIndex ::= CHOICE {
    literal-character-string  SEQUENCE {
        add-to-table          BOOLEAN,
        character-string      EncodedCharacterString },
    string-index              INTEGER (0..one-meg) }

```

C.14.2 Esta subcláusula C.14 se invoca para codificar un valor del tipo **NonIdentifyingStringOrIndex** cuando la codificación se inicia en el primer bit de un octeto (véase también C.15). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.14.3 Si está presente la alternativa **literal-character-string**, se añade el bit '0' (discriminador) al tren de bits, y se codifica **literal-character-string** como se describe en las dos subcláusulas siguientes.

C.14.3.1 Si el valor del componente **add-to-table** es **VERDADERO**, entonces se añade el bit '1' al tren de bits, en otro caso se añade el bit '0'.

C.14.3.2 El valor del componente **character-string** se codifica como se describe en C.19.

C.14.4 Si está presente la alternativa **string-index**, se añade el bit '1' (discriminador) al tren de bits y se codifica **string-index** como se describe en C.26.

C.15 Codificación del tipo **NonIdentifyingStringOrIndex** comenzando por el tercer bit de un octeto

C.15.1 El tipo **NonIdentifyingStringOrIndex** se define en 7.14 como sigue:

```

NonIdentifyingStringOrIndex ::= CHOICE {
    literal-character-string  SEQUENCE {
        add-to-table          BOOLEAN,

```

```

        character-string      EncodedCharacterString },
string-index                 INTEGER (0..one-meg) }

```

C.15.2 Esta subcláusula C.15 se invoca para codificar un valor del tipo **NonIdentifyingStringOrIndex** cuando la codificación se inicia en el tercer bit de un octeto (véase también C.14). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.15.3 Si está presente la alternativa **literal-character-string**, se añade el bit '0' (discriminador) al tren de bits y se codifica **literal-character-string** como se describe en las dos subcláusulas siguientes.

C.15.3.1 Si el valor del componente **add-to-table** es **VERDADERO**, se añade el bit '1' al tren de bits, en otro caso, se añade el bit '0'.

C.15.3.2 El valor del componente **character-string** se codifica como se describe en C.20.

C.15.4 Si está presente la alternativa **string-index**, se añade el bit '1' (discriminador) al tren de bits y se codifica **string-index** como se describe en C.28.

C.16 Codificación del tipo **NameSurrogate**

C.16.1 El tipo **NameSurrogate** se define en 7.15 como sigue:

```

NameSurrogate ::= SEQUENCE {
    prefix-string-index          INTEGER(1..one-meg) OPTIONAL,
    namespace-name-string-index INTEGER(1..one-meg) OPTIONAL,
    local-name-string-index     INTEGER(1..one-meg) }
(CONSTRAINED BY {-- prefix-string-index sólo estará presente si
-- existe namespace-name-string-index --})

```

C.16.2 Un valor del tipo **NameSurrogate** se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre se inicia en el séptimo bit de un octeto y finaliza en el octavo bit de otro octeto.

C.16.3 Si está presente el componente opcional **prefix-string-index**, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.16.4 Si está presente el componente opcional **namespace-name-string-index**, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.16.5 Si está presente el componente opcional **prefix-string-index**, se añade el bit '0' (relleno) al tren de bits y se codifica el componente como se describe en C.25.

C.16.6 Si está presente el componente opcional **namespace-name-string-index**, se añade el bit '0' (relleno) al tren de bits y se codifica el componente como se describe en C.25.

C.16.7 El bit '0' (relleno) se añade al tren de bits y se codifica el componente **local-name-string-index** como se describe en C.25.

C.17 Codificación del tipo **QualifiedNameOrIndex** comenzando por el segundo bit de un octeto

C.17.1 El tipo **QualifiedNameOrIndex** se define en 7.16 como sigue:

```

QualifiedNameOrIndex ::= CHOICE {
    literal-qualified-name SEQUENCE {
        prefix          IdentifyingStringOrIndex OPTIONAL
                        -- categoría PREFIX --,
        namespace-name  IdentifyingStringOrIndex OPTIONAL
                        -- categoría NAMESPACE NAME --,
        local-name      IdentifyingStringOrIndex
                        -- categoría LOCAL NAME -- },
    name-surrogate-index INTEGER (1..one-meg) }

```

C.17.2 Esta subcláusula C.17 se invoca para codificar un valor del tipo **QualifiedNameOrIndex** cuando la codificación se inicia en el segundo bit de un octeto (véase también C.18). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.17.3 Si está presente la alternativa `literal-qualified-name`, se añaden los cuatro bits '1111' (identificación) y el bit '0' (relleno) al tren de bits y se codifica `literal-qualified-name` como se describe en las cuatro subcláusulas siguientes.

C.17.3.1 Para cada uno de los componentes opcionales `prefix` y `namespace-name` (en este orden), si está presente el componente, se añade el bit '1' (presencia) al tren de bits, en otro caso, se añade el bit '0' (ausencia).

C.17.3.2 Si está presente el componente opcional `prefix`, se codifica como se describe en C.13.

C.17.3.3 Si está presente el componente opcional `namespace-name`, se codifica como se describe en C.13.

C.17.3.4 El componente `local-name` se codifica como se describe en C.13.

C.17.4 Si está presente la alternativa `name-surrogate-index`, se codifica como se describe en C.25.

C.18 Codificación del tipo `QualifiedNameOrIndex` comenzando por el tercer bit de un octeto

C.18.1 El tipo `QualifiedNameOrIndex` se define en 7.16 como sigue:

```
QualifiedNameOrIndex ::= CHOICE {
    literal-qualified-name SEQUENCE {
        prefix IdentifyingStringOrIndex OPTIONAL
        namespace-name IdentifyingStringOrIndex OPTIONAL
        local-name IdentifyingStringOrIndex
        name-surrogate-index INTEGER (1..one-meg) }
    -- categoría PREFIX --,
    -- categoría NAMESPACE NAME --,
    -- categoría LOCAL NAME -- }
```

C.18.2 Esta subcláusula C.18 se invoca para codificar un valor del tipo `QualifiedNameOrIndex` cuando la codificación se inicia en el tercer bit de un octeto (véase también C.17). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.18.3 Si está presente la alternativa `literal-qualified-name`, se añaden los cuatro bits '1111' (identificación) al tren de bits y se codifica `literal-qualified-name` como se describe en las cuatro subcláusulas siguientes.

C.18.3.1 Para cada uno de los componentes opcionales `prefix` y `namespace-name` (en este orden), si está presente el componente, se añade el bit '1' (presencia) al tren de bits, en otro caso se añade el bit '0' (ausencia).

C.18.3.2 Si está presente el componente opcional `prefix`, se codifica como se describe en C.13.

C.18.3.3 Si está presente el componente opcional `namespace-name`, se codifica como se describe en C.13.

C.18.3.4 El componente `local-name` se codifica como se describe en C.13.

C.18.4 Si está presente la alternativa `name-surrogate-index`, se codifica como se describe en C.27.

C.19 Codificación del tipo `EncodeCharacterString` comenzando por el tercer bit de un octeto

C.19.1 El tipo `EncodeCharacterString` se define en 7.17 como sigue:

```
EncodedCharacterString ::= SEQUENCE {
    encoding-format CHOICE {
        utf-8 NULL,
        utf-16 NULL,
        restricted-alphabet INTEGER (1..256),
        encoding-algorithm INTEGER (1..256) },
    octets NonEmptyOctetString }
```

C.19.2 Esta subcláusula C.19 se invoca para codificar un valor del tipo `EncodeCharacterString` cuando la codificación se inicia en el tercer bit de un octeto (véase también C.20). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit de otro octeto.

C.19.3 El valor del componente `encoding-format` se codifica como se describe en las cuatro subcláusulas siguientes.

C.19.3.1 Si está presente la alternativa `utf-8`, se añaden los dos bits '00' (discriminador) al tren de bits.

C.19.3.2 Si está presente la alternativa `utf-16`, se añaden los dos bits '01' (discriminador) al tren de bits.

C.19.3.3 Si está presente la alternativa **restricted-alphabet**, se añaden los dos bits '10' (discriminador) al tren de bits y se codifica **restricted-alphabet** como se describe en C.29.

C.19.3.4 Si está presente la alternativa **encoding-algorithm**, se añaden los dos bits '11' (discriminador) al tren de bits y se codifica **encoding-algorithm** como se describe en C.29.

C.19.4 El componente **octets** se codifica como se describe en C.23.

C.20 Codificación del tipo EncodeCharacterString comenzando por el quinto bit de un octeto

C.20.1 El tipo **EncodeCharacterString** se define en 7.17 como sigue:

```

EncodedCharacterString ::= SEQUENCE {
    encoding-format      CHOICE {
        utf-8            NULL,
        utf-16           NULL,
        restricted-alphabet  INTEGER(1..256),
        encoding-algorithm  INTEGER(1..256) },
    octets              NonEmptyOctetString }

```

C.20.2 Esta subcláusula C.20 se invoca para codificar un valor del tipo **EncodeCharacterString** cuando la codificación se inicia en el quinto bit de un octeto (véase también C.19). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit de otro octeto.

C.20.3 El valor del componente **encoding-format** se codifica como se describe en las cuatro subcláusulas siguientes.

C.20.3.1 Si está presente la alternativa **utf-8**, se añaden los dos bits '00' (discriminador) al tren de bits.

C.20.3.2 Si está presente la alternativa **utf-16**, se añaden los dos bits '01' (discriminador) al tren de bits.

C.20.3.3 Si está presente la alternativa **restricted-alphabet**, se añaden los dos bits '10' (discriminador) al tren de bits y se codifica **restricted-alphabet** como se describe en C.29.

C.20.3.4 Si está presente la alternativa **encoding-algorithm**, se añaden los dos bits '11' (discriminador) al tren de bits y se codifica **encoding-algorithm** como se describe en C.29.

C.20.4 El componente **octets** se codifica como se describe en C.24.

C.21 Codificación de la longitud de un tipo sequence-of

C.21.1 Esta subcláusula se invoca para codificar la longitud de un tipo **sequence-of** que está codificada con un campo longitud que precede a los elementos del tipo **sequence-of**.

NOTA – Esta codificación siempre se inicia en el primer bit de un octeto y finaliza en el octavo bit del mismo o de otro octeto.

C.21.2 Si el valor se encuentra en la gama de 1 a 128, se añade el bit '0' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de siete bits y se añade.

C.21.3 Si el valor está en la gama de 129 a 2^{20} , se añade el bit '1' y los tres bits '000' (relleno) al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de veinte bits y se añade.

C.22 Codificación del tipo NonEmptyOctetString comenzando por el segundo bit de un octeto

C.22.1 El tipo **NonEmptyOctetString** se define en 7.2 como sigue:

```

NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))

```

C.22.2 Esta subcláusula C.22 se invoca para codificar un valor del tipo **NonEmptyOctetString** cuando la codificación se inicia en el segundo bit de un octeto (véanse también C.23 y C.24). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit de otro octeto.

C.22.3 La longitud de la cadena de octetos se codifica como se describe en las tres subcláusulas siguientes.

C.22.3.1 Si la longitud se encuentra en la gama de 1 a 64, se añade el bit '0' al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de seis bits y se añade.

ISO/CEI 24824-1:2005 (S)

C.22.3.2 Si la longitud se encuentra en la gama de 65 a 320, se añaden los dos bits '10' y los cinco bits '00000' (relleno) al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de ocho bits y se añade.

C.22.3.3 Si la longitud se encuentra en la gama de 321 a 2^{32} , se añaden los dos bits '11' y los cinco bits '00000' (relleno) al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de treinta y dos bits y se añade.

C.22.4 Los bits que forman los octetos de la cadena de octetos se añaden al tren de bits (en orden).

C.23 Codificación del tipo `NonEmptyOctetString` comenzando por el quinto bit de un octeto

C.23.1 El tipo `NonEmptyOctetString` se define en 7.2 como sigue:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

C.23.2 Esta subcláusula C.23 se invoca para codificar un valor del tipo `NonEmptyOctetString` cuando la codificación se inicia en el quinto bit de un octeto (véase también C.22 y C.24). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit de otro octeto.

C.23.3 La longitud de la cadena de octetos se codifica como se describe en las tres subcláusulas siguientes.

C.23.3.1 Si la longitud se encuentra en la gama de 1 a 8, se añade el bit '0' al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de tres bits y se añade.

C.23.3.2 Si la longitud se encuentra en la gama de 9 a 264, se añaden los dos bits '10' y los dos bits '00' (relleno) al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de ocho bits y se añade.

C.23.3.3 Si la longitud se encuentra en la gama de 265 a 2^{32} , se añaden los dos bits '11' y los dos bits '00' (relleno) al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de treinta y dos bits y se añade.

C.23.4 Los bits que forman los octetos de la cadena de octetos se añaden al tren de bits (en orden).

C.24 Codificación del tipo `NonEmptyOctetString` comenzando por el séptimo bit de un octeto

C.24.1 El tipo `NonEmptyOctetString` se define en 7.2 como sigue:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

C.24.2 Esta subcláusula C.24 se invoca para codificar un valor del tipo `NonEmptyOctetString` cuando la codificación se inicia en el séptimo bit de un octeto (véanse también C.22 y C.23). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit de otro octeto.

C.24.3 La longitud de la cadena de octetos se codifica como se describe en las tres subcláusulas siguientes.

C.24.3.1 Si la longitud se encuentra en la gama de 1 a 2, se añade el bit '0' al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de un bit y se añade.

C.24.3.2 Si la longitud se encuentra en la gama de 3 a 258, se añaden los dos bits '10' al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de ocho bits y se añade.

C.24.3.3 Si la longitud se encuentra en la gama de 259 a 2^{32} , se añaden los dos bits '11' al tren de bits y la longitud, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de treinta y dos bits y se añade.

C.24.4 Los bits que forman los octetos de la cadena de octetos se añaden al tren de bits (en orden).

C.25 Codificación de números enteros en la gama de 1 a 2^{20} comenzando por el segundo bit de un octeto

C.25.1 Esta subcláusula C.25 se invoca para codificar un valor de un número entero en la gama de 1 a 2^{20} cuando la codificación se inicia en el segundo bit de un octeto (véanse también C.26, C.27 y C.28). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.25.2 Si el valor se encuentra en la gama de 1 a 64, se añade el bit '0' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de seis bits y se añade.

C.25.3 Si el valor se encuentra en la gama de 65 a 8 256, se añaden los dos bits '10' al tren de bits, y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de trece bits y se añade.

C.25.4 Si el valor se encuentra en la gama de 8257 a 2^{20} , se añaden los dos bits '11' y el bit '0' (relleno) al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de veinte bits y se añade.

C.26 Codificación de números enteros en la gama de 0 a 2^{20} comenzando por el segundo bit de un octeto

C.26.1 Esta subcláusula C.26 se invoca para codificar un valor de un número entero en la gama de 0 a 2^{20} cuando la codificación se inicia en el segundo bit de un octeto (véanse también C.25, C.27 y C.28). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.26.2 Si el valor es cero, se añaden los siete bits '1111111' al tren de bits, en otro caso se codifica el valor como se describe en C.25.

C.27 Codificación de números enteros en la gama de 1 a 2^{20} comenzando por el tercer bit de un octeto

C.27.1 Esta subcláusula C.27 se invoca para codificar un valor de un número entero en la gama de 1 a 2^{20} cuando la codificación se inicia en el tercer bit de un octeto (véanse también C.25, C.26 y C.28). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.27.2 Si el valor se encuentra en la gama de 1 a 32, se añade el bit '0' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de cinco bits y se añade.

C.27.3 Si el valor se encuentra en la gama de 33 a 2080, se añaden los tres bits '100' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de once bits y se añade.

C.27.4 Si el valor se encuentra en la gama de 2081 a 526368, se añaden los tres bits '101' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de diecinueve bits y se añade.

C.27.5 Si el valor se encuentra en la gama de 526369 a 2^{20} , se añaden los tres bits '110' y los siete bits '0000000' (relleno) al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de veinte bits y se añade.

C.28 Codificación de números enteros en la gama de 1 a 2^{20} comenzando por el cuarto bit de un octeto

C.28.1 Esta subcláusula C.28 se invoca para codificar un valor de un número entero en la gama de 1 a 2^{20} cuando la codificación se inicia en el cuarto bit de un octeto (véanse también C.25, C.26 y C.27). El valor se codifica realizando las acciones siguientes (en orden).

NOTA – Una codificación de este tipo siempre finaliza en el octavo bit del mismo o de otro octeto.

C.28.2 Si el valor se encuentra en la gama de 1 a 16, se añade el bit '0' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de cuatro bits y se añade.

C.28.3 Si el valor se encuentra en la gama de 17 a 1040, se añaden los tres bits '100' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de diez bits y se añade.

C.28.4 Si el valor se encuentra en la gama de 1041 a 263184, se añaden los tres bits '101' al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de dieciocho bits y se añade.

C.28.5 Si el valor se encuentra en la gama de 263185 a 2^{20} , se añaden los tres bits '110' y los seis bits '000000' (relleno) al tren de bits y el valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de veinte bits y se añade.

C.29 Codificación de números enteros en la gama de 1 a 256

C.29.1 Esta subcláusula C.29 se invoca para codificar un valor de un número entero en la gama de 1 a 256.

NOTA – Una codificación de este tipo siempre se inicia en el quinto bit o en el séptimo bit de un octeto y finaliza en el cuarto bit o en el sexto bit (respectivamente) del octeto siguiente.

C.29.2 El valor, menos el límite inferior de la gama, se codifica como un número entero sin signo en un campo de ocho bits y se añade al tren de bits.

Anexo D

Ejemplos de codificación de infosets XML como documentos infoset rápidos

(Este anexo no es parte integrante de la presente Recomendación | Norma Internacional)

D.1 Introducción a los ejemplos

D.1.1 Este anexo utiliza los convenios tipográficos siguientes para los números:

- para un número representado en base diez se utiliza **bold Courier** para los dígitos del número, seguido del subíndice "10" (por ejemplo, **11₁₀**); y
- para un número representado en base dieciséis (un número hexadecimal) se utiliza **bold Courier** para los dígitos del número, seguido del subíndice "16" (por ejemplo, **0b1f₁₆**); y
- si se establece explícitamente la base de un número, entonces se omite el subíndice.

D.1.2 Este anexo presenta dos ejemplos de codificaciones posibles de un orden en lenguaje de negocios universal (UBL, *universal business language*) [1] en un documento infoset rápido. UBL está diseñado para proporcionar una sintaxis universalmente comprendida y comercialmente reconocida para documentos oficiales de negocios.

D.1.3 En D.3 se presenta el conjunto infoset XML para el ejemplo de orden UBL.

D.1.4 El primer documento infoset rápido tiene un vocabulario inicial que hace referencia al vocabulario externo. La subcláusula D.4 define el contenido del vocabulario externo y los octetos del documento infoset rápido y explica algunas secuencias de octetos.

D.1.5 El segundo documento infoset rápido no tiene vocabulario inicial. La subcláusula D.5 describe los octetos de dicho documento infoset rápido y explica algunas secuencias de octetos.

NOTA – El vocabulario final de este documento infoset rápido es el mismo que el vocabulario final del documento infoset rápido descrito en D.4.

D.1.6 Los octetos de D.4 y D.5 se representan en una serie de tablas con dos columnas cada una. La primera columna enumera las posiciones iniciales en hexadecimales de 32 octetos consecutivos del documento infoset rápido y la segunda columna enumera los octetos en notación hexadecimal. Se destacan los caracteres hexadecimales que contienen los bits correspondientes a la identificación y terminación de los elementos de información.

D.1.7 Las explicaciones de algunas secuencias de octetos de los documentos infoset rápidos (en D.4 y D.5) se presentan en tablas con las columnas siguientes:

- La columna 1 presenta la posición, en hexadecimal, de los octetos enumerados en la columna 2.
- La columna 2 presenta los octetos del documento infoset rápido asociados con un elemento de información importante y las propiedades del elemento. Un octeto se representa en base 2 seguido por el mismo octeto representado en base 16 (hexadecimal) entre paréntesis, por ejemplo, **11110000 (f0)**.
- La columna 3 presenta una descripción detallada de los octetos de la columna 2 y hace referencia a las subcláusulas del anexo C para su explicación y clarificación.
- La columna 4 presenta una parte del documento infoset XML o una parte del documento XML 1.0 (si procede) correspondiente a los octetos de la columna 2.

D.1.8 En estos ejemplos todos los fragmentos de elementos de información **character** que tengan menos de seis caracteres se añaden a la tabla CONTENT CHARACTER CHUNK y la propiedad **[normalized value]** de todos los elementos de información **attribute** que contienen menos de seis caracteres se añaden a la tabla ATTRIBUTE VALUE.

D.1.9 En D.2 se indican los tamaños del documento XML 1.0 y de los documentos infoset rápidos y los tamaños comprimidos (utilizando GZIP) de dichos documentos.

D.2 Tamaño de los documentos ejemplo (incluida la compresión basada en redundancia)

D.2.1 El cuadro D.1 presenta los tamaños de todos los documentos. La columna 1 enumera los documentos UBL. La columna 2 enumera los tamaños de los documentos y la columna 3 los tamaños de los documentos comprimidos GZIP (con opciones por defecto) [2].

NOTA 1 – El documento XML 1.0 de orden UBL no contiene espacios en blanco (véase D.3.1.2)

NOTA 2 – Para cada documento se codifican todos los caracteres utilizando codificación de caracteres UTF-8.

NOTA 3 – No se serializa ninguna declaración XML (véase 12.3) para los documentos infoset rápidos.

Cuadro D.1 – Tamaños iniciales y tamaños comprimidos GZIP de los documentos

Documento UBL	Tamaño	Tamaño comprimido GZIP
Documento XML 1.0	3 311	8 93
Documento infoset rápido con vocabulario externo	6 84	5 46
Documento infoset rápido sin vocabulario inicial	1 322	8 60

D.2.2 El tamaño del documento infoset rápido con una referencia a un vocabulario externo es el de menor tamaño y también el menor en tamaño cuando se comprime con GZIP. La relación entre el tamaño comprimido GZIP y el tamaño del documento infoset rápido indica que este documento infoset rápido tiene muy poca información redundante.

D.2.3 En todos los casos los tamaños comprimidos GZIP de los documentos infoset rápidos son menores que los tamaños comprimidos GZIP del documento XML 1.0. Es más, el tamaño del documento infoset rápido con una referencia a un vocabulario externo es menor que el tamaño comprimido GZIP del documento XML 1.0.

D.3 Ejemplo de orden UBL

D.3.1 Ejemplo de orden Joinery

D.3.1.1 El ejemplo de orden UBL se toma de [1]. En particular, el ejemplo de orden Joinery se ha elegido (véase [xml/joinery/UBL-Order-1.0-Joinery-Example.xml](#)) por las razones siguientes:

- es un ejemplo real desarrollado independientemente de la presente Recomendación | Norma Internacional sin ninguna relación particular con conjuntos de información rápidos;
- es de libre disposición; y
- hace un uso amplio de espacios de nombres XML, lo que constituye un buen ejemplo para presentar cómo soporta infoset rápido los espacios de nombres XML.

D.3.1.2 El ejemplo de orden Joinery se ha modificado en lo siguiente:

- los últimos tres elementos **OrderLine** se han suprimido; y
NOTA 1 – Esto reduce el documento XML 1.0 a un tamaño reutilizable para su presentación en la presente Recomendación | Norma Internacional.
- se han suprimido todos los espacios en blanco.
NOTA 2 – Esto representa un caso de utilización más realista para conjuntos de información que se pueden poner en serie, transmitir por la red y relacionar.

D.3.2 Documento XML 1.0 de orden Joinery

El documento XML 1.0 de orden Joinery, con las modificaciones establecidas en D.3.1.2 a) pero sin espacios en blanco para que sea más legible, se presenta como sigue:

```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns:res="urn:oasis:names:tc:ubl:codelist:AcknowledgementResponseCode:1:0"
xmlns:cbc="urn:oasis:names:tc:ubl:CommonBasicComponents:1:0"
xmlns:cac="urn:oasis:names:tc:ubl:CommonAggregateComponents:1:0"
xmlns:cur="urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:oasis:names:tc:ubl:Order:1:0"
xsi:schemaLocation="urn:oasis:names:tc:ubl:Order:1:0 ../xsd/maindoc/UBL-Order-1.0.xsd">
  <BuyersID>S03-034257</BuyersID>
  <cbc:IssueDate>2003-02-03</cbc:IssueDate>
  <cac:BuyerParty>
    <cac:Party>
      <cac:PartyName>
        <cbc:Name>Jerry Builder plc</cbc:Name>
      </cac:PartyName>
      <cac:Address>
        <cbc:StreetName>Marsh Lane</cbc:StreetName>
        <cbc:CityName>Nowhere</cbc:CityName>
        <cbc:PostalZone>NR18 4XX</cbc:PostalZone>
        <cbc:CountrySubentity>Norfolk</cbc:CountrySubentity>
      </cac:Address>
      <cac:Contact>
        <cbc:Name>Eva Brick</cbc:Name>
      </cac:Contact>
    </cac:Party>
  </cac:BuyerParty>
  <cac:SellerParty>
    <cac:Party>
```

```

<cac:PartyName>
  <cbc:Name>Specialist Windows plc</cbc:Name>
</cac:PartyName>
<cac:Address>
  <cbc:BuildingName>Snowhill Works</cbc:BuildingName>
  <cbc:CityName>Little Snoring</cbc:CityName>
  <cbc:PostalZone>SM2 3NW</cbc:PostalZone>
  <cbc:CountrySubentity>Whereshire</cbc:CountrySubentity>
</cac:Address>
</cac:Party>
</cac:SellerParty>
<cac:Delivery>
  <cbc:RequestedDeliveryDateTime>2003-02-24T00:00:00</cbc:RequestedDeliveryDateTime>
  <cac:DeliveryAddress>
    <cbc:StreetName>Riverside Rd.</cbc:StreetName>
    <cbc:BuildingName>Plot 17, Whitewater Estate</cbc:BuildingName>
    <cbc:CityName>Whetstone</cbc:CityName>
    <cbc:CountrySubentity>Middlesex</cbc:CountrySubentity>
  </cac:DeliveryAddress>
</cac:Delivery>
<cac:OrderLine>
  <cac:LinItem>
    <cac:BuyersID>A</cac:BuyersID>
    <cbc:Quantity quantityUnitCode="unit">2</cbc:Quantity>
    <cac:Item>
      <cac:SellersItemIdentification>
        <cac:ID>236WV</cac:ID>
        <cac:PhysicalAttribute>
          <cac:AttributeID>wood</cac:AttributeID>
          <cbc:Description>soft</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>finish</cac:AttributeID>
          <cbc:Description>primed</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>fittings</cac:AttributeID>
          <cbc:Description>satin</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>glazing</cac:AttributeID>
          <cbc:Description>single</cbc:Description>
        </cac:PhysicalAttribute>
      </cac:SellersItemIdentification>
    </cac:Item>
  </cac:LinItem>
</cac:OrderLine>
<cac:OrderLine>
  <cac:LinItem>
    <cac:BuyersID>B</cac:BuyersID>
    <cbc:Quantity quantityUnitCode="unit">3</cbc:Quantity>
    <cac:Item>
      <cac:SellersItemIdentification>
        <cac:ID>340TW</cac:ID>
        <cac:PhysicalAttribute>
          <cac:AttributeID>hand</cac:AttributeID>
          <cbc:Description>RH</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>wood</cac:AttributeID>
          <cbc:Description>hard</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>finish</cac:AttributeID>
          <cbc:Description>stain</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>fittings</cac:AttributeID>
          <cbc:Description>brass</cbc:Description>
        </cac:PhysicalAttribute>
        <cac:PhysicalAttribute>
          <cac:AttributeID>glazing</cac:AttributeID>
          <cbc:Description>double</cbc:Description>
        </cac:PhysicalAttribute>
      </cac:SellersItemIdentification>
    </cac:Item>
  </cac:LinItem>
</cac:OrderLine>

```

```

                </cac:SellersItemIdentification>
            </cac:Item>
        </cac:LineItem>
    </cac:OrderLine>
</Order>

```

D.4 Documento infosef rápido de orden UBL con vocabulario externo

En D.4.1 se presenta el vocabulario externo del documento infosef rápido. Los octetos (como caracteres hexadecimales) del documento infosef rápido se presentan en D.4.2. En D.4.3 se ofrecen explicaciones detalladas de algunas secuencias de los octetos de D.4.2. El documento infosef rápido no se puede considerar como autocontenido puesto que se requiere información externa (el vocabulario externo) para generar un conjunto de información XML completo.

NOTA – El documento infosef rápido todavía puede procesarse mediante un relacionador infosef rápido que no pueda obtener las tablas de vocabulario dadas en URI, pero no se puede quitar la referencia a índices de la tabla de vocabulario para obtener la información necesaria y generar las propiedades de los elementos de información.

D.4.1 El vocabulario externo de orden UBL

D.4.1.1 El vocabulario externo del documento infosef rápido está especificado para que sea el vocabulario final obtenido del ejemplo infosef XML de orden UBL (véase D.3.1.2) modificado posteriormente para que:

- no tenga elementos de información **character**; y
- tenga propiedades [**normalized value**] vacías de los elementos de información **attribute**.

NOTA 1 – Esto representa un caso real en el que no se conoce con antelación cuál será el contenido definido por la aplicación (elementos de información **character** y/o propiedades [**normalized value**] de los elementos de información **attribute**) de un conjunto de información XML.

NOTA 2 – En la práctica no se espera que el documento a serializar se utilice para generar el vocabulario externo. Se prevé que las herramientas utilicen el esquema y puedan utilizar instancias infosef XML del esquema para el análisis de las frecuencias de las cadenas y se atribuirán nombres cualificados de forma que se asignen los valores de índice menores a la información que se produce con mayor frecuencia (por ejemplo, la frecuencia de las propiedades [**local name**] en conjuntos de información XML puede seguir una serie exponencial).

D.4.1.2 El URI del vocabulario externo es **urn:oasis:names:tc:ubl:Order:1.0:joinery:example**.

D.4.1.3 El cuadro D.2 presenta el vocabulario del conjunto de información XML de orden UBL (las tablas de vocabulario). La columna 1 indica los índices de tabla de vocabulario de las tablas de vocabulario (índice), la columna 2 indica los datos de entrada a la tabla de vocabulario de la tabla PREFIX (dato de entrada del prefijo), la columna 3 los datos de entrada a la tabla de vocabulario de la tabla NAMESPACE NAME (dato de entrada del nombre de espacio de nombres), la columna 4 los datos de entrada a la tabla LOCAL NAME (dato de entrada del nombre local), la columna 5 los datos de entrada a la tabla de vocabulario de la tabla ELEMENT NAME (dato de entrada del nombre de elemento) y la columna 6 los datos de entrada a la tabla ATTRIBUTE NAME (dato de entrada del nombre de atributo). Los valores de índice para los datos de entrada de name surrogate de las tablas ELEMENT NAME y ATTRIBUTE NAME se presentan en el orden especificado por los componentes del tipo **NameSurrogate** (**prefix-name-string-index**, **namespace-name-string-index** y **local-name-string-index**). Un carácter "_" especifica que el valor está ausente (lo que sólo se produce para valores de los componentes **prefix-name-string-index** y **namespace-name-string-index**).

NOTA 1 – El primer dato de entrada (índice 1) para el prefijo y el nombre de espacio de nombres correspondiente al prefijo XML, "xml", y el nombre de espacio de nombres XML, "<http://www.w3.org/XML/1998/namespace>", están incorporados (véanse 7.2.21 y 7.2.22).

NOTA 2 – Se han recortado los datos de entrada de nombre de espacios de nombres largos (URI).

NOTA 3 – Para el primer dato de entrada de nombres de elemento (índice 1) no hay referencia a un prefijo (puesto que el valor está ausente, representado por "-"), hay una referencia al séptimo dato de entrada de nombre de espacio de nombres (índice 7) para la propiedad [**namespace name**] ("urn:oasis:names:tc:ubl:Order:1.0") y existe una referencia al primer dato de entrada de nombre local (índice 1) para la propiedad [**local name**] ("Order").

Cuadro D.2 – Vocabulario para el conjunto de información XML de orden UBL

Índice	Prefijo	Nombre de espacio de nombres	Nombre local	Nombre de elemento	Nombre de atributo
1	xml	http://www.w3.org/XML/1998/namespace	Order	_ 7 1	6 6 2
2	resAcknowledgementResponseCode:1:0	schemaLocation	_ 7 3	_ _ 23
3	cbcCommonBasicComponents:1:0	BuyersID	3 3 4	
4	cacCommonAggregateComponents:1:0	IssueDate	4 4 5	
5	curCurrencyCode:1:0	BuyerParty	4 4 6	
6	xsiXMLSchema-instance	Party	4 4 7	
7	Order:1:0	PartyName	3 3 8	
8			Name	4 4 9	
9			Address	3 3 10	
10			StreetName	3 3 11	
11			CityName	3 3 12	
12			PostalZone	3 3 13	
13			CountrySubentity	4 4 14	
14			Contact	4 4 15	
15			SellerParty	3 3 16	
16			BuildingName	4 4 17	
17			Delivery	3 3 18	
18			RequestedDeliveryDateTime	4 4 19	
19			DeliveryAddress	4 4 20	
20			OrderLine	4 4 21	
21			LineItem	4 4 3	
22			Quantity	3 3 22	
23			quantityUnitCode	4 4 24	
24			Item	4 4 25	
25			SellersItemIdentification	4 4 26	
26			ID	4 4 27	
27			PhysicalAttribute	4 4 28	
28			AttributeID	3 3 29	
29			Description		

D.4.2 Octetos (como caracteres hexadecimales) del documento infoset rápido

El cuadro D.3 presenta los octetos del documento infoset rápido para el ejemplo de orden UBL presentado en D.3.

NOTA – Los caracteres hexadecimales que contienen bits que corresponden a la identificación y terminación de elementos de información están subrayados.

Cuadro D.3 – Octetos (como caracteres hexadecimales) del documento infoset rápido

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100002010002f75726e3a6f617369733a6e616d65733a74633a75626c3a4f
000020	726465723a313a303a6a6f696e6572793a6578616d706c6578cf8181cf8282cf
000040	8383cf8484cf8585cd86f00000083b75726e3a6f617369733a6e616d65733a74
000060	633a75626c3a4f726465723a313a30202e2e2f2e2e2f7873642f6d61696e646f
000080	632f55424c2d4f726465722d312e302e787364f00182075330332d3033343235
0000a0	37f00282073230332d30322d3033f003040506820e4a65727279204275696c
0000c0	64657220706c63ff070882074d61727368204c616e65f00982044e6f77686572
0000e0	65f00a82054e52313820345858f00b82044e6f72666f6c6bfff0c068206457661
000100	20427269636bffff0d04050682135370656369616c6973742057696e646f7773
000120	20706c63ff070e820b536e6f7768696c6c20576f726b73f009820b4c6974746c
000140	6520536e6f72696e67f00a8204534d3220334e57f00b82075768657265736869
000160	7265ffff0f108210323030332d30322d32345430303a30303a3030f01108820a
000180	5269766572736964652052642ef00e8217506c6f742031372c20576869746577
0001a0	6174657220457374617465f00982065768657473746f6e65f00b82064d696464
0001c0	6c65736578fff01213149041f0550143756e6974f09032f01617189202323336
0001e0	5756f0191a9201776f6f64f01b9201736f6674ff191a820366696e697368f01b
000200	82037072696d6564ff191a820566697474696e6773f01b9202736174696eff19
000220	1a8204676c617a696e67f01b820373696e676c65fffff1213149042f0550180
000240	f09033f016171892023334305457f0191a920168616e64f01b915248ff191aa3
000260	f01b920168617264ff191a820366696e697368f01b9202737461696eff191a82
000280	0566697474696e6773f01b92026272617373ff191a8204676c617a696e67f01b
0002a0	8203646f75626c65fffff
0002ac	

D.4.3 Explicación de la codificación

D.4.3.1 Codificación del elemento de información document y del elemento de información Order element

La explicación siguiente detalla la codificación inicial del documento infoset rápido (incluido el URI del vocabulario externo) y el elemento de información raíz **element**. En particular, se exponen la codificación de un elemento de información **document**, una secuencia de elementos de información **namespace**, un elemento de información **element** y un elemento de información **attribute**. El cuadro D.4 presenta el fragmento del documento infoset rápido para la codificación del elemento de información **document** y del elemento de información **Order element** de D.3.2. El cuadro D.5 indica la codificación. El fragmento en XML 1.0 se presenta como sigue:

```
<Order xmlns:res="urn:oasis:names:tc:ubl:odelist:AcknowledgementResponseCode:1:0"
xmlns:cbc="urn:oasis:names:tc:ubl:CommonBasicComponents:1:0"
xmlns:cac="urn:oasis:names:tc:ubl:CommonAggregateComponents:1:0"
xmlns:cur="urn:oasis:names:tc:ubl:odelist:CurrencyCode:1:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:oasis:names:tc:ubl:Order:1:0"
xsi:schemaLocation="urn:oasis:names:tc:ubl:Order:1:0 ../xsd/maindoc/UBL-Order-1.0.xsd">
```

Cuadro D.4 – Octetos (como caracteres hexadecimales) del fragmento

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100002010002f75726e3a6f617369733a6e616d65733a74633a75626c3a4f
000020	726465723a313a303a6a6f696e6572793a6578616d706c6578cf8181cf8282cf
000040	8383cf8484cf8585cd86f00000083b75726e3a6f617369733a6e616d65733a74
000060	633a75626c3a4f726465723a313a30202e2e2f2e2e2f7873642f6d61696e646f
000080	632f55424c2d4f726465722d312e302e787364f0

Cuadro D.5 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
00 01	11100000 (e0) 00000000 (00)	Los octetos están presentes al principio de cada documento infoset rápido (véase 12.6)	Elemento de información document
02 03	00000000 (00) 00000001 (01)	Los octetos son la codificación del número de versión (véase 12.9).	
04 05 06	00100000 (20) 00010000 (10) 00000000 (00)	<p>Los octetos son la codificación de la presencia de un vocabulario inicial y de una referencia a un vocabulario externo al vocabulario inicial.</p> <p>El octeto en la posición 04₁₆, valor 20₁₆, tiene un '0' (relleno) para el primer bit (véase 12.8). El tercer bit es '1' indicando que está presente el componente initial-vocabulary y que están ausentes los otros seis componentes opcionales (véase C.2.3).</p> <p>El octeto en la posición 05₁₆, valor 10₁₆, tiene tres '0' (relleno) para los tres primeros bits (véase C.2.5). El cuarto bit es '1' indicando que está presente el external-vocabulary del initial-vocabulary. Los últimos cuatro bits son '0' (del bit quinto al octavo) indicando que cuatro de los restantes doce componentes opcionales están ausentes (véase C.2.5.1).</p> <p>El octeto en la posición 06₁₆, valor 00₁₆, tiene '0' para todos los bits indicando que los últimos ocho de los doce componentes opcionales están ausentes (véase C.2.5.1).</p>	
07 08 37	00101111 (2F) 01110101 (75) 01100101 (65)	<p>Los octetos son la codificación del URI del vocabulario externo.</p> <p>El octeto en la posición 07₁₆, valor 2F₁₆, tiene un '0' (relleno) para el primer bit (véase C.2.5.2). El URI se codifica como caracteres UTF-8 (véase C.22). El segundo bit es '0' indicando que la longitud del URI es superior o igual a 1₁₀ e inferior o igual a 64₁₀ octetos, y que la longitud, menos el límite inferior, se codifica en los bits tercero a octavo como un entero sin signo (véase C.22.3.1). El entero sin signo es 47₁₀ y la longitud es 48₁₀ (el límite inferior es 1).</p> <p>Los 48₁₀ octetos de los caracteres codificados UTF-8 (del URI) se codifican entre el octeto en la posición 08₁₆ y el octeto en la posición 37₁₆.</p>	
38	01111000 (78)	<p>El octeto es la codificación inicial de un vástago del elemento de información document.</p> <p>El octeto en la posición 38₁₆, valor 78₁₆, tiene un '0' (identificación) para el primer bit indicando que existe un vástago del elemento de información document, y que es un elemento de información element (véase C.2.11.2). El segundo bit es '1' indicando que el elemento de información element tiene atributos (véase C.3.3). Los bits tercero al sexto son '1110' seguidos por '00' (relleno) en los bits séptimo y octavo, indicando que están presentes elementos de información attribute de espacio de nombres (véase C.3.4.1).</p>	Elemento de información element con la propiedad [namespace attribute]

Cuadro D.5 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
39 3a 3b	11001111 (cF) 10000001 (81) 10000001 (81)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con propiedades de indexación [prefix] y [normalized value].</p> <p>El octeto en la posición 39₁₆, valor cF₁₆, tiene '110011' (identificación) para los bits primero a sexto (del primer bit al quinto) indicando que está presente un elemento de información attribute de espacio de nombres (véase C.3.4.2). El séptimo bit es '1' indicando que está presente la propiedad [prefix]. El octavo bit es '1' indicando que está presente la propiedad [normalized value].</p> <p>El octeto en la posición 3a₁₆, valor 81₁₆, tiene '1' para el primer bit indicando que un índice está codificado, y el índice en la tabla PREFIX identificará la propiedad [prefix] (véase C.13.4). El segundo bit es '0' indicando que el índice es superior o igual a 1₁₀ e inferior o igual a 64₁₀, y el índice se codifica en los bits tercero a octavo como un número entero sin signo (véase C.25.2). El entero sin signo es 1₁₀ y el índice es 2₁₀ (el límite inferior es 1₁₀), con el resultado de que la propiedad [prefix] es "res" cuando se quita la referencia en la tabla PREFIX.</p> <p>El octeto en la posición 3b₁₆, valor 81₁₆, tiene '1' para el primer bit indicando que se ha codificado un índice, y el índice en la tabla NAMESPACE NAME identificará la propiedad [normalized value] (véase C.13.4). El segundo bit es '0' indicando que el índice es superior o igual a 1₁₀ e inferior o igual a 64₁₀, y el índice se codifica en los bits tercero a octavo como un número entero sin signo (véase C.25.2). El entero sin signo es 1₁₀ y el índice es 2₁₀ (el límite inferior es 1₁₀), con el resultado de que la propiedad [normalized value] es ".ResponseCode:1.0" cuando se quita la referencia en la tabla NAMESPACE NAME.</p>	xmlns:res= "....ResponseCode:1:0"
3c 3d 3e	11001111 (cF) 10000010 (82) 10000010 (82)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con las propiedades indexadas [prefix] y [normalized value].</p> <p>El índice de la propiedad [prefix] es 3₁₀, lo que resulta en un valor de "cbc" se quita la referencia en la tabla PREFIX.</p> <p>El índice de la propiedad [normalized value] es 3₁₀, lo que resulta en un valor de "....sicComponents:1.0" cuando se quita la referencia en la tabla NAMESPACE NAME.</p>	xmlns:cbc= "....sicComponents:1:0"
3f 40 41	11001111 (cF) 10000011 (83) 10000011 (83)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con las propiedades indexadas [prefix] y [normalized value].</p>	xmlns:cac= "....ateComponents:1:0"
42 43 44	11001111 (cF) 10000100 (84) 10000100 (84)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con las propiedades indexadas [prefix] y [normalized value].</p>	xmlns:cur= "....CurrencyCode:1:0"
45 46 47	11001111 (cF) 10000101 (85) 10000101 (85)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con las propiedades indexadas [prefix] y [normalized value].</p>	xmlns:xsi= "....Schema-instance"
48 49	11001101 (cd) 10000110 (86)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con una propiedad indexada [normalized value].</p> <p>El octeto en la posición 48₁₆, valor cd₁₆, tiene un séptimo bit '0' indicando que está ausente la propiedad [prefix] y un octavo bit '1' indicando que está presente la propiedad [normalized value].</p>	xmlns="....Order:1:0"
4a	11110000 (f0)	<p>El octeto es la codificación del terminador para la secuencia de los elementos de información attribute de espacio de nombres.</p> <p>El octeto en la posición 4a₁₆, valor f0₁₆, tiene '1111' (terminación) para los primeros cuatro bits (del primer bit al cuarto) y es el terminador de la secuencia. Cuatro de los seis '0' (relleno) están presentes en los bits quinto a octavo (véase C.3.4.3).</p>	

Cuadro D.5 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
4b	<u>00000000</u> (00)	<p>El octeto es la codificación de un nombre cualificado indexado del elemento de información element.</p> <p>El octeto en la posición 4b₁₆, valor, 00₁₆, tiene los dos últimos de los seis '0' (relleno) en el primero y segundo bits (véase C.3.4.3). El tercer bit es '0' indicando que el nombre cualificado no es un nombre cualificado literal (véase C.18.3) y está indexado. El índice es superior o igual a 1₁₀ e inferior o igual a 32₁₀, y el índice se codifica en los bits cuarto a octavo como un número entero sin signo (véase C.27.2). El entero sin signo es 0₁₀ y el índice es 1₁₀ (el límite inferior es 1₁₀), lo que resulta en un nombre cualificado con una propiedad [namespace name] "...Order:1.0" y una propiedad [local name] "Order" (no existe propiedad [prefix] para este nombre cualificado) cuando se quita la referencia en la tabla ELEMENT NAME.</p>	<Order ...
4c 4d 4e 4f 92	00000000 (00) <u>00001000</u> (08) 01111011 (3b) 01110101 (75) 01101000 (64)	<p>Los octetos son la codificación de un elemento de información attribute con un nombre cualificado indexado y una propiedad [normalized value]. La presencia de elementos de información attribute se indicó en el octeto en la posición 38₁₆ (el segundo bit es '1').</p> <p>El octeto en la posición 4c₁₆, valor 00₁₆, tiene un primer bit '0' (identificación) que indica que está presente un elemento de información attribute (véase C.3.6.1). El segundo bit es '0' indicando que el nombre cualificado no es un nombre cualificado literal (véase C.17.3) y está indexado. El índice es superior o igual a 1₁₀ e inferior o igual a 64₁₀, y el índice está codificado en los bits tercero a octavo como un entero sin signo (véase C.25.2). El entero sin signo es 0₁₀ y el índice es 1₁₀ (el límite inferior es 1₁₀), lo que resulta en un nombre cualificado con una propiedad [prefix] "xsi", una propiedad [namespace name] "...Schema-instance" y una propiedad [local name] "schemaLocation" cuando se quita la referencia en la tala ATTRIBUTE NAME.</p> <p>El octeto en la posición 4d₁₆, valor 08₁₆, es la codificación inicial de una cadena o índice que no identifica (véase C.14) para la propiedad [normalized value]. El primer bit es '0' indicando que está presente una cadena de caracteres literal (véase C.14.3). El segundo bit es '0' indicando que la cadena de caracteres literal no se añadirá la tabla ATTRIBUTE VALUE. Los bits tercero y cuarto, ambos '0', indican que el formato de codificación de la cadena es UTF-8 (véase C.19.3.1). El quinto y sexto bits son '1' y '0' respectivamente indicando que la longitud de los octetos de los caracteres UTF-8 codificados (la propiedad [normalized value]) es superior o igual a 9₁₀ octetos e inferior o igual a 264₁₀ octetos y que la longitud, menos el límite inferior, se codifica en ocho bits del octeto siguiente como un número entero sin signo (véase C.23.3.2). Los bits séptimo y octavo son '0' (relleno) (véase C.23.3.2).</p> <p>El octeto en la posición 4e₁₆, valor 3b₁₆, es la codificación del entero sin signo. La longitud de octetos de los caracteres codificados UTF-8 es 68₁₀ (el límite inferior es 9₁₀).</p> <p>Los 68₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición 4e₁₆ y el octeto en la posición 92₁₆.</p>	xsi:schemaLocation="...."
93	<u>11110000</u> (f0)	<p>El octeto es la codificación del terminador para la secuencia de elementos de información attribute.</p> <p>El octeto en la posición 93₁₆, valor f0₁₆, tiene '1111' para los cuatro primeros bits (los bits primero a cuarto) y es el terminador para la secuencia. Están presentes cuatro '0' (relleno) (bits quinto a octavo) puesto que el elemento de información Order element tiene vástagos (véase D.3.2).</p>	

D.4.3.2 Codificación del elemento de información Address element del elemento de información BuyerParty Element

La explicación siguiente detalla la codificación del elemento de información **Address element** del elemento de información **BuyerParty element** del documento infosec rápido. En particular, se explica la codificación de elementos de información **element** y de elementos de información **character**. El cuadro D.6 presenta el fragmento del documento infosec rápido para la codificación del elemento de información **Address element** del elemento de información **BuyerParty element** de D.3.2. El cuadro D.7 detalla su codificación. El fragmento en XML 1.0 se presenta como sigue:

```
<cac:Address>
  <cbc:StreetName>Marsh Lane</cbc:StreetName>
  <cbc:CityName>Nowhere</cbc:CityName>
  <cbc:PostalZone>NR18 4XX</cbc:PostalZone>
  <cbc:CountrySubentity>Norfolk</cbc:CountrySubentity>
</cac:Address>
```

Cuadro D.6 – Octetos (como caracteres hexadecimales) del fragmento

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
0000c0	070882074d61727368204c616e65f00982044e6f77686572
0000e0	65f00a82054e52313820345858f00b82044e6f72666f6c6bff

Cuadro D.7 – Detalles de codificación

Octeto(s)	Descripción	Infosec XML o XML
c8 00001111 (07)	El octeto es la codificación del elemento de información Address element . El octeto en la posición c8 ₁₆ , valor 07 ₁₆ , tiene un '0' (identificación) para el primer bit indicando que existe un vástago para un elemento de información element (vástago del elemento de información Party element), y que ese vástago es un elemento de información element (véase C.3.7.2). El segundo bit es '0' indicando que el elemento de información element no tiene atributos (véase C.3.3). El tercer bit es '0' indicando que el nombre cualificado no es un nombre cualificado literal (véase C.18.3) y está indexado. El índice es superior o igual a 1 ₁₀ e inferior o igual a 32 ₁₀ y el índice se codifica en los bits cuarto a octavo como un número entero sin signo (véase C.27.2). El entero sin signo es 7 ₁₀ y el índice es 8 ₁₀ (el límite inferior es 1 ₁₀), lo que resulta en un nombre cualificado con una propiedad [prefix] de "cac", una propiedad [namespace name] de "...gateComponents:1.0" y una propiedad [local name] de "Address" cuando se quita la referencia en la tabla ELEMENT NAME.	<cac:Address>
c9 00001000 (08)	El octeto es la codificación del elemento de información StreetName element . El elemento de información element tiene un índice de 9 ₁₀ , lo que resulta en un nombre cualificado con una propiedad [prefix] de "cbc", una propiedad [namespace name] "...BasicComponents:1.0" y una propiedad [local name] "StreetName" cuando se quita la referencia en la tabla ELEMENT NAME .	<cbc:StreetName>

Cuadro D.7 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
ca cb cc d5	10000010 (82) 00000111 (07) 01001101 (4d) 01100101 (65)	<p>Los octetos son la codificación de elementos de información character del elemento de información StreetName element.</p> <p>El octeto en la posición ca₁₆, valor 82₁₆, tiene '10' (identificación) para los dos primeros bits (los bits primero y segundo) indicando que existe un vástago del elemento de información element (vástago del elemento de información StreetName element), y el vástago es un conjunto de elementos de información character (véase C.3.7.5). El tercer bit es '0' indicando que está presente una cadena de caracteres literales (véase C.15.3). El cuarto bit es '0' indicando que la cadena de caracteres literales no se añadirá la tabla CONTENT CHARACTER CHUNK. Los bits quinto y sexto, ambos '0', indican que el formato de codificación del fragmento es UTF-8 (véase C.20.3.1). Los bits séptimo y octavo son '1' y '0' respectivamente indicando que la longitud de los octetos de los caracteres codificados UTF-8 (el fragmento de los elementos de información character) es superior o igual a 3₁₀ octetos e inferior o igual a 258₁₀ octetos, y que la longitud menos el límite inferior está codificada en ocho bits del octeto siguiente como un número entero sin signo (véase C.24.3.2).</p> <p>El octeto en la posición cb₁₆, valor 07₁₆, es el número entero sin signo. La longitud de los octetos de los caracteres codificados UTF-8 es 10₁₀ (el límite inferior es 3₁₀).</p> <p>Los 10₁₀ octetos de los caracteres codificados UTF-8 se codifican entre el octeto en la posición cc₁₀ y el octeto en la posición d5₁₀.</p>	Elemento de información character "Marsh Lane"
d6	11110000 (f0)	<p>El octeto es el terminador del elemento de información StreetName element.</p> <p>El octeto en la posición d6₁₆, valor f0₁₆, tiene '1111' (terminador) para los cuatro primeros bits (bits primero a cuarto) y es el terminador para el elemento de información StreetName element (véase C.3.8). Los bits quinto a octavo son '0' (relleno) puesto que se genera un vástago mas (par) (elemento de información CityName element) (véase C.3.7.1).</p>	</cbc:StreetName>
d7	00001001 (09)	<p>El octeto es la codificación del elemento de información CityName element.</p> <p>El elemento de información element tiene un índice de 10₁₀, lo que resulta en un nombre cualificado con una propiedad [prefix] de "cbc", una propiedad [namespace name] de "...BasicComponents:1:0" y una propiedad [local name] de "CityName" cuando se quita la referencia en la tabla ELEMENT NAME.</p>	<cbc:CityName>
d8 d9 da e0	10000010 (82) 00000100 (04) 01001110 (4e) 01100101 (65)	<p>Los octetos son la codificación de elementos de información character del elemento de información CityName element.</p> <p>Los octetos 7₁₀ de los caracteres codificados UTF-8 se codifican entre el octeto en la posición da₁₆ y el octeto en la posición e0₁₆.</p>	Elemento de información character "Nowhere"
e1	11110000 (f0)	<p>El octeto es el terminador del elemento de información CityName element.</p>	</cbc:CityName>
e2	00001010 (0a)	<p>El octeto es la codificación del elemento de información PostalZone element.</p> <p>El elemento de información element tiene un índice de 11₁₀, lo que resulta en un nombre cualificado con una propiedad [prefix] de "cbc", una propiedad [namespace name] de "...BasicComponents:1:0" y una propiedad [local name] de "PostalZone" cuando se quita la referencia en la tabla ELEMENT NAME.</p>	<cbc:PostalZone>
e3 e4 e5 ec	10000010 (82) 00000101 (05) 01001110 (4e) 01011000 (58)	<p>Los octetos son la codificación de los elementos de información character del elemento de información PostalZone element.</p> <p>Los octetos 8₁₀ de los caracteres codificados UTF-8 se codifican entre el octeto en la posición e5₁₆ y el octeto en la posición ec₁₆.</p>	Elemento de información character "NR18 4XX"
ed	11110000 (f0)	<p>El octeto es el terminador del elemento de información PostalZone element.</p>	</cbc:PostalZone>

Cuadro D.7 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
ee	00001011 (0b)	El octeto es la codificación del elemento de información CountrySubentity element . El elemento de información element tiene un índice de 12 ₁₀ , lo que resulta en un nombre cualificado con una propiedad [prefix] de "cbc", una propiedad [namespace name] de "...BasicComponents:1:0" y una propiedad [local name] de "CountrySubentity" cuando se quita la referencia en la tabla ELEMENT NAME.	<cbc:CountrySubentity>
ef f0 f1 ... f7	10000010 (82) 00000100 (04) 01001110 (4e) ... 01101011 (6b)	Los octetos son la codificación de los elementos de información character del elemento de información CountrySubentity element . Los octetos 7 ₁₀ de los caracteres codificados UTF-8 están codificados entre el octeto en la posición f1 ₁₆ y el octeto en la posición f7 ₁₆ .	Elemento de información character "Norfolk"
f8	11111111 (ff)	El octeto es el terminador del elemento de información CountrySubentity element y del elemento de información Address element . El octeto en la posición f8 ₁₆ , valor ff ₁₆ , tiene '1111' (terminador) para los cuatro primeros bits (los bits primero a cuarto) y es el terminador para el elemento de información CountrySubentity element (véase C.3.8). Los últimos cuatro bits (bits quinto a octavo) son '1111' y son el terminador para el elemento de información Address element (véase C.3.8).	</cbc:CountrySubentity> </cac:Address>

D.5 Documento infoset rápido de orden UBL sin vocabulario inicial

Los octetos (en caracteres hexadecimales) del documento infoset rápido se presentan en D.5.1. En D.5.2 se presentan explicaciones detalladas de algunas secuencias de octetos de D.5.1. El vocabulario final de este documento infoset rápido y del anterior documento infoset rápido será el mismo puesto que los índices de la tabla de vocabulario de las tablas en el vocabulario externo se generan en el mismo orden. Puesto que las cadenas están integradas en el documento infoset rápido este tendrá un tamaño mayor. El coste de incluir las cadenas es de 635₁₀ bytes (el tamaño de este documento infoset rápido menos el tamaño del documento infoset anterior), lo que supone aproximadamente la mitad del tamaño del documento (para documentos más grandes esta diferencia debería ser inferior puesto que el vocabulario tenderá a tener un tamaño fijo). A diferencia del documento infoset rápido anterior este documento se puede considerar autocontenido puesto que el conjunto de información XML se puede generar sin ninguna información externa (un vocabulario externo).

D.5.1 Octetos (como caracteres hexadecimales) del documento infoset rápido

El cuadro D.8 presenta los octetos del documento infoset rápido para el ejemplo de orden UBL presentado en D.3.

NOTA – Se subrayan los caracteres hexadecimales que contienen los bits correspondientes a la identificación y a la terminación de elementos de información.

Cuadro D.8 – Octetos (como caracteres hexadecimales) del documento infoset rápido

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	<u>e00100000078cf027265733e75726e3a6f617369733a6e616d65733a74633a75</u>
000020	626c3a636f64656c6973743a41636b6e6f776c656467656d656e74526573706f
000040	6e7365436f64653a313a30 <u>cf026362632f75726e3a6f617369733a6e616d6573</u>
000060	3a74633a75626c3a436f6d6d6f6e4261736963436f6d706f6e656e74733a313a
000080	30cf026361633375726e3a6f617369733a6e616d65733a74633a75626c3a436f
0000a0	6d6d6f6e416767726567617465436f6d706f6e656e74733a313a30 <u>cf02637572</u>
0000c0	2f75726e3a6f617369733a6e616d65733a74633a75626c3a636f64656c697374
0000e0	3a43757272656e6379436f64653a313a30 <u>cf0278736928687474703a2f2f7777</u>
000100	772e77332e6f72672f323030312f584d4c536368656d612d696e7374616e6365
000120	<u>cd1f75726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a31</u>
000140	3a30 <u>f03d86044f726465727b85850d736368656d614c6f636174696f6e083b75</u>
000160	726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a313a3020
000180	2e2e2f2e2e2f7873642f6d61696e646f632f55424c2d4f726465722d312e302e
0001a0	787364f03d8607427579657273494482075330332d303334323537f03f828208
0001c0	4973737565446174658207323030332d30322d3033 <u>f03f838309427579657250</u>
0001e0	617274793 <u>f83830450617274793f83830850617274794e616d653f8282034e61</u>
000200	6d65820e4a65727279204275696c64657220706c63ff3f838306416464726573
000220	733 <u>f8282095374726565744e616d6582074d61727368204c616e65f03f828207</u>
000240	436974794e616d6582044e6f7768657265f03f828209506f7374616c5a6f6e65
000260	<u>82054e52313820345858f03f82820f436f756e747279537562656e7469747982</u>
000280	044e6f72666f6c6b <u>ff3f838306436f6e7461637406820645766120427269636b</u>
0002a0	<u>ffff3f83830a53656c6c6572506172747904050682135370656369616c697374</u>
0002c0	2057696e646f777320706c63ff03f82820b4275696c64696e674e616d65820b
0002e0	536e6f7768696c6c20576f726b73f009820b4c6974746c6520536e6f72696e67
000300	<u>f00a8204534d3220334e57f00b820757686572657368697265ffff3f83830744</u>
000320	656c69766572793f82821852657175657374656444656c697665727944617465
000340	54696d658210323030332d30322d32345430303a30303a3030f03f83830e4465
000360	6c69766572794164647265737308820a5269766572736964652052642ef00e82
000380	17506c6f742031372c205768697465776174657220457374617465f009820657
0003a0	68657473746f6e65f00b82064d6964646c65736578fff03f8383084f72646572
0003c0	4c696e653f8383074c696e654974656d3f8383829041f07f8282075175616e74
0003e0	697479780f7175616e74697479556e6974436f646543756e6974f09032f03f83
000400	83034974656d3f83831853656c6c6572734974656d4964656e74696669636174
000420	696f6e3f83830149449202323365756f03f838310506879736963616c417474
000440	7269627574653f83830a41747472696275746549449201776f6f64f03f82820a
000460	4465736372697074696f6e9201736f6674ff191a820366696e697368f01b8203
000480	7072696d6564ff191a820566697474696e6773f01b9202736174696eff191a82
0004a0	04676c617a696e67f01b820373696e676c65ffffff1213149042f0550180f090
0004c0	33f016171892023334305457f0191a920168616e64f01b915248ff191aa3f01b
0004e0	<u>920168617264ff191a820366696e697368f01b9202737461696eff191a820566</u>
000500	697474696e6773f01b92026272617373ff191a8204676c617a696e67f01b8203
000520	646f75626c65ffffff
00052a	

D.5.2 Explicación de la codificación

D.5.2.1 Codificación del elemento de información document y del elemento de información Order element

La explicación siguiente detalla la codificación inicial del documento infoset rápido y del elemento de información **element** raíz. En particular se explican la codificación de un elemento de información **document**, una secuencia de elementos de información **namespace**, un elemento de información **element** y un elemento de información **attribute**. El cuadro D.9 presenta el fragmento del documento infoset rápido para la codificación del elemento de información

element y para el elemento de información **Order element** de D.3.2. El cuadro D.10 detalla su codificación. El fragmento en XML 1.0 se presenta como sigue:

```
<Order xmlns:res="urn:oasis:names:tc:ubl:codelist:AcknowledgementResponseCode:1:0"
xmlns:cbc="urn:oasis:names:tc:ubl:CommonBasicComponents:1:0"
xmlns:cac="urn:oasis:names:tc:ubl:CommonAggregateComponents:1:0"
xmlns:cur="urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:oasis:names:tc:ubl:Order:1:0"
xsi:schemaLocation="urn:oasis:names:tc:ubl:Order:1:0 ../xsd/maindoc/UBL-Order-1.0.xsd">
```

Cuadro D.9 – Octetos (como caracteres hexadecimales) del fragmento

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100000078cf027265733e75726e3a6f617369733a6e616d65733a74633a75
000020	626c3a636f64656c6973743a41636b6e6f776c656467656d656e74526573706f
000040	6e7365436f64653a313a30cf026362632f75726e3a6f617369733a6e616d6573
000060	3a74633a75626c3a436f6d6d6f6e4261736963436f6d706f6e656e74733a313a
000080	30cf026361633375726e3a6f617369733a6e616d65733a74633a75626c3a436f
0000a0	6d6d6f6e416767726567617465436f6d706f6e656e74733a313a30cf02637572
0000c0	2f75726e3a6f617369733a6e616d65733a74633a75626c3a636f64656c697374
0000e0	3a43757272656e6379436f64653a313a30cf0278736928687474703a2f2f7777
000100	772e77332e6f72672f323030312f584d4c536368656d612d696e7374616e6365
000120	cd1f75726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a31
000140	3a30f03d86044f726465727b85850d736368656d614c6f636174696f6e083b75
000160	726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a313a3020
000180	2e2e2f2e2e2f7873642f6d61696e646f632f55424c2d4f726465722d312e302e
0001a0	787364f0

Cuadro D.10 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
00 01	11100000 (e0) 00000000 (00)	Los octetos están presentes al principio de cada documento infoset rápido (véase 12.6).	Elemento de información document
02 03	00000000 (00) 00000001 (01)	Los octetos son la codificación del número de versión (véase 12.9).	
04	00000000 (00)	Los octetos son la codificación de la presencia de un vocabulario inicial y de otros componentes del tipo Document . El octeto en la posición 04 ₁₆ , valor 00 ₁₆ , tiene un '0' (relleno) para el primer bit (véase 12.8). Los bits segundo a octavo son '000000' indicando que todos los componentes iniciales del tipo Document están ausentes (incluido el componente initial-vocabulary cuya ausencia se indica en el tercer bit, véase C.2.3).	
05	01111000 (78)	El octeto es la codificación inicial de un vástago del elemento de información document . El octeto en la posición 05 ₁₆ , valor 78 ₁₆ , tiene un '0' (identificación) para el primer bit indicando que existe un vástago del elemento de información document y que este es un elemento de información element (véase C.2.11.2). El segundo bit es '1' indicando que el elemento de información element tiene atributos (véase C.3.3). Los bits tercero a sexto son '1110' seguidos por '00' (relleno) en los bits séptimo y octavo, indicando que están presentes elementos de información attribute de espacio de nombres (véase C.3.4.1).	Elemento de información element con la propiedad [namespace attribute].

Cuadro D.10 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
06	11001111 (cF)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con propiedades [prefix] y [normalized value] literales.</p> <p>El octeto en la posición 06₁₆, valor cF₁₆, tiene '110011' (identificación) para los bits primero a sexto indicando que está presente un elemento de información attribute de espacio de nombres (véase C.3.4.2). El séptimo bit es '1' indicando que está presente la propiedad [prefix]. El octavo bit es '1' indicando que está presente la propiedad [normalized value].</p> <p>El octeto en la posición 07₁₆, valor 02₁₆, tiene '0' para el primer bit indicando que se está codificando una cadena de caracteres literales para la propiedad [prefix] (véase C.13.3). El segundo bit es '0' indicando que la longitud de los caracteres codificados UTF-8 es superior o igual a 1₁₀ e inferior o igual a 64₁₀ y la longitud se codifica en los bits tercero a octavo como un número entero sin signo (véase C.22.3.1). El entero sin signo es 2₁₀ y su longitud es 3₁₀ (el límite inferior es 1₁₀).</p> <p>Los 3₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [prefix]) se codifican entre el octeto en la posición 08₁₆ y el octeto en la posición 0a₁₀. Se añadirá la cadena "res" la tabla PREFIX (con un índice de 2₁₀).</p> <p>Los octetos en la posición 0b₁₆, valor 3e₁₆, tiene '0' para el primer bit indicando que se codifica una cadena de caracteres literales para la propiedad [normalized value] (véase C.13.3). El segundo bit es '0' indicando que la longitud de los caracteres codificados UTF-8 es superior o igual a 1₁₀ e inferior o igual a 64₁₀ y la longitud se codifica en los bits tercero a octavo como un número entero sin signo (véase C.22.3.1). El entero sin signo es 62₁₀ y la longitud es 63₁₀ (el límite inferior es 1₁₀).</p> <p>Los 63₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición 0c₁₆ y el octeto en la posición 4a₁₆. La cadena "...ResponseCode:1:0" se añadirá a la tabla NAMESPACE NAME (con un índice de 2₁₀).</p>	<p>xmlns:res=</p> <p>"...ResponseCode:1:0"</p>
07	00000010 (02)		
08	01110010 (72)		
....		
0a	01110010 (73)		
0b	00111110 (3e)		
0c	01110101 (75)		
....		
4a	01110000 (30)		
4b	11001111 (cF)		
4c	00000010 (02)		
4d	01100011 (63)		
....		
4f	01100011 (63)		
50	00101111 (2F)		
51	01110101 (75)		
....		
80	01110000 (30)		
81	11001111 (cF)	<p>Los octetos son la codificación del elemento de información attribute de espacio de nombres con propiedades literales [prefix] y [normalized value].</p> <p>Los 3₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [prefix]) se codifican entre el octeto en la posición 83₁₆ y el octeto en la posición 85₁₆. La cadena "cac" se añadirá a la tabla PREFIX (con un índice de 4₁₀).</p> <p>Los 52₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición 87₁₆ y el octeto en la posición ba₁₆. La cadena "...ateComponents:1:0" se añadirá a la tabla NAMESPACE NAME (con un índice de 4₁₀).</p>	<p>xmlns:cac=</p> <p>"...ateComponents:1:0"</p>
82	00000010 (02)		
83	01100011 (63)		
....		
85	01100011 (63)		
86	00110011 (33)		
87	01110101 (75)		
....		
ba	01110000 (30)		

Cuadro D.10 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
bb bc bd bf c0 c1 f0	11001111 (cƒ) 00000010 (02) 01100011 (63) 01100011 (63) 00101111 (2ƒ) 01110101 (75) 01110000 (30)	Los octetos son la codificación del elemento de información attribute de espacio de nombres con las propiedades literales [prefix] y [normalized value] . Los 3 ₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [prefix]) se codifican entre el octeto en la posición b _{d16} y el octeto en la posición b _{f16} . La cadena "cur" se añadirá a la tabla PREFIX (con un índice de 5 ₁₀). Los 48 ₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición c ₁₁₆ y el octeto en la posición f ₀₁₆ . La cadena "...CurrencyCode:1:0" se añadirá a la tabla NAMESPACE NAME (con un índice de 5 ₁₀).	xmlns:cur= "...CurrencyCode:1:0"
f1 f2 f3 f5 f6 f7 11f	11001111 (cƒ) 00000010 (02) 01111000 (78) 01101001 (69) 00101000 (28) 01101000 (68) 01110111 (77)	Los octetos son la codificación del elemento de información attribute de espacio de nombres con propiedades literales [prefix] y [normalized value] . Los 3 ₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [prefix]) se codifican entre el octeto en la posición f ₃₁₆ y el octeto en la posición f ₅₁₆ . La cadena "xsi" se añadirá a la tabla PREFIX (con un índice de 6 ₁₀). Los 41 ₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición f ₇₁₆ y el octeto en la posición 11f ₁₆ . La cadena "...Schema-instance" se añadirá a la tabla NAMESPACE NAME (con un índice de 6 ₁₀).	xmlns:xsi= "...Schema-instance"
120 121 122 141	11001101 (cd) 00011111 (1ƒ) 01110101 (75) 00110000 (30)	Los octetos son la codificación del elemento de información attribute de espacio de nombres con una propiedad [normalized value] indexada. Los 32 ₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición 122 ₁₆ y el octeto en la posición 141 ₁₆ . Se añadirá la cadena "...Order:1:0" a la tabla NAMESPACE NAME (con un índice de 7 ₁₀).	xmlns="...Order:1:0"
142	11110000 (f0)	El octeto es la codificación del terminador para la secuencia de elementos de información attribute de espacio de nombres. El octeto en la posición 142 ₁₆ , valor f ₀₁₆ , tiene '1111' (terminador) para los primeros cuatro bits (los bits primero a cuarto) y es el terminador de la secuencia. Cuatro de los seis '0' (relleno) están presentes en los bits quinto a octavo (véase C.3.4.3).	
143 144 145 146 14a	00111101 (3d) 10000110 (86) 00000100 (04) 01001111 (4ƒ) 01110010 (72)	Los octetos son la codificación de un nombre cualificado literal del elemento de información element . El octeto en la posición 143 ₁₆ , valor 3d ₁₆ , tiene los dos últimos bits de los seis '0' (relleno) en los bits cero y segundo (véase C.3.4.3). Los bits tercero a quinto son '1111' indicando que el nombre cualificado es un nombre cualificado literal (véase C.18.3). El séptimo bit es '0' indicando que el nombre cualificado no tiene una propiedad [prefix] . El bit octavo es '1' indicando que el nombre cualificado tiene una propiedad [namespace name] . El octeto en la posición 144 ₁₆ , valor 86 ₁₆ , tiene '1' para el primer bit indicando que la propiedad [namespace name] no es una cadena literal y está indexada (véase C.13.4). El segundo bit es '0' indicando que el índice es superior o igual a 1 ₁₀ e inferior o igual a 64 ₁₀ y el índice está codificado en los bits tercero a octavo como un número entero sin signo (véase C.25.2). El entero sin signo es 6 ₁₀ y el índice es 7 ₁₀ (el límite inferior es 1 ₁₀), lo que resulta en una propiedad [namespace name] de "...Order:1:0" cuando se quita la referencia en la tabla NAMESPACE NAME. El octeto en la posición 145 ₁₆ , valor 04 ₁₆ , tiene '0' para el primer bit indicando que se codifica una cadena de caracteres literales para la propiedad [local name] (véase C.13.3). El segundo bit es '0' indicando que la longitud de los caracteres codificados UTF-8 es superior o igual a 1 ₁₀ e inferior o igual a 64 ₁₀ , y la longitud se codifica en los bits tercero a octavo como un número entero sin signo (véase C.22.3.1). El entero sin signo es 4 ₁₀ y la longitud es 5 ₁₀ (el límite inferior es 1 ₁₀).	<Order ...

Cuadro D.10 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
		<p>Los 5₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [local name]) se codifican entre el octeto en la posición 14b₁₆ y el octeto en la posición 14a₁₆. Se añadirá la cadena 'Order' a la tabla LOCAL NAME (con un índice de 1₁₀).</p> <p>Se añadirá el nombre cualificado sin propiedad [prefix], con una propiedad [namespace name] "...Order:1:0" (índice 7₁₀) y una propiedad [local name] "Order" (índice 1₁₀) a la tabla ELEMENT NAME (con un índice de 1₁₀).</p>	
14b	01111011 (7b)	<p>Los octetos son la codificación de un elemento de información attribute con un nombre cualificado literal y una propiedad [normalized value]. La presencia de elementos de información attribute se indicó en el octeto en la posición 05₁₆ (el segundo bit es '1').</p> <p>El octeto en la posición 14b₁₆, valor 7b₁₆, tiene un primer bit '0' (identificación) indicando que está presente un elemento de información attribute (véase C.3.6.1). Los bits segundo a quinto son '1111' indicando que el nombre cualificado es un nombre cualificado literal (véase C.17.3). El sexto bit es '0' (relleno) (véase C.17.3). El séptimo bit es '1' indicando que el nombre cualificado tiene una propiedad [prefix]. El octavo bit es '1' indicando que el nombre cualificado tiene una propiedad [namespace name].</p> <p>El octeto en la posición 14c₁₆, valor 85₁₆, tiene '1' para el primer bit indicando que la propiedad [prefix] no es una cadena literal y está indexada (véase C.13.4). El segundo bit es '0' indicando que el índice es superior o igual a 1₁₀ e inferior o igual a 64₁₀, y el índice se codifica en los bits tercero a octavo como un número entero sin signo (véase C.25.2). El entero sin signo es 5₁₀ y el índice es 6₁₀ (el límite inferior es 1₁₀), lo que resulta en una propiedad [prefix]"xsi" cuando se quita la referencia en la tabla NAMESPACE NAME.</p> <p>El octeto en la posición 14d₁₆, valor 85₁₆, tiene '1' para el primer bit indicando que la propiedad [namespace name] no es una cadena literal y está indexada (véase C.13.4). El segundo bit es '0' indicando que el índice es superior o igual a 1₁₀ e inferior o igual a 64₁₀ y el índice se codifica en los bits tercero a octavo como un número entero sin signo (véase C.25.2). El entero sin signo es 5₁₀ y el índice es 6₁₀ (el límite inferior es 1₁₀), lo que resulta en una propiedad [namespace name] de "...Schema-instance" cuando se quita la referencia en la tabla NAMESPACE NAME.</p> <p>El octeto en la posición 14e₁₆, valor 0d₁₆, tiene '0' para el primer bit indicando que una cadena de caracteres literales está codificada para la propiedad [local name] (véase C.13.3). El segundo bit es '0' indicando que la longitud de los caracteres codificados UTF-8 es superior o igual a 1₁₀ e inferior o igual a 64₁₀ y la longitud está codificada en los bits tercero a octavo como un número entero sin signo (véase C.22.3.1). El entero sin signo es 13₁₀ y la longitud es 14₁₀ (el límite inferior es 1₁₀).</p> <p>Los 14₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [local name]) se codifican entre el octeto en la posición 14f₁₀ y el octeto en la posición 15c₁₀. Se añadirá la cadena "SchemaLocation" a la tabla LOCAL NAME (con un índice de 2₁₀).</p> <p>El nombre cualificado con una propiedad [prefix]"xsi" (con índice 6₁₀), una propiedad de [namespace name] "...Schema-instance" (índice 6₁₀), y una propiedad [local name] de "schemaLocation" (índice 2₁₀) se añadirá a la tabla ATTRIBUTE NAME (con un índice de 1₁₀).</p>	xsi:schemaLocation="...."
14c	10000101 (85)		
14d	10000101 (85)		
14e	00001101 (0d)		
14f	01110011 (73)		
....		
15c	01101110 (6e)		
15d	00001000 (08)		
15e	00111011 (3b)		
15f	01110101 (75)		
....		
1a2	01100100 (64)		

Cuadro D.10 – Detalles de codificación

Octeto(s)	Descripción	Infoset XML o XML
	<p>El octeto en la posición 15d₁₆, valor 08₁₆, es la codificación inicial de una cadena o índice que no identifica (véase C.14) para la propiedad [normalized value]. El primer bit es '0' indicando que está presente una cadena de caracteres literales (véase C.14.3). El segundo bit es '0' indicando que la cadena de caracteres literales no se añadirá a la tabla ATTRIBUTE VALUE. El tercer y cuarto bits, ambos '0', indican que el formato de codificación de la cadena es UTF-8 (véase C.19.3.1). Los bits quinto y sexto son '1' y '0' respectivamente indicando que la longitud de los octetos de los caracteres UTF-8 (la propiedad [normalized value]) es superior o igual a 9₁₀ octetos e inferior o igual a 264₁₀ octetos y que la longitud, menos el límite inferior, se codifica en ocho bits en el octeto siguiente como un número entero sin signo (véase C.22.3.2). Los bits séptimo y octavo son '0' (relleno) (véase C.22.3.2).</p> <p>El octeto en la posición 15e₁₆, valor 3b₁₆, es la codificación del número entero sin signo. La longitud de los octetos de los caracteres codificados UTF-8 es 68₁₀ (el límite inferior es 9₁₀).</p> <p>Los 68₁₀ octetos de los caracteres codificados UTF-8 (de la propiedad [normalized value]) se codifican entre el octeto en la posición 15f₁₆ y el octeto en la posición 1a2₁₆.</p>	
1a3	<p>11110000 (f0)</p> <p>El octeto es la codificación del terminador para la secuencia de elementos de información attribute.</p> <p>El octeto en la posición 1a3₁₆, valor f0₁₆, tiene '1111' para los cuatro primeros bits (del bit primero al cuarto) y es el terminador de la secuencia. Están presentes cuatro '0' (relleno) (del quinto al octavo bit) puesto que el elemento de información Order element tiene un vástago (véase D.3.2).</p>	

D.5.2.2 Codificación del elemento de información Address element del elemento de información BuyerParty element

La explicación siguiente detalla la codificación del elemento de información Address element del elemento de información BuyerParty element del documento infoset rápido. En particular se explica la codificación de los elementos de información element y de los elementos de información character. El cuadro D.11 presenta el fragmento del documento infoset rápido para la codificación del elemento de información Address Element del elemento de información BuyerParty element de D.3.2. El cuadro D.12 detalla su codificación. El fragmento en XML 1.0 se presenta como sigue:

```
<cac:Address>
  <cbc:StreetName>Marsh Lane</cbc:StreetName>
  <cbc:CityName>Nowhere</cbc:CityName>
  <cbc:PostalZone>NR18 4XX</cbc:PostalZone>
  <cbc:CountrySubentity>Norfolk</cbc:CountrySubentity>
</cac:Address>
```

Cuadro D.11 – Octetos (como caracteres hexadecimales) del fragmento

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000200	3f838306416464726573
000220	733f8282095374726565744e616d6582074d61727368204c616e65f03f828207
000240	436974794e616d6582044e6f7768657265f03f828209506f7374616c5a6f6e65
000260	82054e52313820345858f03f82820f436f756e747279537562656e7469747982
000280	044e6f72666f6c6bfff

Cuadro D.12 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
216	00111111 (3f)	Los octetos son la codificación del elemento de información	<cac:Address>
217	10000011 (83)	Address element.	
218	10000011 (83)	El octeto en la posición 216 ₁₆ , valor 3f ₁₆ , tiene un '0'	
219	00000110 (06)	(identificación) para el primer bit indicando que existe un vástago	
21a	01000001 (41)	de un elemento de información element (vástago del elemento de	
....	información Party element) y que es un elemento de información	
220	01110011 (73)	element (véase C.3.7.2). El segundo bit es '0' indicando que el	
		elemento de información element no tiene atributos (véase C.3.3).	
		Los bits tercero a quinto son '1111' indicando que el nombre	
		cualificado es un nombre cualificado literal (véase C.18.3). El	
		séptimo bit es '1' indicando que el nombre cualificado tiene una	
		propiedad [prefix] . El octavo bit es '1' indicando que el nombre	
		cualificado tiene una propiedad [namespace name] .	
		El octeto en la posición 217 ₁₆ , valor 83 ₁₆ , tiene '1' para el primer bit	
		indicando que la propiedad [prefix] no es una cadena literal y está	
		indexada (véase C.13.4). El segundo bit es '0' indicando que el	
		índice es superior o igual a 1 ₁₀ e inferior o igual a 64 ₁₀ , y el índice	
		está codificado en los bits tercero a octavo como un número entero	
		sin signo (véase C.25.2). El entero sin signo es 3 ₁₀ y el índice es 4 ₁₀	
		(el límite inferior es 1 ₁₀), lo que resulta en una propiedad [prefix]	
		de "cac" cuando se quita la referencia en la tabla PREFIX.	
		El octeto en la posición 218 ₁₆ , valor 83 ₁₆ , tiene '1' para el primer bit	
		indicando que la propiedad [namespace name] no es una cadena	
		literal y está indexada (véase C.13.4). El segundo bit es '0'	
		indicando que el índice es superior o igual a 1 ₁₀ e inferior o igual a	
		64 ₁₀ y el índice está codificado en los bits tercero a octavo como un	
		número entero sin signo (véase C.25.2). El entero sin signo es 3 ₁₀ y	
		el índice es 4 ₁₀ (el límite inferior es 1 ₁₀), lo que resulta en una	
		propiedad [namespace name] de "...ateComponents:1:0" cuando se	
		quita la referencia en la tabla NAMESPACE NAME.	
		El octeto en la posición 219 ₁₆ , valor 06 ₁₆ , tiene '0' para el primer bit	
		indicando que una cadena de caracteres literales está codificada	
		para la propiedad [local name] (véase C.13.3). El segundo bit es '0'	
		indicando que la longitud de los caracteres codificados UTF-8 es	
		superior o igual a 1 ₁₀ e inferior o igual a 64 ₁₀ y la longitud está	
		codificada en los bits tercero a octavo como un número entero sin	
		signo (véase C.22.3.1). El entero sin signo es 6 ₁₀ y la longitud	
		es 7 ₁₀ (el límite inferior es 1 ₁₀).	
		Los 7 ₁₀ octetos de los caracteres codificados UTF-8 (de la	
		propiedad [local name]) están codificados entre el octeto en la	
		posición 21a ₁₆ y el octeto en la posición 220 ₁₆ . Se añadirá la	
		cadena "Address" a la tabla LOCAL NAME (con un índice de 9 ₁₀).	
		El nombre cualificado con una propiedad [prefix] de "cac"	
		(índice 4 ₁₀), una propiedad [namespace name] de	
		".....ateComponents:1:0" (índice 4 ₁₀) y una propiedad	
		[local name] de "Order" (índice 1 ₁₀) se añadirá a la tabla	
		ELEMENT NAME (con un índice de 1 ₁₀).	
221	00111111 (3f)	Los octetos son la codificación del elemento de información	<cbc:StreetName>
222	10000010 (82)	StreetName element.	
223	10000010 (82)	La propiedad [local name] "StreetName" se añadirá a la tabla	
224	00001001 (09)	LOCAL NAME (con un índice de 10 ₁₀).	
225	01000001 (53)	El nombre cualificado con una propiedad [prefix] de "cbc"	
....	(índice 3 ₁₀), una propiedad [namespace name] de	
22e	01100101 (65)	"...BasicComponents:1:0" (índice 3 ₁₀) y una propiedad	
		[local name] de "StreetName" (índice 10 ₁₀) se añadirá a la tabla	
		ELEMENT NAME (con un índice de 9 ₁₀).	

Cuadro D.12 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
22f 230 231 23a	10000010 (82) 00000111 (07) 01001101 (4d) 01100101 (65)	<p>Los octetos son la codificación de elementos de información character del elemento de información StreetName element.</p> <p>El octeto en la posición 22f₁₆, valor 82₁₆, tiene '10' (identificación) para los dos primeros bits (los bits primero y segundo) indicando que existe un vástago de un elemento de información element (vástago del elemento de información StreetName element) y que el vástago es un fragmento de elementos de información character (véase C.3.7.5). El tercer bit es '0' indicando que está presente una cadena de caracteres literales (véase C.15.3). El cuarto bit es '0' indicando que la cadena de caracteres literales no se añadirá a la tabla CONTENT CHARACTER CHUNK. Los bits quinto y sexto, ambos '0', indican que el formato de codificación del fragmento es UTF-8 (véase C.20.3.1). Los bits séptimo y octavo son '1' y '0' respectivamente indicando que la longitud de los octetos de los caracteres codificados UTF-8 (el fragmento de elementos de información character) es superior o igual a 3₁₀ octetos e inferior o igual a 258₁₀ octetos y que la longitud, menos el límite inferior, está codificada en ocho bits del octeto siguiente como un número entero sin signo (véase C.24.3.2).</p> <p>El octeto en la posición 230₁₆, valor 07₁₆, es el número entero sin signo. La longitud de los octetos de los caracteres codificados UTF-8 es 10₁₀ (el límite inferior es 3₁₀).</p> <p>Los octetos 10₁₀ de los caracteres codificados UTF-8 están codificados entre el octeto en la posición 231₁₆ y el octeto en la posición 23a₁₆.</p>	Elemento de información character "Marsh Lane"
23b	11110000 (f0)	El octeto es el terminador para el elemento de información StreetName element .	</cbc:StreetName>
23c 23d 23e 23f 240 247	00111111 (3f) 10000010 (82) 10000010 (82) 00000111 (07) 01000011 (43) 01100101 (65)	<p>Los octetos son la codificación del elemento de información CityName element.</p> <p>La propiedad [local name] "CityName" se añadirá a la tabla LOCAL NAME (con un índice de 10₁₀).</p> <p>El nombre cualificado con una propiedad [prefix] de "cbc" (índice 3₁₀), una propiedad [namespace name] de ".....BasicComponents:1:0" (índice 3₁₀) y una propiedad [local name] de "CityName" (índice 11₁₀) se añadirá a la tabla ELEMENT NAME (con un índice de 10₁₀).</p>	<cbc:CityName>
248 249 24a 250	10000010 (82) 00000100 (04) 01001110 (4e) 01100101 (65)	<p>Los octetos son la codificación de elementos de información character del elemento de información CityName element.</p> <p>Los 7₁₀ octetos de los caracteres codificados UTF-8 se codifican entre el octeto en la posición 24a₁₆ y el octeto en la posición 250₁₆.</p>	Elemento de información character "Nowhere"
251	11110000 (f0)	El octeto es el terminador del elemento de información CityName element .	</cbc:CityName>
252 253 254 255 256 25f	00111111 (3f) 10000010 (82) 10000010 (82) 00001001 (09) 01000011 (50) 01100101 (65)	<p>Los octetos son la codificación del elemento de información PostalZone element.</p> <p>La propiedad [local name] "PostalZone" se añadirá a la tabla LOCAL NAME (con un índice de 12₁₀).</p> <p>El nombre cualificado con una propiedad [prefix] de "cbc" (índice 3₁₀), una propiedad [namespace name] de ".....BasicComponents:1:0" (índice 3₁₀), y una propiedad [local name] de "PostalZone" (índice 12₁₀) se añadirá a la tabla ELEMENT NAME (con un índice de 11₁₀).</p>	<cbc:PostalZone>
260 261 262 269	10000010 (82) 00000101 (05) 01001110 (4e) 01011000 (58)	<p>Los octetos son la codificación de elementos de información character del elemento de información PostalZone element.</p> <p>Los 8₁₀ octetos de los caracteres codificados UTF-8 se codifican entre el octeto en la posición 262₁₆ y el octeto en la posición 269₁₆.</p>	Elemento de información character "NR18 4XX"
26a	11110000 (f0)	El octeto es el terminador del elemento de información PostalZone element .	</cbc:PostalZone>

Cuadro D.12 – Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
26b	00111111 (3f)	Los octetos son la codificación del elemento de información CountrySubentity element .	<cbc:CountrySubentity>
26c	10000010 (82)		
26d	10000010 (82)	La propiedad [local name] "CountrySubentity" se añadirá a la tabla LOCAL NAME (con un índice de 13 ₁₀).	
26e	00001111 (0f)		
26f	01000011 (43)	El nombre cualificado con una propiedad [prefix] de "cbc" (índice 3 ₁₀), una propiedad [namespace name] de ".....BasicComponents:1:0" (índice 3 ₁₀) y una propiedad [local name] de "CountrySubentity" (índice 13 ₁₀) se añadirá a la tabla ELEMENT NAME (con un índice de 12 ₁₀).	
27e	01111001 (79)		
27f	10000010 (82)	Los octetos son la codificación de elementos de información character del elemento de información CountrySubentity element .	Elemento de información character "Norfolk"
280	00000100 (04)		
281	01001110 (4e)	Los 7 ₁₀ octetos de los caracteres codificados UTF-8 se codifican entre el octeto en la posición 281 ₁₆ y el octeto en la posición 287 ₁₆ .	
287	01101011 (6b)		
288	11111111 (ff)	El octeto es el terminador del elemento de información CountrySubentity element y del elemento de información Address element . El octeto en la posición 288 ₁₆ , valor ff ₁₆ , tiene '1111' (terminador) para los cuatro primeros bits (del bit primero al cuarto) y es el terminador para el elemento de información CountrySubentity element (véase C.3.8). Los últimos cuatro bits (del quinto al octavo bit) son '1111' y es el terminador del elemento de información Address element (véase C.3.8).	

D.5.2.3 Codificación del elemento de información **BuyersID element** del primer elemento de información **Linitem element**

La explicación siguiente detalla la codificación del elemento de información **BuyersID element** del primer elemento de información **Linitem element** del documento infoset rápido. En particular se explica la codificación de un elemento de información **element** cuya propiedad **[local name]** ha sido indexada antes que ese elemento de información. El cuadro D.13 presenta el fragmento del documento infoset rápido para la codificación del elemento de información **BuyersID element** del primer elemento de información **Linitem element** de D.3.2. El cuadro D.14 detalla su codificación. El fragmento en XML 1.0 se presenta como sigue:

```
<cac:Linitem>
  <cac:BuyersID>A</cac:BuyersID>
```

Cuadro D.13 – Octetos (como caracteres hexadecimales) del fragmento

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
0003c0	<u>3f8383074c696e654974656d3f8383829041f0</u>

Cuadro D.14– Detalles de codificación

	Octeto(s)	Descripción	Infoset XML o XML
3c4 3c5 3c6 3c7 3c8 3cf	00111111 (3f) 10000011 (83) 10000011 (83) 00001111 (07) 01001010 (4c) 01101101 (6d)	Los octetos son la codificación del elemento de información LinItem element . La propiedad [local name] "LinItem" se añadirá a la tabla LOCAL NAME (con un índice de 21 ₁₀). El nombre cualificado con una propiedad [prefix] de "cac" (índice 4 ₁₀), una propiedad [namespace name] de ".....ateComponents:1:0" (índice 4 ₁₀) y una propiedad [local name] de "LinItem" (índice 21 ₁₀) se añadirá a la tabla ELEMENT NAME (con un índice de 20 ₁₀).	<cac:LinItem>
3d0 3d1 3d2 3d3	00111111 (3f) 10000011 (83) 10000011 (83) 10000010 (82)	El octeto es la codificación del elemento de información BuyersID element . Las propiedades [prefix] , [namespace name] y [local name] han sido indexadas puesto que las cadenas asociadas se han generado antes que este elemento de información. La propiedad [local name] se indexó procesando el primer vástago del elemento de información Order element , es decir, el elemento de información BuyersID element con la propiedad [namespace name] "...Order:1:0". El nombre cualificado con una propiedad [prefix] de "cac" (índice 4 ₁₀), una propiedad [namespace name] de ".....ateComponents:1:0" (índice 4 ₁₀) y una propiedad [local name] de "BuyersID" (índice 3 ₁₀) se añadirá a la tabla ELEMENT NAME (con un índice de 21 ₁₀).	<cac:BuyersID>
3d4 3d5	10010000 (90) 01000001 (41)	Los octetos son la codificación de los elementos de información character del elemento de información BuyersID element . El octeto en la posición 3d4 ₁₆ , valor 90 ₁₆ , tiene '10' (identificación) para los dos primeros bits (los bits primero y segundo) indicando que existe un vástago del elemento de información element (vástago del elemento de información BuyersID element) y que es un conjunto de elementos de información character (véase C.3.7.5). El tercer bit es '0' indicando que está presente una cadena de caracteres literales (véase C.15.3). El cuarto bit es '1' indicando que la cadena de caracteres literales se añadirá a la tabla CONTENT CHARACTER CHUNK (en este ejemplo se añaden cadenas con menos de 6 ₁₀ caracteres a la tabla CONTENT CHARACTER CHUNK o a la tabla ATTRIBUTE VALUE). Los bits quinto y sexto, ambos '0', indican que el formato de codificación del fragmento es UTF-8 (véase C.20.3.1). El séptimo bit es '0' indicando que la longitud de los octetos de los caracteres codificados UTF-8 (el fragmento de los elementos de información character) es superior o igual a 1 ₁₀ octetos e inferior o igual a 2 ₁₀ octetos y que la longitud, menos el límite inferior, se codifica en los ocho bits como un número entero sin signo (véase C.24.3.1). El entero sin signo es 0 ₁₀ y la longitud es 1 ₁₀ . El octeto del carácter codificado UTF-8 se codifica mediante el octeto en la posición 41 ₁₆ .	Elemento de información "A" character .
3d6	11110000 (f0)	El octeto es el terminador del elemento de información BuyersID element .	</cac:BuyersID>

Anexo E

Asignación de valores de identificador de objetos

(Este anexo no es parte integrante de la presente Recomendación | Norma Internacional)

En esta Recomendación | Norma Internacional se asignan los identificadores de objetos y los descriptores de objetos siguiente.

```
{ joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoaset(0) modules(0)
fast-infoaset(0) }
```

"Módulo ASN.1 de conjunto de información rápido"

```
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoaset(0) modules(0)
fast-infoaset-edm(1) }
```

"Módulo de definición de codificación de conjunto de información rápido"

```
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoaset(0) modules(0)
fast-infoaset-elm(2) }
```

"Módulo de enlace de codificación de conjunto de información rápido"

```
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoaset(0) encodings(1)
optional-xml-declaration(0) } -- Definido como finf-doc-opt-decl en A.1
```

"Un documento infoaset rápido que contiene una declaración XML"

```
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoaset(0) encodings(1)
no-xml-declaration(1) } -- Definido como finf-doc-no-decl en A.1
```

"Documento infoaset rápido sin una declaración XML"

BIBLIOGRAFÍA

- [1] OASIS *Universal Business Language (UBL) 1.0*.
- [2] IETF RFC 1952 (1996), *GZIP file format specification version 4.3*.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedia
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedia
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos, comunicaciones de sistemas abiertos y seguridad
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet y Redes de la próxima generación
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación