UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T X.691

(04/95)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS –

NOTACIÓN DE SINTAXIS ABSTRACTA UNO

TECNOLOGÍA DE LA INFORMACIÓN – REGLAS DE CODIFICACIÓN DE NOTACIÓN DE SINTAXIS ABSTRACTA UNO – ESPECIFICACIÓN DE LAS REGLAS DE CODIFICACIÓN COMPACTADA

Recomendación UIT-T X.691
Reemplazada por una versión más reciente

(Anteriormente «Recomendación del CCITT»)

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. En el UIT-T, que es la entidad que establece normas mundiales (Recomendaciones) sobre las telecomunicaciones, participan unos 179 países miembros, 84 empresas de explotación de telecomunicaciones, 145 organizaciones científicas e industriales y 38 organizaciones internacionales.

Las Recomendaciones las aprueban los Miembros del UIT-T de acuerdo con el procedimiento establecido en la Resolución N.° 1 de la CMNT (Helsinki, 1993). Adicionalmente, la Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, aprueba las Recomendaciones que para ello se le sometan y establece el programa de estudios para el periodo siguiente.

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. El texto de la Recomendación UIT-T X.691 se aprobó el 10 de abril de 1995. Su texto se publica también, en forma idéntica, como Norma Internacional ISO/CEI 8825-2.

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.

Reemplazada por una versión más reciente

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

(Febrero de 1994)

ORGANIZACIÓN DE LAS RECOMENDACIONES DE LA SERIE X

Dominio	Recomendaciones
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 enunciados 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 móviles de transmisión de datos	X.350-X.369
Gestión	X.370-X.399
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.649
Denominación, direccionamiento y registro	X.650-X.679
Notación de sintaxis abstracta uno	X.680-X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.700-X.799
SEGURIDAD	X.800-X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Cometimiento, concurrencia y recuperación	X.850-X.859
Tratamiento de transacciones	X.860-X.879
Operaciones a distancia	X.880-X.899
TRATAMIENTO ABIERTO DISTRIBUIDO	X.900-X.999

Reemplazada por una versión más reciente

ÍNDICE

Reemplazada por una versión más reciente

Resumen

En esta Recomendación | Norma Internacional se describe un conjunto de reglas de codificación que pueden aplicarse a valores de todos los tipos ASN.1 para lograr una representación más compacta que la proporcionada por las reglas de codificación básica y sus derivadas (descritas en la Recomendación X.690).

Introducción

Las publicaciones Rec. UIT-T X.680 | ISO/CEI 8824-1, Rec. UIT-T X.681 | ISO/CEI 8824-2, Rec. UIT-T X.682 | ISO/CEI 8824-3, Rec. UIT-T X.683 | ISO/CEI 8824-4, Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1, Rec. UIT-T X.681/Enm. 1 | ISO/CEI 8824-2/Enm. 1, juntas, describen la notación de sintaxis abstracta uno (ASN.1), una notación para la definición de los mensajes que se han de intercambiar entre aplicaciones pares.

La presente Recomendación | Norma Internacional define reglas de codificación aplicables a valores de tipos definidos utilizando la notación especificada en la Rec. UIT-T X.680 | ISO/CEI 8824-1. La aplicación de estas reglas de codificación produce una sintaxis de transferencia para estos valores. En la especificación de estas reglas de codificación está implícito que las mismas se han de utilizar también para la decodificación.

Hay más de un conjunto de reglas de codificación que se pueden aplicar a valores de tipos ASN.1. La presente Recomendación | Norma Internacional define un conjunto de reglas de codificación compactada, denominadas así porque logran una representación más compacta que la obtenida con las reglas de codificación básica y sus derivadas descritas en la Rec. UIT-T X.690 | ISO/CEI 8825-1, a la cual se hace referencia para algunas partes de la especificación de estas reglas de codificación compactada.

NORMA INTERNACIONAL

RECOMENDACIÓN UIT-T

TECNOLOGÍA DE LA INFORMACIÓN – REGLAS DE CODIFICACIÓN DE NOTACIÓN DE SINTAXIS ABSTRACTA UNO – ESPECIFICACIÓN DE LAS REGLAS DE CODIFICACIÓN COMPACTADA

1 Alcance

La presente Recomendación | Norma Internacional especifica un conjunto de reglas de codificación compactada (PER, *packed encoding rules*), que se pueden usar para derivar una sintaxis de transferencia para valores de tipos definidos en la Rec. UIT-T X.680 | ISO/CEI 8824-1. Estas reglas de codificación compactada son aplicables también para decodificar esa sintaxis abstracta, con el fin de identificar los valores de datos transferidos.

Las reglas de codificación especificadas en la presente Recomendación | Norma Internacional:

- se utilizan en el momento de la comunicación;
- están diseñadas para usarlas cuando el interés principal en la elección de reglas de codificación es minimizar el tamaño de la representación de valores;
- permiten la extensión de una sintaxis abstracta por adición de valores suplementarios, preservando las codificaciones de los valores existentes, para todas las formas de extensión descritas en la Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1.

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 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 indicadas 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 mantiene una lista de las Recomendaciones UIT-T actualmente vigentes.

2.1 Recomendaciones | Normas Internacionales idénticas

- Recomendación UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: El modelo básico.*
- Recomendación UIT-T X.216 (1994) | ISO/CEI 8822:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Definición del servicio de presentación.*
- Recomendación UIT-T X.226 (1994) | ISO/CEI 8823-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Protocolo de presentación con conexión: Especificación del protocolo.*
- Recomendación UIT-T X.680 (1994) | ISO/CEI 8824-1:1995, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica.*
- Recomendación UIT-T X.680 (1994)/Enm. 1 (1995) | ISO/CEI 8824-1:1995/Amd. 1:1995, *Tecnología de la información – Notación de sintaxis abstracta uno – Especificación de la notación básica – Enmienda 1: Reglas de extensibilidad.*
- Recomendación UIT-T X.681 (1994) | ISO/CEI 8824-2:1995, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de objetos de información.*
- Recomendación UIT-T X.681 (1994)/Enm. 1 (1995) | ISO/CEI 8824-2:1995/Amd. 1:1995, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación del objeto de información – Enmienda 1: Reglas de extensibilidad.*

- Recomendación UIT-T X.682 (1994) | ISO/CEI 8824-3:1995, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones.*
- Recomendación UIT-T X.683 (1994) | ISO/CEI 8824-4:1995, *Tecnología de la información – Notación de sintaxis abstracta uno: Parametrización de las especificaciones de notación de sintaxis abstracta uno.*
- Recomendación UIT-T X.690 (1994) | ISO/CEI 8825-1:1995, *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.*

2.2 Referencias adicionales

- Recomendación X.208 del CCITT (1988), *Especificación de la notación de sintaxis abstracta uno (NSA.1).*
- ISO/CEI 2022:1994, *Information technology – Character code structure and extension techniques*.
- ISO 2375:1985, *Data processing – Procedure for registration of escape sequences*.
- ISO 6093:1985, *Information processing – Representation of numerical values in character strings for information interchange*.
- ISO/CEI 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*.
- ISO *International Register of Coded Character Sets to be Used with Escape Sequences*.
- ISO/CEI 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*.

3 Definiciones

A los efectos de la presente Recomendación | Norma Internacional se aplican las siguientes definiciones.

3.1 Definición del servicio de presentación básico

En la presente Recomendación | Norma Internacional se aplican las siguientes definiciones de la Rec. UIT-T X.216 | ISO/CEI 8822:

a) conjunto de contextos definido;

b) identificador de contexto de presentación.

3.2 Especificación de la notación básica

A los efectos de la presente Recomendación | Norma Internacional, se aplican todas las definiciones que figuran en la Rec. UIT-T X.680 | ISO/CEI 8824-1.

3.3 Extensibilidad de la ASN.1

Se aplican las siguientes definiciones de la Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1:

a) marcador de extensión;

b) series de extensión;

c) raíz de extensión;

d) adición de extensión.

3.4 Especificación de objetos de información

A los efectos de la presente Recomendación | Norma Internacional se aplican todas las definiciones que figuran en la Rec. UIT-T X.681 | ISO/CEI 8824-2.

3.5 Especificación de constricciones

La presente Recomendación | Norma Internacional utiliza los siguientes términos definidos en la Rec. UIT-T X.682 | ISO/CEI 8824-3:

a) constricción de relación de componentes;

b) constricción de tabla.

3.6 Parametrización de la especificación de ASN.1

La presente Recomendación | Norma Internacional utiliza el siguiente término definido en la Rec. UIT-T X.683 | ISO/CEI 8824-4:

– constricción variable.

3.7 Reglas de codificación básica

La presente Recomendación | Norma Internacional utiliza los siguientes términos definidos en la Rec. UIT-T X.690 | ISO/CEI 8825-1:

a) conformidad dinámica;

b) conformidad estática;

c) valor de datos;

d) codificación (de un valor de datos);

e) emisor;

f) receptor.

3.8 Definiciones adicionales

Además, se aplican las siguientes definiciones:

3.8.1 codificación de entero binario con complemento de dos: Codificación de un número entero en un campo de bits alineados por octeto de una longitud especificada, o en el número mínimo de octetos que acomodarán ese número entero codificado como un entero formado por complemento de dos, que proporciona representaciones para números enteros que son iguales, mayores o menores que cero, como se especifica en 10.4.

NOTAS

1 El valor de un número binario complemento de dos se obtiene numerando los bits en los octetos de contenido, comenzando la numeración por el bit 1 del último octeto como bit cero y terminándola por el bit 8 del primer octeto. A cada bit se asigna un valor numérico de 2^N, donde N es la posición en la mencionada secuencia de numeración. El valor del número binario complemento de dos se obtiene sumando los valores numéricos asignados a cada bit que esté puesto a 1, excluido el bit 8 del primer octeto, después de lo cual se resta de este valor el valor numérico asignado al bit 8 del primer octeto si dicho bit está puesto a 1.

2 *Número entero* es un sinónimo del término matemático *entero*. Se utiliza para evitar la confusión con el tipo *entero* de la notación de sintaxis abstracta uno.

3.8.2 valor de sintaxis abstracta: Valor de sintaxis abstracta (definido como el conjunto de valores de un tipo notación de sintaxis abstracta uno simple), que se codificará por las reglas de codificación compactada, o que se generará mediante una decodificación por las reglas de codificación compactada.

NOTA – El tipo notación de sintaxis abstracta uno simple asociado con una sintaxis abstracta se identifica formalmente por un objeto de clase ABSTRACT-SYNTAX.

3.8.3 campo de bits: Producto de alguna parte del mecanismo de codificación que consiste en un conjunto ordenado de bits que no son necesariamente un múltiplo de ocho, y que no comienzan necesariamente por un bit situado en el límite de un octeto en la codificación completa del valor de sintaxis abstracta.

3.8.4 codificación canónica: Codificación completa de un valor de sintaxis abstracta obtenida por la aplicación de reglas de codificación que no tienen opciones dependientes de la implementación; estas reglas resultan en la definición de una correspondencia de uno a uno entre cadenas de bits inequívocas y únicas en la sintaxis de transferencia y valores en la sintaxis abstracta.

3.8.5 tipo compuesto: Tipo conjunto, secuencia, conjunto de, secuencia de, elección, valor de datos de presentación insertado, externo o cadena de caracteres no constreñida.

3.8.6 valor compuesto: Valor de un tipo compuesto.

3.8.7 número entero constreñido: Número entero que está constreñido por constricciones visibles a las reglas de codificación compactada a permanecer dentro de una gama "lb" a "ub" con el valor "lb" menor o igual que "ub", y los valores de "lb" y "ub" como valores permitidos.

NOTA – Los números enteros constreñidos aparecen en la codificación que identifica la alternativa elegida de un tipo elección, la longitud de carácter, los tipos cadena de octeto y de bits cuya longitud ha sido restringida por constricciones visibles a las reglas de codificación compactada a una longitud máxima, la cuenta del número de componentes en un tipo secuencia de o conjunto de, que ha sido restringido por constricciones visibles a las reglas de codificación compactada a un número máximo de componentes, el valor de un tipo entero que ha sido constreñido por constricciones visibles a las reglas de codificación compactada a permanecer dentro de valores mínimos y máximos finitos, y el valor que indica una enumeración en un tipo enumerado.

3.8.8 constricción de tamaño efectiva (para un tipo de cadena constreñida): Una sola constricción de tamaño finito que se podría aplicar a un tipo de cadena incorporada y cuyo efecto sería permitir solamente todas aquellas longitudes que puedan estar presentes en el tipo de cadena constreñida.

NOTA – Por ejemplo, lo siguiente tiene una constricción de tamaño efectiva:

A ::= IA5String (SIZE(1..4) | SIZE(10..15))

porque se puede escribir igualmente con una sola constricción de tamaño que se aplica a todos los valores:

A ::= IA5String (SIZE(1..4 | 10..15))

mientras que lo siguiente no tiene constricción de tamaño efectiva porque la cadena puede ser arbitrariamente larga si no contiene otros caracteres que 'a', 'b' y 'c':

B ::= IA5String (SIZE(1..4) | FROM("abc"))

3.8.9 constricción de alfabeto permitido efectiva (para un tipo de cadena de caracteres restringida constreñida): Constricción de alfabeto permitido única que se podría aplicar a un tipo cadena de caracteres de multiplicador conocido incorporado y cuyo efecto sería permitir solamente todos aquellos caracteres que pueden estar presentes en cualquier posición de carácter de cualquiera de los valores en el tipo de cadena de caracteres restringida constreñida.

NOTA – Una constricción de alfabeto permitido efectiva es o bien todo el alfabeto del tipo de cadena de caracteres no constreñida, o bien una especificación de alfabeto permitido que sea un superconjunto de todas las constricciones de alfabeto permitido impuestas al tipo. Por ejemplo, en:

Ax ::= IA5String (FROM("AB") | FROM("CD"))

Bx ::= IA5String (SIZE(1..4) | FROM("abc"))

"Ax" tiene una constricción de alfabeto permitido efectiva que consiste en todo el alfabeto IA5String porque no hay constricción de alfabeto permitido que se aplique a todos los valores de "Ax". Lo mismo es válido para "Bx". Por otra parte, lo siguiente tiene una constricción de alfabeto permitido efectiva de "ABCDE" porque hay una constricción de alfabeto permitido especificada que se aplica a todos los valores:

A ::= IA5String (FROM("AB") | FROM("CD") | FROM("ABCDE"))

3.8.10: índice de enumeración: El número entero no negativo asociado con un ítem de enumeración en un tipo enumerado. Los índices de enumeración se determinan clasificando las enumeraciones en orden ascendente por su valor de enumeración, después de lo cual se les asigna un índice de enumeración que es cero para la primera enumeración, uno para la segunda enumeración, y así sucesivamente hasta la última en la lista clasificada.

NOTA – En las enumeraciones de raíz, los ítems de enumeración se clasifican separadamente de los ítems de enumeración adicional.

3.8.11 extensible para codificación de reglas de codificación compactada: Propiedad de un tipo cuya definición contiene un marcador de extensión que afecta a la codificación de reglas de codificación compactada.

3.8.12 lista de campos: Conjunto ordenado de valores de campos de bits y/o de campos de bits alineados en octeto que se produce como resultado de la aplicación de estas reglas de codificación a componentes de un valor.

NOTA – (Didáctica) El modelo empleado en la presente Recomendación | Norma Internacional utiliza el término "lista de campos" para indicar una lista enlazada (o lista referenciada) de tampones, cada uno de los cuales contiene una codificación, una longitud en bits, y un indicador de alineación en octeto de "campo de bits" o "campo de bits alineados en octeto". Cada codificación es la de un valor de un tipo de notación de sintaxis abstracta uno. El indicador de alineación de octeto determina si la codificación se ha de alinear sobre un límite de octeto cuando se utiliza para formar la codificación completa del valor de sintaxis abstracta, o si se debe añadir inmediatamente después del último bit de la codificación anterior en la codificación completa. La noción de "lista de campos" tiene fines descriptivos solamente y no indica un método de implementación.

3.8.13 longitud indefinida: Codificación cuya longitud es mayor que 64K-1 o cuya longitud máxima no puede ser determinada a partir de la notación de notación de sintaxis abstracta uno.

3.8.14 tipo de longitud fija: Un tipo tal que el valor del determinante de longitud más externo en una codificación de este tipo puede ser determinado (utilizando los mecanismos especificados en la presente Recomendación | Norma Internacional) a partir de la notación de tipo (después de la aplicación de constricciones visibles a las reglas de codificación compactada solamente) y es el mismo para todos los posibles valores del tipo.

3.8.15 valor fijo: Valor que puede ser determinado (utilizando los mecanismos especificados en la presente Recomendación | Norma Internacional) y que es el único valor permitido (después de la aplicación de constricciones visibles a las reglas de codificación compactada solamente) del tipo que lo gobierna.

3.8.16 tipo de cadena de caracteres de multiplicador conocido: Un tipo de cadena de caracteres restringida donde el número de octetos en la codificación es un múltiplo fijo conocido del número de caracteres en la cadena de caracteres para todos los valores de las cadenas de caracteres permitidas. Los tipos de cadenas de caracteres de multiplicador conocidos son: cadena IA5, cadena imprimible, cadena visible, cadena numérica, cadena universal y cadena BMP.

3.8.17 determinante de longitud: Cuenta (de bits, octetos, caracteres o componentes) que determina la longitud de una parte o de toda una codificación de reglas de codificación compactada.

3.8.18 número entero no negativo normalmente pequeño: Parte de una codificación que representa valores de un entero no negativo no acotado, pero donde es más probable que se produzcan valores pequeños que valores grandes.

3.8.19 longitud normalmente pequeña: Codificación de longitud que representa valores de una longitud no limitada, pero en la cual las longitudes pequeñas tienen mayor probabilidad de aparición que las grandes.

3.8.20 campo de bits alineados en octeto: Producto de alguna parte del mecanismo de codificación que consiste en un conjunto ordenado de bits que no son necesariamente un múltiplo de ocho, pero que tienen que comenzar en el límite de octeto en la codificación completa del valor de sintaxis abstracta.

3.8.21 codificación de entero binario no negativo: Codificación de un número entero constreñido o semiconstreñido en un campo de bits de una longitud especificada, o en un campo de bits alineados en octeto de una longitud especificada, o en el número mínimo de octetos que acomodarán ese número entero codificado como un entero binario no negativo que proporciona representaciones para números enteros mayores o iguales que cero, como se especifica en 10.3

NOTA – El valor de un número binario complemento de dos se obtiene numerando los bits en los octetos de contenido, comenzando con el bit 1 del último octeto como bit cero y terminando la numeración con el bit 8 del primer octeto. A cada bit se asigna un valor numérico de 2^N, donde N es la posición en la mencionada secuencia de numeración. El valor del número binario complemento de dos se obtiene sumando los valores numéricos asignados a cada bit que esté puesto a 1.

3.8.22 constricción visible a las reglas de codificación compactada: Caso de utilización de la notación de constricciones de notación de sintaxis abstracta uno que afecta a la codificación de reglas de codificación compactada de un valor.

3.8.23 codificación de retransmisión segura: Codificación completa de un valor en sintaxis abstracta que puede ser decodificado (incluidas cualesquiera codificaciones insertadas) sin conocer el conjunto de contextos definidos por la capa de presentación que formó el entorno en el cual se realizó la codificación.

3.8.24 número entero semiconstreñido: Número entero que está constreñido por constricciones visibles a las reglas de codificación compactada a rebasar o ser igual a algún valor "lb" con el valor "lb" como un valor permitido, y que no es un número entero constreñido.

NOTA – Los números completos semiconstreñidos se producen en la codificación de la longitud de tipos de cadenas de caracteres de octetos y de bits no constreñidas (y en algunos casos, constreñidas), el cómputo del número de componentes en tipos secuencia de y conjunto de no constreñidos (y en algunos casos constreñidos), y el valor de un tipo entero que ha sido constreñido para rebasar algún valor mínimo.

3.8.25 tipo simple: Tipo que no es un tipo compuesto.

3.8.26 textualmente dependiente: Término utilizado para identificar el caso en que, si se utiliza algún nombre de referencia para evaluar un conjunto de elementos, el valor del conjunto de elementos es considerado dependiente de ese nombre de referencia, sin tener en cuenta si la aritmética de conjunto real que se está aplicando en realidad es tal que el valor final del conjunto de elementos es independiente del valor del conjunto de elementos real asignado en realidad al nombre de referencia.

NOTA – Por ejemplo, la siguiente definición de "Foo" es textualmente dependiente de "Bar" aunque "Bar" no tiene efecto sobre el conjunto de valores de "Foo" (de acuerdo con 9.3.4, la constricción sobre "Foo" no es visible a las reglas de codificación compactada porque "Bar" está constreñido por una constricción de tabla y "Foo" es textualmente dependiente de "Bar").

```
MY-CLASS ::= CLASS { &name PrintableString, &age INTEGER } WITH SYNTAX{&name , &age}

MyObjectSet MY-CLASS ::= { {"Jack", 7} | {"Jill", 5} }

Bar ::= MY-CLASS.&age ({MyObjectSet})

Foo ::= INTEGER (Bar | 1..100)
```

3.8.27 número entero no constreñido: Número entero que no está constreñido por constricciones visibles a las reglas de codificación compactada.

NOTA – Los números enteros no constreñidos se producen solamente en la codificación de un valor del tipo entero.

4 Abreviaturas

ASN.1 Notación de sintaxis abstracta uno (*abstract syntax notation one*)

BER Reglas de codificación básica de ASN.1 (*basic encoding rules of ASN.1*)

PER Reglas de codificación compactada de ASN.1 (*packed encoding rules of ASN.1*)

CER Reglas de codificación canónica de ASN.1 (*canonical encoding rules of ASN.1*)

16K 16384

32K 32768

48K 49152

64K 65536

5 Notación

La presente Recomendación | Norma Internacional hace referencia a la notación definida en la Rec. UIT-T X.680 | ISO/CEI 8824-1.

6 Convenios

6.1 La presente Recomendación | Norma Internacional define el valor de cada octeto en una codificación mediante la utilización de los términos "bit más significativo" y "bit menos significativo".

NOTA – En las especificaciones de capa inferior se usa la misma notación para definir el orden de transmisión en una línea serie, o la asignación de bits a canales paralelos.

6.2 A los efectos de la presente Recomendación | Norma Internacional, los bits de un octeto se numeran de 8 a 1, siendo el bit 8 el "bit más significativo" y el bit 1 el "bit menos significativo".

6.3 El término "octeto" se utiliza frecuentemente en la presente Recomendación | Norma Internacional para indicar "ocho bits". La utilización de este término en lugar de "ocho bits" no implica alineación. Cuando se desea hacer referencia a la alineación, ésta se indica explícitamente en la presente Recomendación | Norma Internacional.

7 Reglas de codificación definidas en la presente Recomendación | Norma Internacional

7.1 La presente Recomendación | Norma Internacional especifica cuatro reglas de codificación (junto con sus identificadores de objeto asociados) que se pueden utilizar para codificar y decodificar los valores de una sintaxis abstracta definida como los valores de un solo tipo ASN.1 (conocido). Esta cláusula describe su aplicabilidad y propiedades.

7.2 Sin conocer el tipo del valor codificado, no es posible determinar la estructura de la codificación (en cualquiera de los algoritmos de las reglas de codificación compactada (PER)). En particular, el fin de la codificación no se puede determinar a partir de la propia codificación sin conocer el tipo codificado.

7.3 Las codificaciones PER son siempre seguras para la retransmisión a condición de que los valores abstractos de los tipos EXTERNAL (externo), EMBEDDED PDV (PDV insertado) y CHARACTER STRING (cadena de caracteres) estén constreñidos para evitar el transporte de identificadores de contextos de presentación.

7.4 El algoritmo de regla de codificación más general especificado en la presente Recomendación | Norma Internacional es BASIC-PER, que en general no produce una codificación canónica.

7.5 Un segundo algoritmo de regla de codificación especificado en la presente Recomendación | Norma Internacional es CANONICAL-PER, que produce codificaciones que son canónicas. Se define como una restricción de elecciones que dependen de la implementación en la codificación BASIC-PER. La codificación CANONICAL-PER produce codificaciones canónicas que tienen aplicaciones cuando hay que aplicar autenticadores a valores abstractos, como se describe en la Rec. UIT-T X.690 | ISO/CEI 8825-1, Anexo D.

NOTA – Cualquier implementación conforme a CANONICAL-PER para la codificación se conforma a BASIC-PER para la codificación. Cualquier implementación conforme a BASIC-PER para la decodificación es conforme a CANONICAL-PER para la decodificación. De este modo, las codificaciones hechas de acuerdo con CANONICAL-PER son codificaciones permitidas por BASIC-PER.

7.6 Si un tipo codificado con BASIC-PER o CANONICAL-PER contiene tipos EMBEDDED PDV, CHARACTER STRING o EXTERNAL, la codificación más externa deja de ser segura para la retransmisión a menos que la sintaxis de transferencia utilizada para todos los tipos EMBEDDED PDV, CHARACTER STRING y EXTERNAL sea segura para la retransmisión. Si un tipo codificado con BASIC-PER o CANONICAL-PER contiene tipos EMBEDDED PDV, EXTERNAL o CHARACTER STRING, la codificación más externa deja de ser canónica a menos que la sintaxis de transferencia utilizada para todos los tipos EMBEDDED PDV, EXTERNAL y CHARACTER STRING sea canónica.

NOTA – Las sintaxis de transferencia de caracteres que soportan todas las sintaxis abstractas de caracteres de la forma {iso standard 10646 level-1 (1) ....} son canónicas. Las que soportan {iso standard 10646 level-2 (2) ....} e {iso standard 10646 level-3 (3) ....} no siempre son canónicas. Todas las sintaxis de transferencia de caracteres mencionadas anteriormente son seguras para la retransmisión.

7.7 BASIC-PER y CANONICAL-PER vienen en dos variantes, la variante ALIGNED (alineada) y la variante UNALIGNED (no alineada). En la variante ALIGNED se insertan bits de relleno de tiempo en tiempo para restablecer la alineación en octetos. En la variante UNALIGNED, no se insertan bits de relleno.

7.8 No hay posibilidades de interfuncionamiento entre la variante ALIGNED y la variante UNALIGNED.

7.9 Las codificaciones PER son autolimitadoras solamente si se conoce el tipo del valor codificado. Las codificaciones son siempre un múltiplo de ocho bits. Cuando se transportan en un tipo EXTERNAL serán transportadas en la alternativa de elección OCTET STRING (cadena de octetos), a menos que el propio tipo EXTERNAL esté codificado en PER, en cuyo caso el valor puede ser codificado como un solo tipo ASN.1 (es decir, un tipo abierto). Cuando se transportan en el protocolo de presentación de OSI, se utilizará la "codificación completa" (definida en la Rec. UIT-T X.226 | ISO/CEI 8823-1) con la alternativa de elección OCTECT STRING.

7.10 Las reglas de la presente Recomendación | Norma Internacional se aplican a ambos algoritmos y a ambas variantes a menos que se indique otra cosa.

7.11 El Anexo C es informativo, y contiene recomendaciones sobre cuáles combinaciones de PER se han de implementar para maximizar las posibilidades de interfuncionamiento.

8 Conformidad

8.1 La conformidad dinámica se especifica en la cláusula 9.

8.2 La conformidad estática es especificada por las normas que especifican la aplicación de estas reglas de codificación compactada.

NOTA – El Anexo C a la presente Recomendación | Norma Internacional proporciona orientación sobre la conformidad estática en relación con el soporte de las dos variantes de los dos algoritmos de reglas de codificación. Esta orientación está destinada a asegurar el interfuncionamiento, a la vez que reconoce los beneficios, para algunas aplicaciones, de codificaciones que no son seguras para la retransmisión, ni canónicas.

8.3 Las reglas de la presente Recomendación | Norma Internacional se especifican desde el punto de vista de un procedimiento de codificación. Las implementaciones no tienen que reflejar el procedimiento especificado, a condición de que la cadena de bits producida como la codificación completa de un valor de sintaxis abstracta sea idéntica a una de las especificadas en la presente Recomendación | Norma Internacional para la sintaxis de transferencia aplicable.

8.4 Las implementaciones que efectúan la decodificación tienen que producir el valor abstracto correspondiente a cualquier cadena de bits recibida que pueda ser producida por un emisor conforme a las reglas de codificación identificadas en la sintaxis de transferencia asociada con el material que se decodifica.

NOTAS

1 En general no hay codificaciones alternativas definidas para la BASIC-PER explícitamente indicada en la presente Recomendación | Norma Internacional. La BASIC-PER se convierte en canónica especificando la operación de retransmisión segura y restringiendo algunas de las opciones de codificación de otras Normas ISO/CEI referenciadas. La CANONICAL-PER proporciona una alternativa para las reglas de codificación distinguida y las reglas de codificación canónica (véase la Rec. UIT-T X.690 | ISO/CEI 8825-1) cuando se requiere una codificación canónica y con retransmisión segura.

2 Cuando se utiliza CANONICAL-PER para proporcionar una codificación canónica, se recomienda que cualquier valor encriptado resultante derivado de ésta tenga asociado un identificador de algoritmo que identifique CANONICAL-PER como la transformación del valor abstracto a una cadena de bits inicial (que es seguidamente troceada ("hashed")).

9 Método de codificación utilizado para PER

9.1 Utilización de la notación de tipos

9.1.1 Estas reglas de codificación utilizan específicamente la notación de tipos ASN.1 indicada en la Rec. UIT-T X.680 | ISO/CEI 8824-1 y sólo se pueden aplicar para codificar los valores de un tipo ASN.1 especificado utilizando esa notación.

9.1.2 En particular, aunque no exclusivamente, son dependientes de la siguiente información retenida en el modelo de tipos y valores ASN.1 que sustenta la utilización de la notación:

a) el anidamiento de tipos elección dentro de tipos elección;

b) los rótulos colocados en los componentes en un tipo conjunto, y en las alternativas en un tipo elección, y los valores dados a una enumeración;

c) si un componente de tipo conjunto o secuencia es facultativo o no;

d) si un componente de tipo conjunto o secuencia tiene un valor DEFAULT o no;

e) la gama restringida de valores de un tipo que se produce mediante la aplicación de constricciones visibles a PER (solamente);

f) si un componente es o no un tipo abierto;

g) si está o no presente un marcador de extensión.

9.2 Utilización de rótulos para proporcionar un orden canónico

La presente Recomendación | Norma Internacional requiere que los componentes de un tipo conjunto y de un tipo elección estén ordenados canónicamente con independencia de la ordenación textual de los componentes. El orden canónico es determinado ordenando los rótulos de los componentes, como se especifica en 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

9.3 Constricciones visibles a PER

NOTA – El hecho de que algunas constricciones de la ASN.1 pueden no ser visibles a PER a los efectos de la codificación y decodificación no afecta en modo alguno a la utilización de estas constricciones en el tratamiento de errores detectados durante la codificación, ni supone que los valores que violan estas constricciones pueden ser transmitidos por un emisor conforme.

9.3.1 Las constricciones que se expresan en texto legible por el hombre o en un comentario de la ASN.1 no son visibles a PER.

9.3.2 Las constricciones de variables no son visibles a PER (véanse 10.4 y 10.5 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

9.3.3 Las constricciones de tablas no son visibles a PER (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3).

9.3.4 Las constricciones cuya evaluación es textualmente dependiente de una constricción de tabla o de una constricción de relación de componentes no son visibles a PER (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3).

9.3.5 Las constricciones de relación de componentes no son visibles a PER (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3).

9.3.6 Las constricciones de tipos de cadenas de caracteres restringidas que no son (véase la cláusula 34 de la Rec. UIT-T X.680 | ISO/CEI 8824-1) tipos de cadenas de caracteres de multiplicador conocido no son visibles a PER (véase 3.8.16).

9.3.7 De acuerdo con lo anterior, todas las constricciones de tamaño son visibles a PER.

9.3.8 La constricción de tamaño efectiva para un tipo constreñido es una constricción de tamaño de modo que se permite solamente si hay algún valor del tipo constreñido que tiene ese tamaño (permitido). Si el tipo constreñido tiene valores de un tamaño que no satisfacen la constricción, no hay constricción de tamaño efectiva.

9.3.9 Las constricciones de alfabeto permitido en tipos de cadenas de caracteres de multiplicador conocido son visibles a PER.

9.3.10 La constricción de alfabeto permitido efectiva para un tipo constreñido es una constricción de alfabeto permitido única de modo que se permite un carácter solamente si hay algún valor del tipo constreñido que contiene ese carácter. Si todos los caracteres del tipo constreñido pueden estar presentes en algún valor del tipo constreñido, entonces la constricción de alfabeto permitido efectiva es el conjunto de caracteres definido para el tipo no constreñido.

NOTAS

1 En la definición de un tipo constreñido, se pueden aplicar múltiples constricciones visibles a PER directamente o a través de la utilización de "ContainedSubtype"s (subtipos contenidos).

2 En el Anexo B se hacen observaciones sobre el efecto de la combinación de constricciones que individualmente son visibles a PER.

9.3.11 Una constricción de tipo interno aplicada a un tipo real es visible a PER.

9.3.12 Una constricción de tipo interno aplicada a un tipo de cadena de caracteres no restringida o a un tipo de pdv insertado es visible a PER solamente cuando se utiliza para restringir el valor del componente de las "sintaxis" a un solo valor, o cuando se utiliza, para restringir "identificación" a la alternativa "fija" (véanse las cláusulas 24 y 27).

9.3.13 Las constricciones sobre los tipos útiles no son visibles a PER.

9.3.14 A reserva de lo anterior, todas las otras constricciones son visibles a PER solamente si se aplican a un tipo entero o, excluidas las constricciones de valores individuales, a un tipo cadena de caracteres de multiplicador conocido.

9.3.15 Si una constricción visible a PER distinta de alfabeto permitido tiene un marcador de extensión, el tipo se define para que sea extensible para codificaciones PER.

NOTAS

1 Si un marcador de extensión está presente en una ConstraintSpec (especificación de constricción) que no es visible a PER y no hay otro marcador de extensión presente en la constricción, el tipo es codificado por PER como si no tuviese marcador de extinción.

2 Si hay múltiples especificaciones de SizeConstraint (constricción de tamaño) aplicadas a un tipo y una de ellas es extensible, el tipo se codifica en PER como si el marcador de extensión estuviese presente en todas las especificaciones de SizeConstraint.

9.3.16 Un tipo es también extensible para codificaciones PER si se ha producido de algunas de las formas siguientes:

a) se deriva de un tipo ENUMERATED (enumerado) (por subtipificación, referencia de tipo o rotulación) y hay un marcador de extensión en la producción "Enumerations" (enumeraciones); o

b) se deriva de un tipo SEQUENCE (secuencia) (por subtipificación, referencia de tipo o rotulación) y hay un marcador de extensión en las producciones "ComponentTypeLists" (listas de tipos de componentes) o "SequenceType" (tipo de secuencia); o

c) se deriva de un tipo SET (conjunto) (por subtipificación, referencia de tipo o rotulación) y hay un marcador de extensión en las producciones "ComponentTypeLists" o "SetType" (tipo de conjunto); o

d) se deriva de un tipo CHOICE (elección) (por subtipificación, referencia de tipo, o rotulación) y hay un marcador de extensión en la producción "AlternativeTypeLists" (listas de tipos alternativos).

9.4 Modelo de tipos y valores utilizados para la codificación

9.4.1 Un tipo ASN.1 es un tipo simple o un tipo construido con otros tipos. La notación permite utilizar referencias de tipos y rotulación de tipos. A los efectos de estas reglas de codificación, la utilización de referencias y rotulación de tipos no producen efecto sobre la codificación y son invisibles en el modelo, salvo lo indicado en 9.2. La notación permite también la aplicación de constricciones y especificaciones de error. Las constricciones visibles a PER están presentes en el modelo como una restricción de los valores de un tipo. Otras constricciones y especificaciones de error no afectan a la codificación y son invisibles en el modelo.

9.4.2 Un valor que se ha de codificar se puede considerar como un valor simple o como un valor compuesto construido utilizando los mecanismos de estructuración, a partir de componentes que son valores simples o compuestos, procediendo en forma paralela a la estructura de la definición de tipos ASN.1.

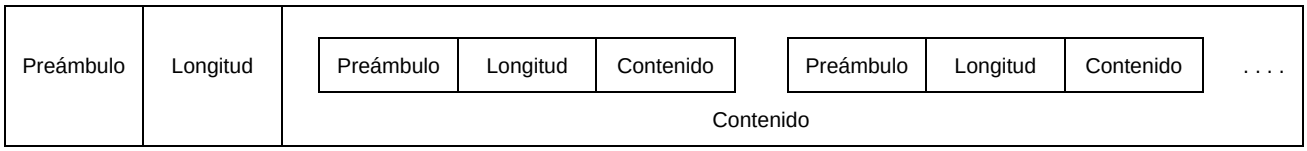
9.5 Estructura de una codificación

9.5.1 Estas reglas de codificación especifican:

a) la codificación de un valor simple en una lista de campos;

b) la codificación de un valor compuesto en una lista de campos, utilizando las listas de campos generadas por la aplicación de estas reglas de codificación a los componentes del valor compuesto; y

c) la transformación de la lista de campos del valor más externo en la codificación completa del valor de sintaxis abstracta (véase 10.1).



NOTA – Preámbulo, longitud y contenido son "campos" que, concatenados unos tras otros, forman una "lista de campos". La lista de campos de un tipo compuesto, salvo el tipo elección, puede estar constituida por los campos de varios valores concatenados unos tras otros. Pueden faltar, o bien el preámbulo, o la longitud y/o el contenido de cualquier valor.

Figura 1 – Codificación de un valor compuesto en una lista de campos

9.5.2 La codificación de un componente de un valor de datos, o bien:

a) consiste en tres partes, indicadas en la Figura 1, que aparecen en el siguiente orden:

1) un preámbulo (véanse las cláusulas 18, 20 y 22);

2) un determinante de longitud (véase 10.9);

3) contenido; o

b) (cuando el contenido es extenso) consiste en un número arbitrario de partes (véase la Figura 2) de las cuales la primera es un preámbulo (véanse las cláusulas 18, 20 y 22) y las partes siguientes son pares de campos de bits alineados en octetos, siendo el primero un determinante de longitud para un fragmento del contenido y el segundo dicho fragmento del contenido; el último par de campos es identificado por la parte de determinante de longitud, como se especifica en 10.9.

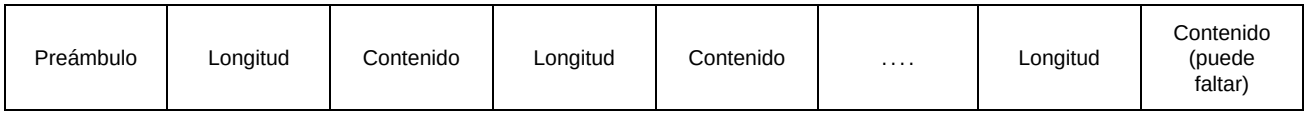


Figura 2 – Codificación de un valor de datos largo

9.5.3 Cada una de las partes mencionadas en 9.5.2 genera, o bien:

a) un campo nulo (nada); o

b) un campo de bits; o

c) un campo de bits alineados en octeto; o

d) una lista de campos que puede contener campos de bits, campos de bits alineados en octeto, o ambas modalidades.

9.6 Tipos que se han de codificar

9.6.1 Las siguientes cláusulas especifican la codificación de los siguientes tipos en una lista de campos: tipos booleano (boolean), entero (integer), enumerado (enumerated), real (real), cadena de bits (bitstring), cadena de octetos (octetstring), nulo (null), secuencia (sequence), secuencia de (sequence-of), conjunto (set), conjunto de (set-of), elección (choice), abierto (open), identificador de objeto (object identifier), pdv insertado (embedded-pdv), externo (external), cadena de caracteres restringida (restricted character string) y cadena de caracteres sin restricciones (unrestricted character string).

9.6.2 El tipo ANY (cualquiera), definido en la Rec. X.208 del CCITT (1988) | ISO/CEI 8824:1990, se codificará como un tipo abierto.

9.6.3 El tipo de selección se codificará como una codificación del tipo seleccionado.

9.6.4 La codificación de tipos rotulados no se incluye en esta Recomendación | Norma Internacional porque, salvo lo indicado en 9.2, la rotulación no es visible en el modelo de tipos y valores utilizado para estas reglas de codificación. Por tanto, los tipos rotulados se codifican de acuerdo con la codificación del tipo que ha sido rotulado.

9.6.5 Los siguientes "tipos útiles" de la cláusula 38 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 se codificarán como si hubiesen sido sustituidos por sus definiciones dadas en la Rec. UIT-T X.680 | ISO/CEI 8824-1:

– tiempo generalizado;

– tiempo universal;

– descriptor de objeto.

Las constricciones en estos tipos útiles no son visibles a PER.

10 Procedimientos de codificación

10.1 Producción de la codificación completa

10.1.1 La lista de campos producida como un resultado de la aplicación de esta Recomendación | Norma Internacional al valor más externo se utilizará para producir la codificación completa del valor de sintaxis abstracta como sigue: cada campo en la lista de campos se tomará en turno y se concatenará al final de la cadena de bits que forma la codificación completa del valor de sintaxis abstracta junto con bits cero adicionales para relleno como se especifica más adelante.

10.1.2 En la variante UNALIGNED de estas reglas de codificación, todos los campos se concatenarán sin relleno. Si el resultado de la codificación del valor más externo es una cadena de bits vacía, la cadena de bits se sustituirá por un solo octeto con todos los bits puestos a cero. Si es una cadena de bits no vacía y no es un múltiplo de ocho bits (cero a siete) se añadirán bits cero para producir un múltiplo de ocho bits.

10.1.3 En la variante ALIGNED de estas reglas de codificación, cualesquiera campos de bits en la lista de campos se concatenarán sin relleno y cualesquiera campos de bits alineados en octeto se concatenarán después que se hayan concatenado los bits cero (cero a siete) para hacer que la longitud de la codificación producida hasta ese momento sea un múltiplo de ocho bits. Si el resultado de la codificación del valor más externo es una cadena de bits vacía, la cadena de bits será sustituida por un sólo octeto con todos los bits puestos a cero. Si es una cadena de bits no vacía y no es un múltiplo de ocho bits, se le añadirán bits cero (cero a siete) para producir un múltiplo de ocho bits.

NOTA – La codificación del valor más externo es la cadena de bits vacía si, por ejemplo, el valor de sintaxis abstracta es del tipo nulo o de un tipo entero constreñido a un solo valor.

10.1.4 La cadena de bits resultante es la codificación completa del valor de sintaxis abstracta.

10.2 Campos de tipo abierto

10.2.1 Para codificar un campo de tipo abierto, el valor del tipo que ocupa en efecto el campo se codificará en una lista de campos que será convertida después a una codificación completa de un valor de sintaxis abstracta, como se especifica en 10.1 para producir una cadena de octetos de longitud "n" (digamos).

10.2.2 A la lista de campos para el valor en el cual se ha de insertar el tipo abierto se añadirá (como se especifica en 10.9) una longitud no constreñida de "n" (en unidades de octetos) y un campo de bits alineados en octeto que contiene los bits producidos en 10.2.1.

NOTA – Cuando el número de octetos en la codificación de tipo abierto es grande, se utilizarán los procedimientos de fragmentación de 10.9, y la codificación del tipo abierto se interrumpirá sin considerar la posición del límite del fragmento en la codificación del tipo que ocupa el campo de tipo abierto.

10.3 Codificación como un entero binario no negativo

NOTA – (Didáctica) Esta subcláusula da precisión al término "codificación de entero binario no negativo", colocando el entero en un campo formado por un número fijo de bits, un campo formado por un número fijo de octetos, o un campo formado por el número mínimo de octetos necesarios para acomodarlo.

10.3.1 Las siguientes cláusulas se refieren a la generación de una codificación entera binaria no negativa de un número entero no negativo en, o bien un campo de bits de longitud especificada, o en un solo octeto, o en un octeto doble, o en el número mínimo de octetos para el valor. Esta cláusula especifica la codificación precisa que se ha de aplicar cuando se hacen estas referencias.

10.3.2 El bit anterior del campo se define como el bit más significativo del primer octeto, y el bit posterior del campo se define como el bit menos significativo del último octeto.

10.3.3 Solamente para la siguiente definición, los bits se numerarán cero para el bit posterior del campo, uno para el bit siguiente, y así sucesivamente hasta el bit anterior del campo.

10.3.4 En una codificación entera binaria no negativa, el valor del número entero representado por la codificación será la suma de los valores especificados por cada bit. Un bit que se pone a "0" tiene valor cero. Un bit con el número "n" que se pone a "1" tiene el valor 2^n.

10.3.5 La codificación que da por suma (según se define anteriormente) el valor que se está codificando es una codificación de ese valor.

NOTA – Cuando el tamaño del campo codificado es fijo (un campo de bits de longitud especificada, un octeto simple, o un octeto doble), hay una codificación única que da por suma el valor que se codifica.

10.3.6 Una codificación entera binaria no negativa por octetos mínimos del número entero (que no predetermina el número de octetos que se ha de utilizar para la codificación) tiene un campo que es un múltiplo de ocho bits y satisface también la condición de que los ocho bits anteriores del campo no serán todos cero a menos que el campo tenga precisamente una longitud de ocho bits.

NOTA – Esta es una condición necesaria y suficiente para producir una codificación única.

10.4 Codificación como un entero binario complemento de dos

NOTA – (Didáctica) Esta subcláusula da precisión al término "codificación entera binaria complemento de dos", poniendo un entero con signo en un campo formado por el número mínimo de octetos para acomodarlo. Estos procedimientos son referenciados en ulteriores especificaciones de codificación.

10.4.1 Las siguientes cláusulas se refieren a la generación de una codificación entera binaria complemento de dos de un número entero (que puede ser negativo, cero, o positivo) en el número mínimo de octetos para el valor. Esta subcláusula especifica la codificación precisa que se ha de aplicar cuando se hacen estas referencias.

10.4.2 El bit anterior del campo se define como el bit más significativo del primer octeto y el bit posterior del campo se define como el bit menos significativo del último octeto.

10.4.3 Para la siguiente definición solamente, los bits se numerarán cero para el bit posterior del campo, uno para el siguiente y así sucesivamente hasta el bit anterior del campo.

10.4.4 En una codificación entera binaria complemento de dos, el valor del número entero representado por la codificación será la suma de los valores especificados por cada bit. Un bit que se pone a "0" tiene valor cero. Un bit con número "n" que se pone a "1" tiene el valor 2^n a menos que sea el bit anterior, en cuyo caso tiene el valor (negativo) $–2^n$.

10.4.5 Toda codificación que da por suma (como se define anteriormente) el valor que se codifica es una codificación de ese valor.

10.4.6 Una codificación entera binaria complemento de dos de octetos mínimos del número entero tiene una anchura de campo que es múltiplo de ocho bits y satisface también la condición de que los nueve bits anteriores del campo no serán todos cero y no serán todos unos.

NOTA – Esta es una condición necesaria y suficiente para producir una codificación única.

10.5 Codificación de un número entero constreñido

NOTA – (Didáctica) Esta subcláusula es referenciada por otras cláusulas y hace referencia a cláusulas anteriores para la producción de una codificación de entero binario no negativo o de entero binario complemento de dos. Para la variante UNALIGNED, el valor se codifica siempre en el número mínimo de bits necesario para representar la gama (definida en 10.5.3). En el resto de esta nota se trata de la variante ALIGNED. Cuando la gama es menor o igual que 255, el valor se codifica en un campo de bits del tamaño mínimo para la gama. Cuando la gama es exactamente 256, el valor se codifica en un campo de bits alineados en octetos de un solo octeto. Cuando la gama es 257 a 64K, el valor se codifica en un campo de bits alineados en octeto de dos octetos. Cuando la gama es mayor que 64K, se hace caso omiso de la gama y el valor se codifica en un campo de bits alineados en octeto formado por el número mínimo de octetos para el valor. En este último caso, los procedimientos posteriores (véase 10.9) codifican también un campo de longitud (usualmente un solo octeto) para indicar la longitud de la codificación. Para los otros casos, la longitud de la codificación es independiente del valor que se codifica, y no se codifica explícitamente.

10.5.1 Esta subcláusula especifica una correspondencia de un número entero constreñido a un campo de bits o un campo de bits alineados en octeto y es invocada en cláusulas posteriores de la presente Recomendación | Norma Internacional.

10.5.2 Los procedimientos de esta subcláusula se invocan solamente si hay un número entero constreñido que se ha de codificar y los valores del límite inferior, "lb" y del límite superior "ub" han sido determinados a partir de la notación del tipo (después de la aplicación de constricciones visibles a PER).

NOTA – No se puede determinar un límite inferior si MIN se evalúa a un número infinito, ni un límite superior si MAX se evalúa a un número infinito. Por ejemplo, no se puede determinar un límite superior o inferior a INTEGER(MIN..MAX).

10.5.3 Defínase "gama" como el valor entero ("ub" – "lb" + 1), y sea "n" el valor que se ha de codificar.

10.5.4 Si "gama" tiene el valor 1, el resultado de la codificación será un campo de bits vacío (sin bits).

10.5.5 Se han de considerar otros cinco casos (que conducen a codificaciones diferentes), de las cuales una se aplica a la variante UNALIGNED y cuatro a la variante ALIGNED.

10.5.6 En el caso de la variante UNALIGNED, el valor ("n" – "lb") se codificará como un entero binario no negativo en un campo de bits como el especificado en 10.3 con un número mínimo de bits necesarios para representar la gama.

NOTA – Si "gama" satisface la desigualdad 2^m < "gama" <= 2^{m+1}, el número de bits es m + 1.

10.5.7 En el caso de la variante ALIGNED, la codificación depende de si:

a) "gama" es menor o igual que 255 (el caso de campo de bit);

b) "gama" es exactamente 256 (el caso de un octeto);

c) "gama" es mayor que 256 y menor o igual que 64K (el caso de dos octetos);

d) "gama" es mayor que 64K (el caso de longitud indefinida).

10.5.7.1 (El caso de campo de bit.) Si "gama" es menor o igual que 255, la invocación de esta cláusula requiere la generación de un campo de bits con un número de bits especificado en la tabla siguiente, y que contiene el valor ("n" – "lb") como una codificación de entero binario no negativo en un campo de bits como se especifica en 10.3.

"Gama"	Tamaño de campo de bits (en bits)
2	1
3, 4	2
5, 6, 7, 8	3
9 a 16	4
17 a 32	5
33 a 64	6
65 a 128	7
129 a 255	8

10.5.7.2 (El caso de un octeto.) Si la gama tiene un valor de 256, el valor ("n" – "lb") se codificará en un campo de bits alineados en octeto de un octeto como un entero binario no negativo, como se especifica en 10.3.

10.5.7.3 (El caso de dos octetos.) Si la "gama" tiene un valor mayor o igual que 257 y menor o igual que 64K, el valor ("n" – "lb") se codificará en un campo de bits alineados en octeto de dos octetos como una codificación de entero binario no negativo, como se especifica en 10.3.

10.5.7.4 (El caso de longitud indefinida.) En los demás casos, el valor ("n" – "lb") se codificará como un entero binario no negativo en un campo de bits alineados en octetos con el número mínimo de octetos especificado en 10.3 y el número de octetos "len" utilizado en la codificación es empleado por otras cláusulas que hacen referencia a esta subcláusula para especificar una codificación de la longitud.

10.6 Codificación de un número entero no negativo normalmente pequeño

NOTA – (Didáctica) Este procedimiento se utiliza cuando se codifica un número entero no negativo que se espera sea pequeño, pero cuyo tamaño es potencialmente ilimitado debido a la presencia de un marcador de extensión. Un ejemplo es un índice de elección.

10.6.1 Si el número entero no negativo, "n", es menor o igual que 63, se añadirá un campo de bits de un bit a la lista de campos con el bit puesto a 0 y "n" se codificará como un entero binario no negativo en un campo de bits de seis bits.

10.6.2 Si "n" es mayor o igual que 64, se añadirá un campo de bits de un solo bit con el bit puesto a 1 a la lista de campos. El valor "n" se codificará como un número entero semiconstreñido con "lb" igual a 0 y se invocarán los procedimientos de 10.9 para añadirlo a la lista de campos precedido por un determinante de longitud.

10.7 Codificación de un número entero semiconstreñido

NOTA – (Didáctica) Este procedimiento se utiliza cuando se puede identificar un límite inferior pero no un límite superior. El procedimiento de codificación coloca las distancias con respecto al límite inferior en el número mínimo de octetos como un entero binario no negativo y requiere una codificación de longitud explícita (típicamente un solo octeto) como se especifica en procedimientos ulteriores.

10.7.1 Esta subcláusula especifica una correspondencia de un número entero semiconstreñido con un campo de bits alineados en octetos, y se invoca en cláusulas ulteriores de esta Recomendación | Norma Internacional.

10.7.2 Los procedimientos de esta cláusula se invocan solamente si se dispone de un número entero semiconstreñido (digamos "n") que se ha de codificar, y el valor de "lb" ha sido determinado a partir de la notación de tipo (después de la aplicación de constricciones visibles a PER).

NOTA – No se puede determinar un límite inferior si MIN se evalúa a un número infinito. Por ejemplo, no se puede determinar un límite inferior a INTEGER(MIN..MAX).

10.7.3 Los procedimientos de esta subcláusula siempre producen el caso de longitud indefinida.

10.7.4 (El caso de longitud indefinida.) El valor ("n" – "lb") se codificará como un entero binario no negativo en un campo de bits alineados en octetos con el número mínimo de octetos especificados en 10.3 y el número de octetos "len" utilizado en la codificación se emplea en otras cláusulas que hacen referencia a esta subcláusula para especificar una codificación de la longitud.

10.8 Codificación de un número entero no constreñido

NOTA – (Didáctica) Este caso sólo se plantea en la codificación del valor de un tipo entero sin límite inferior. El procedimiento codifica el valor como un entero binario complemento de dos en el número mínimo de octetos requerido para acomodar la codificación y requiere una codificación de longitud explícita (típicamente un solo octeto) como se especifica en procedimientos ulteriores.

10.8.1 Esta subcláusula especifica una correspondencia de un número entero sin constricciones (digamos "n") con un campo de bits alineados en octetos y se invoca en cláusulas ulteriores de esta Recomendación | Norma Internacional.

10.8.2 Los procedimientos de esta subcláusula siempre producen el caso de longitud indefinida.

10.8.3 (El caso de longitud indefinida.) El valor "n" se codificará como un entero binario complemento de dos en un campo de bits alineados en octeto con el número mínimo de octetos especificados en 10.4 y el número de octetos "len" utilizados en la codificación se emplea en otras cláusulas que hacen referencia a esta subcláusula para especificar una codificación de la longitud.

10.9 Reglas generales para codificar un determinante de longitud

NOTAS

1 (Didáctica) Los procedimientos de esta subcláusula se invocan cuando se necesita un campo de longitud explícita para alguna parte de la codificación con independencia de si la cuenta de la longitud está limitada o no por encima (por constricciones visibles a PER). La parte de la codificación a la cual se aplica la longitud puede ser una cadena de bits (con la cuenta de longitud en bits), una cadena de octetos (con la cuenta de longitud en octetos), una cadena de caracteres de multiplicador conocido (con la cuenta de longitud en caracteres) o una lista de campos (con la cuenta de longitud en componentes de una secuencia de o de un conjunto de).

2 (Didáctica) En el caso de la variante ALIGNED, si la cuenta de longitud está acotada por encima por un límite superior que es menor que 64K, la codificación de número entero constreñido se utiliza para la longitud. Para gamas suficientemente pequeñas, el resultado es un campo de bits. En los demás casos, la longitud no constreñida (digamos "n") se codifica en un campo de bits alineados en octetos en una de las tres maneras siguientes (en orden ascendente del tamaño):

a) ("n" menor que 128) un solo octeto que contiene "n" con el bit 8 puesto a cero;

b) ("n" menor que 16K) dos octetos que contienen "n" con el bit 8 del primer octeto puesto a 1 y el bit 7 puesto a cero;

c) ("n" grande) un solo octeto que contiene una cuenta "m" con el bit 8 puesto a 1 y el bit 7 puesto a 1. La cuenta "m" es uno a cuatro y la longitud indica que sigue un fragmento del material (un múltiplo "m" de 16K ítems). Para todos los valores de "m", el fragmento va seguido por otra codificación de longitud para el resto del material.

3 (Didáctica) En la variante UNALIGNED, si la cuenta de longitud está acotada por encima por un límite superior que es menor que 64K, se utiliza la codificación de número entero constreñido para codificar la longitud en el número mínimo de bits necesarios para representar la gama. En los demás casos, la longitud no constreñida (digamos "n") se codifica en un campo de bits de la manera descrita anteriormente en la Nota 2.

10.9.1 Esta subcláusula no se invoca si, de acuerdo con la especificación de cláusulas posteriores, el valor del determinante de longitud "n" es fijado por la definición del tipo (constreñido por constricciones visibles a PER) a un valor menor que 64K.

10.9.2 Esta subcláusula se invoca para añadir a la lista de campos un campo, o lista de campos, precedidos por un determinante de longitud "n" que determina, o bien:

a) la longitud en octetos de un campo asociado (las unidades son octetos); o

b) la longitud en bits de un campo asociado (las unidades son bits); o

c) el número de codificaciones de componentes en una lista de campos asociada (las unidades son componentes de un conjunto de o de secuencia de); o

d) el número de caracteres en el valor de un tipo de cadena de caracteres de multiplicador conocido asociada (las unidades son caracteres).

10.9.3 (Variante ALIGNED) Los procedimientos para la variante ALIGNED se especifican en 10.9.3.1-10.9.3.8.4. (Los procedimientos para la variante UNALIGNED se especifican en 10.9.4.)

10.9.3.1 Como resultado del análisis de la definición de tipo (que se especifica en cláusulas posteriores), el determinante de longitud (un número completo "n") habrá dado, o bien:

a) una longitud normalmente pequeña con el límite inferior "lb" igual a uno; o

b) un número entero constreñido con un límite inferior "lb" (mayor o igual que cero) y un límite superior "ub" menor que 64K; o

c) un número entero semiconstreñido con un límite inferior "lb" (mayor o igual que cero), o un número entero constreñido con un límite inferior "lb" (mayor o igual que cero) y un límite superior "ub" mayor o igual que 64K.

10.9.3.2 Las subcláusulas que invocan los procedimientos de esta subcláusula habrán determinado un valor para "lb", el límite inferior de la longitud (ésta es cero si la longitud no tiene constricciones), y para "ub", el límite superior de la longitud. "ub" no se fija si no hay límite superior determinable a partir de las constricciones visibles a PER.

10.9.3.3 Cuando el determinante de longitud es un número entero constreñido con "ub" menor que 64K, se añadirá a la lista de campos la codificación del número entero constreñido para el determinante de longitud como se especifica en 10.5. Si "n" no es cero, estará seguido del campo o lista de campos asociados, y se finalizan estos procedimientos. Si "n" es cero, no habrá otra adición a la lista de campos, y se finalizan estos procedimientos.

NOTAS

1 Por ejemplo:

A ::= Foo (SIZE (3..6)) -- *La longitud se codifica en un campo de bits de dos bits*

B ::= Foo (SIZE (40000..40254)) -- *La longitud se codifica en un campo de bits de ocho bits*

C ::= Foo (SIZE (0..32000)) -- *La longitud se codifica en un campo de bits alineado en octeto de dos octetos*

D ::= Foo (SIZE (64000)) -- *La longitud no se codifica*

2 El efecto de no hacer adiciones cuando "n" es igual a cero es que el relleno hasta llegar al límite de un octeto no se produce cuando se invocan estos procedimientos para añadir un campo de bits alineado en octeto de longitud cero, a menos que así se requiera por lo indicado en 10.5.

10.9.3.4 Cuando el determinante de longitud es una longitud normalmente pequeña y "n" es menor que o igual a 64, se añadirá un campo de bits de un solo bit a la lista de campos con el bit puesto a cero, y el valor "n – 1" se codificará como un entero binario no negativo en un campo de bits de seis bits. A esto seguirá el campo asociado, y se finalizan estos procedimientos. Si "n" es mayor que 64, se añadirá un campo de bits de un solo bit a la lista de campos con el bit puesto a 1, seguido de la codificación de "n" como un determinante de longitud no constreñido, seguido por el campo asociado, de acuerdo con 10.9.3.5-10.9.3.8.4.

NOTA – Las longitudes normalmente pequeñas sólo se utilizan para indicar la longitud de la tabla de bits ("bitmap") que prefija los valores de adición de extensión de un tipo secuencia o conjunto.

10.9.3.5 En otro caso (ausencia de constricciones, o "ub" grande), "n" se codifica y se añade a la lista de campos, seguido de los campos asociados como se especifica más adelante.

NOTA – La cota inferior, "lb", no afecta a las codificaciones de longitud especificadas en 10.9.3.6-10.9.3.8.4.

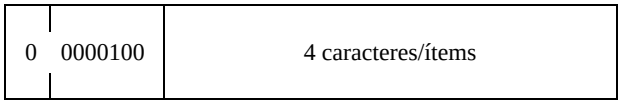
10.9.3.6 Si "n" es menor que o igual a 127, se codificará como un entero binario no negativo (mediante los procedimientos de 10.3) en los bits 7 (más significativo) a 1 (menos significativo) de un solo octeto y el bit 8 se pondrá a cero. Esto se añadirá a la lista de campos como un campo de bits alineado en octeto, seguido del campo o la lista de campos asociados, y se finalizan estos procedimientos.

NOTA – Por ejemplo, si en lo siguiente un valor de "A" tiene una longitud de cuatro caracteres, y un valor de "B" tiene una longitud de cuatro ítems,

A ::= IA5String

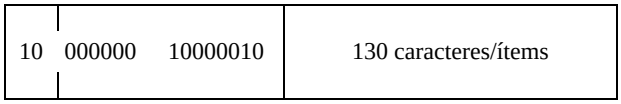
B ::= Foo (SIZE (4..123456))

ambos valores se codifican de modo que el octeto de longitud ocupe un octeto, y con el primer bit puesto a cero para indicar que la longitud es menor o igual que 127:



10.9.3.7 Si "n" es mayor que 127 y menor que 16K, se codificará como un entero binario no negativo (mediante los procedimientos de 10.3) en los bits del bit 6 del octeto uno (más significativo) al bit 1 del octeto dos (menos significativo) de un campo de bits alineado en octeto de dos octetos, con el bit 8 del primer octeto puesto a 1 y el bit 7 del primer octeto puesto a cero. Esto se añadirá a la lista de campos seguido por el campo o la lista de campos asociados, y se finalizan estos procedimientos.

NOTA – Si en el ejemplo de 10.9.3.6 un valor de "A" tiene una longitud de 130 caracteres y el de "B" tiene una longitud de 130 ítems, ambos valores se codifican de modo que el componente de longitud ocupe dos octetos, y con los dos primeros bits puestos a 10 para indicar que la longitud es mayor que 127 y menor que 16K.



10.9.3.8 Si "n" es mayor que o igual a 16K, se añadirá a la lista de campos un solo octeto en un campo de bits alineado en octeto con el bit 8 puesto a 1 y el bit 7 puesto a 1, y los bits 6 a 1 codificarán el valor 1, 2, 3 ó 4 como un entero binario no negativo (mediante los procedimientos de 10.8). Este octeto único irá seguido por una parte del campo o de la lista de campos asociados, como se especifica más adelante.

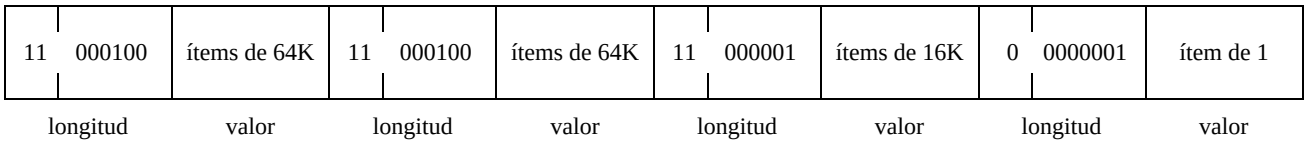
NOTA – El valor de los bits 6 a 1 está limitado a 1-4 (en lugar de estar comprendido entre los límites teóricos de 0-63) con el fin de limitar el número de ítems que una implementación está obligada a conocer a un número más manejable (64K en lugar de 1024K).

10.9.3.8.1 El valor (1 a 4) de los bits 6 a 1 se multiplicará por 16K con lo que se obtiene una cuenta (digamos "m"). El valor entero que se elegirá como contenido de los bits 6 a 1 será el valor máximo admitido de modo que el campo o la lista de campos asociados contengan más de, o exactamente, "m" octetos, bits, componentes o caracteres, según proceda.

NOTAS

1 La forma no fragmentada trata longitudes de hasta 16K. La fragmentación proporciona por tanto longitudes de hasta 64K con una granularidad de 16K.

2 Si en el ejemplo de 10.9.3.6 un valor de "B" tiene una longitud de 144K + 1 (es decir, 64K + 64K + 16K + 1) ítems, el valor se fragmenta, con los dos primeros bits de los tres primeros fragmentos puestos a 11 para indicar que siguen de uno a cuatro bloques, cada uno con ítems de 16K, y que otro componente de longitud seguirá al último bloque de cada segmento:



10.9.3.8.2 La parte del contenido especificada por "m" se añadirá a la lista de campos como:

a) un campo de bits alineados en octeto de "m" octetos que contienen los primeros "m" octetos del campo asociado, para unidades que son octetos; o

b) un campo de bits alineados en octetos de "m" bits que contienen los primeros "m" bits del campo asociado, para unidades que son bits; o

c) la lista de campos que codifican los primeros "m" componentes en la lista de campos asociada, para unidades que son componentes de tipos conjunto de o secuencia de; o

d) un campo de bits alineados en octeto de "m" caracteres que contienen los primeros "m" caracteres del campo asociado, para unidades que son caracteres.

10.9.3.8.3 Los procedimientos de 10.9 se aplicarán de nuevo para añadir la parte restante del campo asociado o la lista de campos a la lista de campos con una longitud que es un número entero semiconstreñido igual a ("n" – "m") con un límite inferior de cero.

NOTA – Si el último fragmento que contiene parte del valor codificado tiene una longitud que es un múltiplo exacto de 16K, es seguido por un fragmento final que consiste solamente en un componente de longitud de un octeto puesto a cero.

10.9.3.8.4 La adición de sólo una parte del campo o campos asociados a la lista de campos con la aplicación repetida de estos procedimientos se denomina **procedimiento de fragmentación**.

10.9.4 (Variante UNALIGNED) Los procedimientos para la variante UNALIGNED se especifican en 10.9.4.1 a 10.9.4.2 (los procedimientos para la variante ALIGNED se especifican en 10.9.3):

10.9.4.1 Si el determinante de longitud "n" que se ha de codificar es un número entero constreñido con "gama" ("ub" – "lb" + 1) menor que 64K, "n" se codificará como un entero binario no negativo (como se especifica en 10.3) utilizando el número mínimo de bits necesarios para codificar la "gama". Si "n" no es cero, irá seguido por un campo o lista de campos asociado, y se finalizan estos procedimientos. Si "n" es cero, no habrá ninguna otra adición a la lista de campos, y se finalizan estos procedimientos.

NOTA – Si "gama" satisface la desigualdad 2^m < "gama" <= 2^{m+1}, el número de bits de este determinante de longitud es m + 1.

10.9.4.2 Si el determinante de longitud "n" que se ha de codificar es un número entero constreñido mayor que o igual a 64K, o es un número entero semiconstreñido, "n" se codificará como se especifica en 10.9.3.4-10.9.3.8.4.

11 Codificación del tipo booleano

11.1 Un valor del tipo booleano se codificará como un campo de bits que consiste en un solo bit.

11.2 El bit se pondrá a 1 para VERDADERO (TRUE) y a 0 para FALSO (FALSE).

11.3 El campo de bits se añadirá a la lista de campos sin determinante de longitud.

12 Codificación del tipo entero

NOTAS

1 (Didáctica – Variante ALIGNED) Las gamas que permiten la codificación de todos los valores en un octeto o menos van en un campo de bits de tamaño mínimo sin cuenta de longitud. Las gamas que permiten la codificación de todos los valores en dos octetos van en dos octetos en un campo de bits alineados en octeto sin cuenta de longitud. En los demás casos, el valor se codifica en el número mínimo de octetos (utilizando la codificación de entero binario no negativo o la codificación de entero binario complemento de dos, según proceda) y se añade un determinante de longitud. En este caso, si el valor de entero se puede codificar en menos de 127 octetos (como una distancia con respecto a cualquier límite inferior que pudiera ser determinado), y no hay cotas superior e inferior finitas, habrá un determinante de longitud de un octeto, si no la longitud se codificará en el menor número de bits necesario. Los demás casos no son de interés práctico, pero se especifican en aras de la integridad.

2 (Didáctica – Variante UNALIGNED) Los enteros constreñidos se codifican en el menor número posible de bits necesarios para representar la gama con independencia de su tamaño. Los enteros no constreñidos se codifican como se indica en la Nota 1.

12.1 Si está presente un marcador de extensión en la especificación de constricciones del tipo entero, se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno. El bit se pondrá a 1 si el valor que se ha de codificar no está dentro de la gama de la constricción de raíz y se pondrá a cero en los demás casos. En el primer caso, el valor se añadirá a una lista de campos como un valor entero no constreñido, como se especifica en 12.2.4 a 12.2.6, y se finaliza este procedimiento. En el segundo caso, el valor se codificará como si el marcador de extensión no estuviese presente.

12.2 Si en la especificación de constricción de un tipo entero no está presente un marcador de extensión, se aplica lo siguiente:

12.2.1 Si constricciones visibles a PER fuerzan el valor entero a ser un valor simple, no habrá adición a la lista de campos y se finalizarán estos procedimientos.

12.2.2 Si constricciones visibles a PER fuerzan el valor entero a ser un número entero constreñido, deberá convertirse en un campo de acuerdo con los procedimientos de 10.5 (codificación de un número entero constreñido), y se aplicarán entonces los procedimientos de 12.2.5 y 12.2.6.

12.2.3 Si constricciones visibles a PER fuerzan el valor entero a ser un número entero semiconstreñido, deberá convertirse en un campo de acuerdo con los procedimientos de 10.7 (codificación de un número entero semiconstreñido), y se aplicarán entonces los procedimientos de 12.2.6.

12.2.4 Si constricciones visibles a PER no fuerzan el valor entero a ser un número entero constreñido, ni semiconstreñido, deberá convertirse en un campo de acuerdo con los procedimientos de 10.8 (codificación de un número entero no constreñido), y se aplicarán entonces los procedimientos de 12.2.6.

12.2.5 Si como resultado de la aplicación de los procedimientos invocados para codificar el valor entero en un campo no se dio el caso de la longitud indefinida (véanse 10.5.7.4 y 10.8.2), el campo se añadirá a la lista de campos y se finalizarán estos procedimientos.

12.2.6 De no ser así (caso de la longitud indefinida), se invocarán los procedimientos de 10.9 para añadir el campo a la lista de campos, que irá precedida por uno de los determinantes siguientes:

a) Un determinante de longitud constreñida "len" (establecido como se especifica en 10.5.7.4) si constricciones visibles a PER limitan el tipo y, si el tipo es extensible, el valor se sitúa dentro de la raíz de extensión. La cota inferior "lb" utilizada en el determinante de longitud será 1, y la cota superior "ub" será la cuenta del número de octetos requeridos para contener la gama del valor entero.

NOTA – La codificación del valor "foo INTEGER (256..1234567) ::= 256" se codificaría así como 00xxxxxx00000000, donde cada 'x' representa un bit de relleno de valor cero que puede o no estar presente, lo que depende de la posición, dentro del octeto, en la que caiga la longitud (por ejemplo, la codificación es 00 xxxxxx 00000000 si la longitud comienza en un límite de octeto, y 00 00000000 si comienza en el antepenúltimo bit de un octeto).

b) Un determinante de longitud no constreñida igual a "len" (establecido como se especifica en 10.7 y 10.8), si constricciones visibles a PER no restringen el tipo con cotas superior e inferior finitas, o si el tipo es extensible y los valores no están comprendidos en la gama de la raíz de extensión.

13 Codificación del tipo enumerado

NOTA – (Didáctica) Un tipo enumerado sin un marcador de extensión se codifica como si fuese un entero constreñido cuya constricción de subtipo no contiene un marcador de extensión. Esto significa que un tipo enumerado se codificará casi siempre en la práctica como un campo de bits en el menor número de bits necesarios para expresar cada enumeración. Cuando hay un marcador de extensión, se codifica como un número entero no negativo normalmente pequeño si el valor no está en la raíz de extensión.

13.1 Las enumeraciones en la raíz de enumeración deberán clasificarse en orden ascendente de su valor de enumeración y se les asignará un índice de enumeración que será cero para la primera enumeración, uno para la segunda, y así sucesivamente hasta la última enumeración en la lista clasificada. A las adiciones de extensión (que siempre se definen en orden ascendente) se les asignará un índice de enumeración que será cero para la primera enumeración, uno para la segunda, y así sucesivamente hasta la última enumeración en las adiciones de extensión.

NOTA – La Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1 requiere que cada adición de extensión sucesiva tenga un valor de enumeración mayor que el de la última.

13.2 Si el marcador de extensión está ausente en la definición del tipo enumerado, se codificará el índice de enumeración. La codificación se efectuará como si se tratara de un valor entero para el cual no está presente un marcador de extensión, donde la cota inferior es cero y la cota superior es el índice de enumeración más grande asociado con el tipo, y se finaliza este procedimiento.

13.3 Si el marcador de extensión está presente, se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno. Este bit se pondrá a 1 si el valor que se ha de codificar no está dentro de la raíz de extensión y a cero en los demás casos. En el primer caso, las adiciones de enumeración se clasificarán en consecuencia y el valor se añadirá a la lista de campos como un número entero no negativo normalmente pequeño cuyo valor es el índice de enumeración de la enumeración adicional y con "lb" puesto a cero, y se finaliza este procedimiento. En el segundo caso, el valor se codificará como si el marcador de extensión no estuviese presente, según se especifica en 13.2.

NOTA – Ninguna de las constricciones visibles a PER que pueda aplicarse a un tipo enumerado es visible a estas reglas de codificación.

14 Codificación del tipo real

NOTA – (Didáctica) Un número real utiliza los octetos de contenido de CER/DER precedidos por un determinante de longitud constituido, en la práctica, por un octeto.

14.1 Si la base del valor abstracto es 10 la base del valor codificado será 10, y si la base del valor abstracto es 2 la base del valor codificado será 2.

14.2 Se aplicará la codificación de REAL especificada para las reglas de la codificación canónica y reglas de codificación distinguida en la Rec. UIT-T X.690 | ISO/CEI 8825-1 para obtener un campo de bits alineado en octeto que constituye los octetos de contenido de la codificación CER/DER. Los octetos de contenido de esta codificación se componen de (digamos) "n" octetos y se colocan en un campo alineado en octeto de "n" octetos. Se invocarán los procedimientos de la cláusula 10.9 para añadir este campo de bits alineado en octeto de "n" octetos a la lista de campos, precedido por un determinante de longitud no constreñida igual a "n".

15 Codificación del tipo cadena de bits

NOTA – (Didáctica) Las cadenas de bits con una longitud constreñida menor o igual que dos octetos no producen alineación de octetos. Las cadenas de bits más grandes están alineadas en octetos. Si la longitud se fija mediante constricciones y el límite superior es inferior a 64K, no hay codificación de longitud explícita, en los demás casos se incluye una codificación de longitud que puede adoptar cualquiera de las formas especificadas anteriormente para las codificaciones de longitud, incluida la fragmentación para grandes cadenas de bits.

15.1 Las constricciones visibles a PER sólo pueden constreñir la longitud de la cadena de bits.

15.2 Cuando no hay constricciones visibles a PER y se aplica 19.7 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, los valores se codificarán sin bits 0 posteriores (obsérvese que esto significa que un valor sin bits 1 se codifica siempre como una cadena de bits vacía).

15.3 Cuando hay una constricción visible a PER y se aplica 19.7 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 (es decir, el tipo de cadena de bits se define con una "NamedBitList"), el valor se codificará añadiendo o suprimiendo bits 0 anteriores, según sea necesario, para asegurar que el tamaño del valor transmitido es el tamaño más pequeño capaz de transportar este valor y que satisface la constricción de tamaño efectiva.

15.4 Sea "ub" el número máximo de bits en la cadena de bits (determinados por las constricciones visibles a PER de la longitud) y "lb" el número mínimo de bits. Si no hay un máximo finito decimos que "ub" no está fijado. Si no hay constricción del mínimo, "lb" tiene el valor cero. Sea "n" bits la longitud del valor de la cadena de bits real que se ha de codificar.

15.5 Si está presente un marcador de extensión en la especificación de constricción de tamaño del tipo cadena de bits, se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno. El bit se pondrá a 1 si la longitud de esta codificación no está dentro de la gama de la raíz de extensión y a cero en los demás casos. En el primer caso, se invocará 15.10 para añadir la longitud como un número entero semiconstreñido a la lista de campos, seguido por el valor de la cadena de bits. En el segundo caso, la longitud y el valor se codificarán como si no estuviese presente el marcador de extensión.

15.6 Si no está presente un marcador de extensión en la especificación de constricciones del tipo cadena de bits, se aplican 15.7 a 15.10.

15.7 Si la cadena de bits está constreñida a ser de longitud cero ("ub" igual a cero), no se codificará (no se harán adiciones a la lista de campos) y se finalizan los procedimientos de esta cláusula.

15.8 Si todos los valores de la cadena de bits están constreñidos a tener la misma longitud ("ub" igual "lb") y esa longitud es menor o igual que dieciséis bits, la cadena de bits se colocará en un campo de bits de la longitud constreñida "ub" que se añadirá a la lista de campos sin determinante de longitud, y se finalizan los procedimientos de esta cláusula.

15.9 Si todos los valores de la cadena de bits están constreñidos a tener la misma longitud ("ub" igual "lb") y esa longitud es mayor que dieciséis bits pero menor que 64K bits, la cadena de bits se colocará en un campo de bits alineados en octeto con la longitud constreñida "ub" (que no es necesariamente un múltiplo de ocho bits), se añadirá a la lista de campos sin determinante de longitud, y se finalizan los procedimientos de esta cláusula.

15.10 Si no se aplican 15.7 a 15.9, la cadena de bits se colocará en un campo de bits alineados en octeto de longitud "n" bits y se invocarán los procedimientos de 10.9 para añadir este campo de bits alineados en octeto de "n" bits a la lista de campos, precedido por un determinante de longitud igual a "n" bits como un número entero constreñido si "ub" está fijado y es menor que 64K o como un número entero semiconstreñido si "ub" no está fijado. "lb" es como se determina anteriormente.

NOTA – La fragmentación se aplica para "ub" no constreñido o grande después de 16K, 32K, 48K o 64K bits.

16 Codificación del tipo cadena de octetos

NOTA – Las cadenas de octetos longitud fija menor o igual que dos octetos no están alineadas en octetos. Todas las otras cadenas de octetos están alineadas en octetos. Las cadenas de octetos de longitud fija se codifican sin octetos de longitud si son más cortas que 64K. Para cadenas de octetos no constreñidas, la longitud se codifica explícitamente (con fragmentación, si es necesario).

16.1 Las constricciones visibles a PER sólo pueden constreñir la longitud de la cadena de octetos.

16.2 Sea "ub" el número máximo de octetos en la cadena de octetos (determinado por las constricciones visibles a PER de la longitud) y "lb" el número mínimo de octetos. Si no hay un máximo finito decimos que "ub" no está fijado. Si no hay constricciones del mínimo, "lb" tiene el valor cero. Sea "n" octetos la longitud del valor de cadena de octetos real que se ha de codificar.

16.3 Si hay una constricción de tamaño visible a PER y el marcador de extensión está presente en ella, se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno. El bit se pondrá a 1 si la longitud de esta codificación no está dentro de la gama de la raíz de extensión, y a cero en los demás casos. En el primer caso, se invocará 16.8 para añadir la longitud como un número entero semiconstreñido a la lista de campos, seguido por el valor de cadena de octetos. En el segundo caso, la longitud y el valor se codificarán como si el marcador de extensión no estuviese presente.

16.4 Si no está presente un marcador de extensión en la especificación de constricciones del tipo cadena de octetos, se aplican 16.5 a 16.8.

16.5 Si la cadena de octetos está constreñida a ser de longitud cero ("ub" igual a cero) no se codificará (no se efectuarán adiciones a la lista de campos), y se finalizan los procedimientos de esta cláusula.

16.6 Si todos los valores de la cadena de octetos están constreñidos a tener la misma longitud ("ub" igual a "lb") y la longitud es menor o igual que dos octetos, la cadena de octetos se colocará en un campo de bits con un número de bits igual a la longitud constreñida "ub" multiplicada por ocho, lo que se añadirá a la lista de campos sin determinante de longitud, y se finalizan los procedimientos de esta cláusula.

16.7 Si todos los valores de la cadena de octetos están constreñidos a tener la misma longitud ("ub" igual a "lb") y esa longitud es mayor que dos octetos pero menor que 64K, la cadena de octetos se colocará en un campo de bits alineados en octeto y se añadirán los octetos "ub" de longitud constreñida a la lista de octetos sin determinante de longitud, y se finalizan los procedimientos de esta cláusula.

16.8 Si no se aplican 16.5 a 16.7, la cadena de octetos se colocará en un campo de bits alineados en octeto de longitud de "n" octetos y se invocarán los procedimientos de 10.9 para añadir este campo de bits alineados en octeto de "n" octetos a la lista de campos, precedido por un determinante de longitud igual a "n" octetos como un número entero constreñido si "ub" está fijado, y como un número entero semiconstreñido si "ub" no está fijado. "lb" es como se determina anteriormente.

NOTA – Los procedimientos de fragmentación se pueden aplicar después de 16K, 32K, 48K o 64K octetos.

17 Codificación del tipo nulo

NOTA – (Didáctica) El tipo nulo es esencialmente un guardador de puesto ("place holder"), que sólo tiene significado práctico en el caso de un componente de elección o de conjunto o secuencia facultativo. La identificación de nulo en una elección, o su presencia como un elemento facultativo se realiza en estas reglas de codificación sin necesidad de tener octetos que representen el nulo. Por consiguiente, los valores nulos nunca contribuyen a los octetos de una codificación.

No se efectuarán adiciones a la lista de campos para un valor nulo.

18 Codificación del tipo secuencia

NOTA – (Didáctica) Un tipo secuencia comienza con un preámbulo que es una tabla de correspondencia de bits. Si el tipo secuencia no tiene marcador de extensión, la tabla de correspondencia de bits sólo registra la presencia o ausencia de componentes por defecto y facultativos en el tipo, codificados como un campo de bits de longitud fija. Si el tipo secuencia sí tiene un marcador de extensión, la tabla de correspondencia de bits está precedida por un solo bit que dice si están realmente presentes en la codificación valores de adiciones de extensión. El preámbulo se codifica sin ningún determinante de longitud a condición de que su longitud sea menor que 64K bits; en los demás casos, se codifica un determinante de longitud para obtener la fragmentación. El preámbulo está seguido por los campos que codifican cada uno de los componentes, tomados en turno. Si no hay adiciones de extensión, inmediatamente antes de que se codifique el primer uno, hay la codificación (como un número entero semiconstreñido) de una cuenta del número de adiciones de extensión en el tipo que se codifica, seguido por una tabla de correspondencia de bits de longitud igual a esta cuenta que registra la presencia o ausencia de valores de cada adición de extensión. Esto va seguido por las codificaciones de las adiciones de extensión como si cada una fuese el valor de un campo de tipo abierto.

18.1 Si el tipo secuencia tiene un marcador de extensión, se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno. El bit será uno si los valores de adiciones de extensión están presentes en esta codificación y cero en los demás casos. (Este bit se denomina "bit de extensión" en el texto que sigue.) Si no hay marcador de extensión, no se añadirá ningún bit de extensión.

18.2 Si el tipo secuencia tiene "n" componentes en la raíz de extensión que están marcados OPTIONAL (facultativo) o DEFAULT (por defecto), se producirá un campo de bits con "n" bits para añadirlo a la lista de campos. Los bits del campo de bits codificarán, tomados en orden, la presencia o ausencia de una codificación de cada componente facultativo o por defecto en el tipo secuencia. Un valor de bit de uno codificará la presencia de la codificación del componente y un valor de bit de cero codificará la ausencia de la codificación del componente. El bit anterior en el preámbulo codificará la presencia o ausencia del primer componente facultativo o por defecto y el bit posterior codificará la presencia o ausencia del último componente facultativo o por defecto.

18.3 Si "n" es menor que 64K, el campo de bits se añadirá a la lista de campos. Si "n" es mayor que o igual a 64K, se invocarán los procedimientos de 10.9 para añadir este campo de bits de "n" bits a la lista de campos, precedido por un determinante de longitud igual a "n" bits como un número entero constreñido con "ub" y "lb" puestos a "n".

NOTA – En este caso, los procedimientos de longitud pasarán por alto "ub" y "lb". Estos procedimientos se invocan para proporcionar la fragmentación de un preámbulo grande. Se prevé que esta situación ocurra sólo raras veces.

18.4 El preámbulo será seguido por una lista de campos de cada uno de los componentes del valor de secuencia que están presentes, tomados en turno.

18.5 Para CANONICAL-PER nunca habrá codificaciones de componentes marcados como DEFAULT si el valor que se va a codificar es el valor por defecto. Para BASIC-PER nunca habrá codificaciones de componentes marcados como DEFAULT si el valor que se va a codificar es el valor por defecto de un tipo simple (véase 3.8.25); en todos los demás casos se codificará únicamente si está explícitamente presente en el valor de secuencia abstracto.

18.6 Esto termina la codificación si el bit de extensión está ausente o es cero. Si el bit de extensión está presente y está puesto a uno, se aplicarán los siguientes procedimientos.

18.7 Sea "n" el número de adiciones de extensión en el tipo que se codifica, entonces se producirá un campo de bits con "n" bits para añadirlo a la lista de campos. Los bits del campo de bits codificarán, tomados en orden, la presencia o ausencia de una codificación de cada adición de extensión en el tipo que se codifica. Un valor de bit de uno codificará la presencia de la codificación de la adición de extensión, y un valor de bit de cero codificará la ausencia de la codificación de la adición de extensión. El bit anterior en el campo de bits codificará la presencia o ausencia de la primera adición de extensión y el bit posterior codificará la presencia o ausencia de la última adición de extensión.

18.8 Se invocarán los procedimientos de 10.9 para añadir este campo de bits de "n" bits a la lista de campos, precedido por un determinante de longitud igual a "n" como una longitud normalmente pequeña.

NOTA – "n" no puede ser cero, porque este procedimiento se invoca solamente si hay por lo menos una adición de extensión que se codifica.

18.9 Esto será seguido por listas de campos que contienen las codificaciones de cada adición de extensión que está presente, tomadas en turno, codificándose cada extensión como si fuese el valor de un campo de tipo abierto, como se especifica en 10.2.1.

19 Codificación del tipo secuencia de

19.1 Las constricciones visibles a PER pueden constreñir el número de componentes del tipo secuencia de.

19.2 Sea "ub" el número máximo de componentes en la secuencia de (determinados por constricciones visibles a PER) y "lb" el número mínimo de componentes. Si no hay un máximo finito o "ub" es mayor o igual que 64K, decimos que "ub" no está fijado. Si no hay constricciones sobre el mínimo, "lb" tiene el valor cero. Sea "n" componentes el número de componentes en el valor de secuencia real que se ha de codificar.

19.3 La codificación de cada componente de la secuencia generará varios campos que se añadirán a la lista de campos para el tipo secuencia de.

19.4 Si hay una constricción visible a PER y está presente en ella un marcador de extensión, se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno. El bit se pondrá a uno si el número de componentes en esta codificación no está dentro de la gama de la raíz de extensión, y a cero en los demás casos. En el primer caso, se

invocará 19.7 para añadir la longitud como un número entero semiconstreñido a la lista de campos, seguido por los valores de componente. En el segundo caso, la longitud y el valor se codificarán como si el marcador de extensión no estuviese presente.

19.5 Si el número de componentes es fijo ("ub" es igual a "lb") y "ub" es menor que 64K, no habrá determinante de longitud para la secuencia de y los campos de cada componente se añadirán en turno a la lista de campos de la secuencia de.

19.6 En los demás casos, se invocarán los procedimientos de 10.9 para añadir la lista de campos generada por los "n" componentes a la lista de campos, precedida por un determinante de longitud igual a "n" componentes como un número entero constreñido si "ub" está fijado, y como un número entero semiconstreñido si "ub" no está fijado. "lb" es como se determina anteriormente.

NOTAS

1 Se pueden aplicar los procedimientos de fragmentación después de componentes de 16K, 32K, 48K o 64K.

2 Los puntos de corte para la fragmentación están entre campos. El número de bits antes de un punto de corte no es necesariamente un múltiplo de ocho.

20 Codificación del tipo conjunto

El tipo conjunto deberá tener los elementos de su "RootComponentTypeList" clasificados en el orden canónico especificado en 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, además, a los efectos de determinar el orden en que se codifican los componentes cuando uno o más componentes son de un tipo elección no rotulado, cada tipo elección no rotulado se ordena como si tuviera un rótulo igual al rótulo más pequeño en la "RootAlternativeTypeList" de ese tipo elección o de cualesquiera tipos elección no rotulados que estuviesen anidados. Los elementos de conjunto que aparecen en la "AdditionalComponentTypeList" se clasificarán en el orden canónico especificado en 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1. El valor de conjunto se codificará entonces como si hubiera sido declarado de tipo secuencia.

EJEMPLO – En lo que sigue, se supone un entorno rotulado de IMPLICIT TAGS (RÓTULOS IMPLICITOS):

```
A ::= SET
{
    a    [3] INTEGER,
    b    [1] CHOICE
    {
        c    [2] INTEGER,
        d    [4] INTEGER
    },
    e    CHOICE
    {
        f    CHOICE
        {
            g    [5] INTEGER,
            h    [6] INTEGER
        },
        i    CHOICE
        {
            j    [0] INTEGER
        }
    }
}
```

el orden en el que se codifican los componentes del conjunto será siempre e, b, a, porque el rótulo [0] es el que clasifica más bajo, le sigue [1], después [3].

NOTA – La Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1 requiere que cada adición de extensión sucesiva tenga un valor de rótulo mayor que la última añadida a la "AdditionalComponentTypeList".

21 Codificación del tipo conjunto de

21.1 Para CANONICAL-PER la codificación de los valores de componentes del tipo conjunto-de aparecerán en orden ascendente, las codificaciones de componentes se comparan como cadenas de bits rellenadas con bits 0, hasta un máximo de siete, para alcanzar un límite de octeto.

NOTA – Cualesquiera bits de relleno añadidos para lograr la alineación en octeto para la clasificación no aparece en la codificación real.

21.2 Para BASIC-PER, el conjunto-de se codificará como si hubiese sido declarado un tipo secuencia de.

22 Codificación del tipo elección

NOTA – (Didáctica) Un tipo elección se codifica codificando un índice que especifica la alternativa elegida. Esta se codifica como para un entero constreñido (a menos que el marcador de extensión esté presente en el tipo elección, en cuyo caso es un número entero no negativo normalmente pequeño) y por tanto ocuparía típicamente un campo de bits de longitud fija del número mínimo de bits necesario para codificar el índice. (Aunque en principio pudiera ser arbitrariamente grande.) Esto va seguido por la codificación de la alternativa elegida, con alternativas que son adiciones de extensión codificadas como si fuesen el valor de un campo de tipo abierto. Cuando la elección sólo tiene una alternativa, no hay codificación para el índice.

22.1 La codificación de los tipos elección no son afectadas por constricciones visibles a PER.

22.2 Cada componente de una elección tiene un índice asociado que tiene el valor cero para la primera alternativa en la raíz de la elección (tomando las alternativas en el orden canónico especificado en 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, uno para el segundo y así sucesivamente hasta el último componente en la extensión de raíz de la elección. Un valor de índice se asigna de manera similar a cada adición de extensión, comenzando con 0 al igual que para los componentes de la raíz de extensión. Sea "n" el valor del índice mayor en la raíz.

NOTA – La Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1 requiere que cada adición de extensión sucesiva tenga un valor de rótulo mayor que el de la última añadida a la "AdditionalAlternativeTypeList".

22.3 A los efectos de la ordenación canónica de alternativas de elección que contienen una elección no rotulada, cada tipo elección no rotulado se ordenará como si tuviese un rótulo igual al rótulo más pequeño en la raíz de extensión de ese tipo elección, o de cualesquiera tipos elección no rotulados anidados dentro de aquéllos.

22.4 Si la elección sólo tiene una alternativa en la raíz de extensión, no habrá codificación para el índice si se elige esta alternativa.

22.5 Si el tipo elección tiene un marcador de extensión, se añadirá primero un bit a la lista de campos en un campo de bits de longitud uno. El bit será 1 si está presente un valor de adición de extensión en la codificación y cero en los demás casos. (Este bit se denomina "bit de extensión" en el texto que sigue.) Si no hay marcador de extensión, no se añadirá ningún bit de extensión.

22.6 Si el bit de extensión está ausente, el índice de elección de la alternativa elegida se codificará en un campo de acuerdo con los procedimientos de la cláusula 12 como si fuese un valor de un tipo entero (sin marcador de extensión en su constricción de subtipo) constreñido a la gama 0 a "n" y ese campo se añadirá a la lista de campos, seguido por los campos de la alternativa elegida, y se ejecutan los procedimientos de esta cláusula.

22.7 Si el bit de extensión está presente y la alternativa elegida está dentro de la raíz de extensión, el índice de elección de la alternativa elegida se codificará como si el marcador de extensión estuviese ausente, de acuerdo con los procedimientos de 22.6, y se ejecutan los procedimientos de esta cláusula.

22.8 Si el bit de extensión está presente y la alternativa elegida no está dentro de la raíz de extensión, el índice de elección de la alternativa elegida se codificará como un número entero no negativo normalmente pequeño con "lb" puesto a 0 y ese campo se añadirá a la lista de campos, seguido por una lista de campos que contiene la codificación de la alternativa elegida codificada como si fuese el valor de un campo de tipo abierto especificado en 10.2, y se ejecutan los procedimientos de esta cláusula.

23 Codificación del tipo identificador de objeto

NOTA – (Didáctica) Una codificación de tipo identificador de objeto utiliza los octetos de contenido de BER precedidos por un determinante de longitud que en la práctica será un solo octeto.

La codificación especificada para BER se aplicará para dar un campo de bits alineados en octeto que es los octetos de contenido de la codificación BER. Los octetos de contenido de esta codificación BER consisten en (digamos) "n" octetos y se colocan en un campo de bits alineados en octeto de "n" octetos. Se invocarán los procedimientos de 10.9 para añadir este campo de bits alineados en octeto a la lista de campos, precedido por un determinante de longitud igual a "n" como una cuenta de octetos de número entero semiconstreñido.

24 Codificación del tipo pdv insertado

NOTA – (Didáctica) El tipo pdv insertado se codifica usualmente mediante un bit de bandera en su comienzo, seguido por un número entero no negativo normalmente pequeño que codifica un "índice", facultativamente seguido por la "identificación" y finalmente seguido por el "valor de datos". El bit de bandera determina cuál de las dos subreglas está en uso en el resto del valor del pdv insertado. Este bit se pone a 1 para indicar que se emplea la regla de "fijación de índice" EP-A y se pone a 0 para indicar que se emplea la subregla "utilización de índice". La subregla "fijación de índice" se utiliza en la primera ocurrencia de una determinada "identificación" en el valor en sintaxis abstracta. Para todas las ocurrencias subsiguientes de ese valor de "identificación" dentro del valor de sintaxis abstracta en el que se define el tipo pdv insertado, se emplea la subregla "utilización de índice". Cuando se utiliza la subregla "fijación de índice", se codifica el valor de "identificación" completo. Un valor de "índice" que aparece con la bandera puesta a 0 en una codificación de pdv insertado tiene que aparecer precisamente una vez como un valor de "índice" con la bandera puesta a 1. Para CANONICAL-PER, sólo se permite la alternativa de "sintaxis" de "identificación" (explícitamente o mediante constricciones de ASN.1 sobre el tipo pdv insertado). Hay una optimización adicional: si la sintaxis abstracta y la sintaxis de transferencia están constreñidas por las constricciones visibles a PER, a ser un solo valor, no hay un bit de bandera único, ni "índice", ni "identificación" presentes en la codificación.

24.1 Hay tres maneras de codificar un tipo pdv insertado:

a) la alternativa de "sintaxis" del tipo pdv insertado se constriñe a un solo valor o se constriñe la "identificación" para que sea una alternativa "fija", en cuyo caso sólo se codificará el valor de datos; esta alternativa se denomina la opción "predefinida";

b) el primer bit del valor pdv insertado se pone a 1, en cuyo caso se prefijará un valor de "índice" a la "identificación" que estará presente, seguida por el "valor de datos"; esta alternativa se denomina la subregla de "fijación de índice" o EP-A;

c) el primer bit del valor pdv insertado se pone a 0, en cuyo caso estarán presentes un valor de "índice" y un "valor de datos" pero la "identificación" estará ausente; esta alternativa se denomina la subregla "utilización de índice" o EP-B.

24.2 La opción "predefinida" se utilizará cuando la "identificación" está constreñida a la alternativa "fija" o cuando la alternativa de "sintaxis" de "identificación" está constreñida a un solo valor. Cuando se utiliza la opción "predefinida", la codificación completa del "valor de datos" se añadirá a un campo de bits alineados en octeto y se invocará 10.9.3.5 para añadirlo a la lista de campos, prefijado con la longitud como un número entero semiconstreñido. Esto completa los procedimientos para esta cláusula.

24.3 Cuando no se selecciona la opción "predefinida", se añadirá un solo bit a la lista de campos en un campo de bits de longitud uno y se pone a uno. Este bit se pondrá a uno si se está empleando la subregla "fijación de índice", en los demás casos se pondrá a 0.

24.4 La subregla de "fijación de índice" se utilizará la primera vez que se especifica un valor abstracto dado para "identificación". Cuando se utiliza la subregla "fijación de índice", se invocará 10.6 para añadir un valor de índice a la lista de campos como un número entero no negativo normalmente pequeño. El primer valor de índice será 0 y será incrementado en 1 para cada valor de "identificación" distinto que se encuentra. Se invocará después la cláusula 22 para añadir la "identificación" a la lista de campos. La codificación completa del "valor de datos" se añadirá después a un campo de bits alineados en octeto y se invocará 10.9 para añadirlo a la lista de campos, prefijado con la longitud como un número entero semiconstreñido. Esto completa los procedimientos de esta cláusula.

24.5 La subregla "utilización de índice" se empleará en todas las ocurrencias subsiguientes de un valor abstracto dado para "identificación". Cuando se emplea la subregla "utilización de índice" se invocará 10.6 para añadir un valor de índice a la lista de campos como un número entero no negativo normalmente pequeño. Este valor de índice corresponderá al que se añadió previamente a la lista de campos para este valor abstracto de "identificación" dado. La codificación completa del "valor de datos" se añadirá después a un campo de bits alineados en octeto y se invocará 10.9 para añadirlo a la lista de campos, prefijado con la longitud como un número entero semiconstreñido. Esto completa los procedimientos para esta cláusula.

24.6 EXPLICACIÓN DIDÁCTICA: De este modo, para cualquier valor de "identificación" dado hay una codificación EP-A (relativamente ineficaz) con un valor de índice único y el valor de "identificación" completo, seguido por número arbitrariamente grande de codificaciones EP-B (eficaces) vinculadas a la codificación EP-A por el valor de índice.

24.7 La codificación EP-A será la codificación PER del bit de fijación de índice/utilización de índice, seguido por el índice, seguido por el siguiente tipo secuencia para el cual se especifica AUTOMATIC TAGS (rótulos automáticos) (véanse 22.6 y 26.3 de la Rec. UIT-T X.680 | ISO/CEI 8824-1):

```
SEQUENCE {
    identification              CHOICE {
        syntaxes                    SEQUENCE {
            abstract                    OBJECT IDENTIFIER,
            transfer                    OBJECT IDENTIFIER },
        syntax                      OBJECT IDENTIFIER,
        presentation-context-id     INTEGER,
        context-negotiation         SEQUENCE {
            presentation-context-id     INTEGER,
            transfer-syntax             OBJECT IDENTIFIER },
        transfer-syntax             OBJECT IDENTIFIER,
        fixed                       NULL },
    data-value                  BIT STRING }
```

24.8 El valor de "valor de datos" será la codificación del valor de datos abstracto que utiliza la sintaxis de transferencia identificada, y el valor de todos los otros campos se codificará utilizando la misma sintaxis de transferencia que los otros valores que aparecen en el valor de sintaxis abstracta.

NOTA – Las dos alternativas de "valor de datos" en la sintaxis abstracta se codifican idénticamente – como una cadena de bits – en la sintaxis de transferencia.

25 Codificación de un valor del tipo externo

25.1 La codificación de un valor del tipo externo será la codificación PER del siguiente tipo secuencia, que se supone se define en un entorno de EXPLICIT TAGS (rótulos explícitos) con un valor especificado en las cláusulas siguientes:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference            OBJECT IDENTIFIER OPTIONAL,
    indirect-reference          INTEGER OPTIONAL,
    data-value-descriptor       ObjectDescriptor OPTIONAL,
    encoding                    CHOICE {
        single-ASN1-type                [0] ABSTRACT-SYNTAX.&Type,
        octet-aligned                   [1] IMPLICIT OCTET STRING,
        arbitrary                       [2] IMPLICIT BIT STRING } }
```

NOTA – Este tipo de secuencia es el mismo especificado en la Rec. X.208 del CCITT (1988) | ISO/CEI 8824 (1990).

25.2 El valor de los componentes depende del valor abstracto que se transmite, que es un valor del tipo especificado en 30.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

25.3 El "descriptor de valor de datos" anterior estará presente solamente si el "descriptor de valor de datos" está presente en el valor abstracto, y tendrá el mismo valor.

25.4 Los valores de "referencia directa" y "referencia indirecta" anteriores estarán presentes o ausentes de acuerdo con el Cuadro 1, que muestra la correspondencia de las alternativas de tipo externo de "identificación" definidas en la Rec. UIT-T X.680 | ISO/CEI 8824-1, cláusula 30.5, con los componentes del tipo externo "referencia directa" y "referencia indirecta" definidos en 8.18.1.

Cuadro 1 – Codificaciones alternativas para "identificación"

identificación	referencia directa	referencia indirecta
syntaxes	*** CANNOT OCCUR ***	*** CANNOT OCCUR ***
syntax	syntax	ABSENT
presentation-context-id	ABSENT	presentation-context-id
context-negotiation	transfer-syntax	presentation-context-id
transfer-syntax	*** CANNOT OCCUR ***	*** CANNOT OCCUR ***
fixed	*** CANNOT OCCUR ***	*** CANNOT OCCUR ***

25.5 El valor de datos se codificará de acuerdo con la sintaxis de transferencia identificada por la codificación, y se colocará en una alternativa de la elección "codificación" como se especifica a continuación.

25.6 Si el valor de datos es el valor de un tipo de datos ASN.1 simple, y si las reglas de codificación para este valor de datos son las mismas que para el tipo de datos "EXTERNAL" completo, la implementación emisora será "tipo ASN.1 simple".

25.7 Si la codificación, convenida o negociada, del valor de datos es un número integral de octetos, la implementación emisora será "alineado en octetos".

NOTA – Un valor de datos que es una serie de tipos ASN.1, y para los cuales la sintaxis de transferencia especifica concatenación simple de las cadenas de octetos producidas aplicando las reglas de codificación básica de ASN.1 a cada tipo ASN.1, está en esta categoría, no en la indicada en 25.6.

25.8 Si la codificación, convenida o negociada, del valor de datos no es un número integral de octetos, la elección de "codificación" será "arbitraria".

25.9 Si la elección de "codificación" es "tipo ASN.1 simple", el tipo ASN.1 reemplazará al tipo abierto, con un valor igual al valor de datos que se ha de codificar.

NOTA – La gama de valores que pudiera producirse en el tipo abierto es determinada por el registro del valor de identificador de objeto asociado con la "referencia directa" y/o el valor entero asociado con la "referencia indirecta".

25.10 Si la elección de "codificación" es "alineada en octetos", el valor de datos se codificará de acuerdo con la sintaxis de transferencia convenida o negociada, y los octetos resultantes formarán el valor de la cadena de octetos.

25.11 Si la elección de "codificación" es "arbitraria", el valor de datos se codificará de acuerdo con la sintaxis de transferencia convenida o negociada y el resultado formará el valor de la cadena de bits.

26 Codificación de los tipos cadenas de caracteres restringidas

NOTAS

1 (Explicación de la variante ALIGNED) Las cadenas de caracteres de longitud fija menor o igual que dos octetos no están alineadas en octetos. Las cadenas de caracteres de longitud variable que están constreñidas a tener una longitud máxima menor que dos octetos no están alineadas en octetos. Todas las demás cadenas de caracteres están alineadas en octetos. Las cadenas de caracteres de longitud fija se codifican sin octetos de longitud si son más cortas que 64K. Para las cadenas de caracteres no constreñidas o cadenas de caracteres constreñidas más largas que 64K-1, la longitud se codifica explícitamente (con fragmentación si es necesario). Cada carácter NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString y UniversalString se codifica en el número de bits que es la potencia de dos más pequeña que puede acomodar todos los caracteres permitidos por la constricción de alfabeto permitido efectiva.

2 (Explicación de la variante UNALIGNED) Las cadenas de caracteres no están alineadas en octetos. Si hay sólo un valor de longitud posible no hay codificación de longitud si son más cortas que 64K caracteres. Para cadenas de caracteres no constreñidas o cadenas de caracteres constreñidas más largas que 64K-1, la longitud se codifica explícitamente (con fragmentación si es necesario). Cada carácter NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString y UniversalString se codifica en el número de bits más pequeño que pueda acomodar todos los caracteres permitidos por la constricción alfabeto permitido efectiva.

3 (Explicación del tamaño de cada carácter codificado) La codificación de cada carácter depende de la constricción alfabeto permitido efectiva (véase 9.3.10) que define el alfabeto en uso para el tipo. Supóngase que este alfabeto consiste (digamos) en un juego de caracteres ALPHA. Para cada uno de los tipos cadena de caracteres de multiplicador conocido (véase 3.8.16) hay un valor entero asociado con cada carácter, obtenido por referencia a alguna tabla de códigos asociada con el tipo cadena de caracteres restringida. El conjunto de valores BETA (digamos) correspondiente al conjunto de caracteres ALPHA se utiliza para determinar la codificación que se ha de emplear, como sigue: el número de bits para la codificación de cada carácter es determinado únicamente por el número de elementos, N, en el juego BETA (o ALPHA). Para la variante UNALIGNED, es el número más pequeño de bits que puede codificar el valor N-1 como un entero binario no negativo. Para la variante ALIGNED, es el número más pequeño de bits que es una potencia de dos y que puede codificar el valor N-1. Supóngase que el número seleccionado de bits es B. Si cada valor en el juego BETA puede ser codificado (sin transformación) en B bits, el valor en el juego BETA se utiliza para representar los caracteres correspondientes en el juego ALPHA. En los demás casos, los valores en el juego BETA se toman en orden ascendente y se sustituyen por valores 0, 1, 2 y así sucesivamente hasta N-1, y éstos son los valores que se utilizan para representar el carácter correspondiente. En resumen: se utiliza siempre el número mínimo de bits (tomados a la siguiente potencia de dos para la variante ALIGNED). Se prefiere utilizar el valor normalmente asociado con el carácter, pero si cualquiera de estos valores no puede ser codificado en el número mínimo de bits, se aplica una compactación.

26.1 Los siguientes tipos de cadenas de caracteres restringidos son tipos de cadenas de caracteres de multiplicador conocido: NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString, UniversalString. Las constricciones efectivas de alfabeto permitido son visibles a PER solamente para estos tipos.

26.2 La notación de constricción de tamaño efectiva puede determinar un límite superior "aub" para la longitud de la cadena de caracteres abstracta. En los demás casos, "aub" no está fijado.

26.3 La notación de constricción de tamaño efectiva puede determinar un límite inferior diferente de cero "alb" para la longitud de la cadena de caracteres abstracta. En los demás casos, "alb" es cero.

NOTA – Las constricciones visibles a PER sólo se aplican a tipos de cadena de caracteres de multiplicador conocido. Para otros tipos de cadenas de caracteres restringidas, "aub" no se fijará y "alb" será cero.

26.4 Si el tipo es extensible para codificaciones PER (véase 9.3.15), se añadirá a la lista de campos un campo de bits que consiste en un solo bit. El bit se pondrá a cero si el valor está dentro de la gama de la raíz de extensión y a uno en los demás casos. Si el valor está fuera de la gama de la raíz de extensión, la siguiente codificación será como si no hubiese una constricción de tamaño efectiva ni una constricción de alfabeto permitido efectivo.

NOTA – Sólo los tipos de cadenas de caracteres de multiplicador conocido pueden ser extensibles para codificaciones PER. Los marcadores de extensibilidad en otros tipos de cadenas de caracteres no afectan a la codificación PER.

26.5 Esta subcláusula se aplica a cadenas de caracteres de multiplicador conocido. La codificación de otros tipos de cadenas de caracteres restringidas se especifican en 26.6.

26.5.1 El alfabeto permitido efectivo se define como el alfabeto permitido por la constricción alfabeto permitido o todo el alfabeto del tipo incorporado si no hay constricción de alfabeto permitido.

26.5.2 Sea N el número de caracteres en el alfabeto permitido efectivo. Sea B el entero más pequeño tal que 2 a la potencia B es mayor o igual que N. Sea B2 la potencia más pequeña de 2 que es mayor que o igual a B. Entonces, en la variante ALIGNED, cada carácter se codificará en B2 bits y en la variante UNALIGNED en B bits. Sea "b" el número de bits identificado por esta regla.

26.5.3 Se asocia un valor numérico "v" con cada carácter por referencia a la cláusula 36 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, como sigue. Para UniversalString, el valor es el utilizado para determinar el orden canónico en 36.4 (el valor está en la gama 0 a $2^{32} – 1$). Para BMPString, el valor es el utilizado para determinar el orden canónico en 36.5 (el valor está en la gama 0 a $2^{16} – 1$). Para NumericString, PrintableString, VisibleString y IA5String, el valor es el definido para la codificación según ISO 646 del carácter correspondiente. (Para IA5String, la gama es 0 a 127, para VisibleString es 32 a 126, para NumericString es 32 a 57 y para PrintableString es 32 a 122. Para IA5String y VisibleString, todos los valores en la gama están presentes, pero para NumericString y PrintableString, no todos los valores en la gama están en uso.)

26.5.4 Sea "lb" el valor más pequeño de la gama para el juego de caracteres en el alfabeto permitido y "ub" el valor más grande. La codificación de un carácter en "b" bits es la codificación de entero binario no negativo del valor "v" identificado como sigue:

a) si "ub" es menor o igual que $2^b – 1$, "v" es el valor especificado en 26.5.3 anterior; en los demás casos,

b) los caracteres se colocan en el orden canónico definido en la cláusula 36 de la Rec. UIT-T X.680 | ISO/CEI 8824-1. Al primero se asigna el valor cero y al siguiente en orden canónico se asigna un valor que es mayor en una unidad que el valor asignado al carácter anterior en el orden canónico. Estos son los valores "v".

NOTA – a) no se puede aplicar nunca a un carácter NumericString constreñido o no constreñido, que siempre se codifica en cuatro bits o menos utilizando b).

26.5.5 La codificación de toda la cadena de caracteres se obtendrá codificando cada carácter (utilizando un valor "v" apropiado) como un entero binario no negativo en "b" bits que se concatenarán para formar un campo de bits que es un múltiplo de "b" bits.

26.5.6 Si "aub" es igual a "alb" y es menor que 64K, el campo de bits se añadirá a la lista de campos como un campo alineado en octetos si "aub" veces "b" es mayor que 16, pero en los demás casos se añadirá como un campo de bits que no está alineado en octetos. Esto completa los procedimientos de esta subcláusula.

26.5.7 Si "aub" no es igual a "alb" o es mayor que o igual a 64K, se invocará 10.9 para añadir un determinante de longitud con "n" como una cuenta de los caracteres en la cadena de caracteres con un límite inferior para el determinante de longitud de "alb" y un límite superior de "aub". Se añadirá entonces el campo de bits como un campo alineado en octetos si "aub" veces "b" es mayor o igual que 16, pero en los demás casos se añadirá como un campo de bits que no está alineado en octetos. Esto completa los procedimientos de esta subcláusula.

26.6 Esta subcláusula se aplica a cadenas de caracteres que no son cadenas de caracteres de multiplicador conocido. En este caso, las constricciones no son nunca visibles a PER y el tipo nunca puede ser extensible para la codificación PER.

26.6.1 Para BASIC-PER, la referencia siguiente a "codificación básica" significa BER. Para CANONICAL-PER significa CANONICAL-BER.

26.6.2 La "codificación básica" se aplicará a la cadena de caracteres para dar un campo de "n" octetos.

26.6.3 Se invocará 10.9 para añadir un determinante de longitud no constreñido con "n" como una cuenta en octetos y el campo de "n" octetos se añadirá como un campo de bits alineados en octetos, y se ejecutan los procedimientos de esta subcláusula.

27 Codificación del tipo cadena de caracteres no restringida

NOTA – (Didáctica) El tipo cadena de caracteres no restringida se codifica por lo general con un bit de bandera único a su comienzo, seguido por un número entero no negativo normalmente pequeño que codifica un "índice", facultativamente seguido por la "identificación", y finalmente seguido por el "valor de cadena". El bit de bandera único determina cuál de dos subreglas se utiliza en el resto del valor de cadena de caracteres no restringida. Este bit se pone a 1 para indicar que se utiliza la subregla de "fijación de índice" EP-A, y se pone a 0 para indicar que se utiliza la subregla de "utilización de índice". La subregla de "fijación de índice" se emplea al aparecer por primera vez una determinada "identificación" en el valor de sintaxis abstracta. Para todas las apariciones siguientes de ese valor de "identificación", dentro del valor de sintaxis abstracta en el que está definido el tipo cadena de caracteres no restringida, se emplea la subregla de "utilización de índice". Cuando se emplea la subregla de "fijación de índice", se codifica el valor "identificación" completo. Un valor "índice" que aparezca con la bandera puesta a 0 en una codificación de cadena de caracteres no restringida deberá aparecer exactamente una vez como un valor índice con la bandera puesta a 1. Para CANONICAL-PER, sólo la alternativa "sintaxis" de "identificación" está autorizada (explícitamente o por medio de constricciones ASN.1 sobre el tipo cadena de caracteres no restringida). Hay una optimización adicional: si tanto la sintaxis abstracta como la sintaxis de transferencia están forzadas por constricciones visibles a PER a ser un valor único, o si "identificación" está constreñida a la alternativa "fija", no estará entonces presente en la codificación ningún bit de bandera único, ni índice, ni "identificación".

27.1 Un tipo de cadena de caracteres no restringida se puede codificar de tres maneras:

a) la alternativa "sintaxis" del tipo de cadena de caracteres no restringida está constreñida a un solo valor o "identificación" está constreñida a la alternativa "fija", en cuyo caso sólo se codificará el "valor de cadena"; esta alternativa se denomina la opción "predefinida";

b) el primer bit del valor de la cadena de caracteres no restringida se pone a 1, en cuyo caso se prefijará un valor "índice" a la "identificación" que estará presente, seguida por el "valor de cadena"; esta alternativa se denomina la subregla de "fijación de índice", o EP-A;

c) el primer bit del valor de la cadena de caracteres no restringida se pone a 0, en cuyo caso estarán presentes un valor "índice" y un "valor de cadena" pero la "identificación" estará ausente; esta alternativa se denomina la subregla de "utilización de índice" o EP-B.

27.2 La opción "predefinida" se utilizará cuando la "identificación" está constreñida a la alternativa "fija" o cuando la alternativa "sintaxis" está constreñida a un solo valor. Cuando se utiliza la opción "predefinida", la codificación completa del "valor de cadena" se añadirá a un campo de bits alineados en octetos y se invocará 10.9.3.5 para añadirlos a la lista de campos, prefijada con la longitud como un número entero semiconstreñido. Esto completa los procedimientos para esta subcláusula.

27.3 Cuando no se selecciona la opción "predefinida", se añadirá un solo bit a la lista de campos en un campo de bits de longitud 1. Este bit se pondrá a 1 si se está empleando la subregla "fijación de índice"; en los demás casos se pondrá a 0.

27.4 La subregla de "fijación de índice" se utilizará la primera vez que se especifica un determinado valor abstracto para "identificación". Cuando se emplea la subregla "fijación de índice", se invoca 10.6 para añadir un valor de índice a la lista de campos como un número entero no negativo normalmente pequeño. El primer valor de índice será 0, y se incrementará en 1 para cada valor "identificación" distinto que se encuentre. Se invocará entonces la cláusula 22 para añadir la "identificación" a la lista de campos. La codificación completa del "valor de cadena" se añadirá entonces a un campo de bits alineados en octeto y se invocará 10.9 para añadirlo a una lista de campos, prefijada con la longitud como un número entero semiconstreñido. Esto completa los procedimientos para esta subcláusula.

27.5 La subregla "utilización de índice" se empleará en todas las apariciones subsiguientes de un determinado valor abstracto para "identificación". Cuando se emplea la subregla "utilización de índice", se invoca 10.6 para añadir un valor de índice a la lista de campos como un número entero no negativo normalmente pequeño. Este valor de índice

corresponderá con el que fue añadido previamente a la lista de campos para este determinado valor abstracto de "identificación". La codificación completa del "valor de cadena" se añadirá después a un campo de bits alineados en octeto y se invocará 10.9 para añadirlo a la lista de campos, prefijada con la longitud como un número entero semiconstreñido. Esto completa los procedimientos para esta subcláusula.

27.6 EXPLICACIÓN DIDÁCTICA: De este modo, para cualquier valor dado de "identificación" hay una codificación EP-A (relativamente ineficaz) con un valor de índice único y el valor de "identificación" completo, seguida por un número arbitrariamente grande de codificaciones EP-B (eficaces) vinculadas a la codificación EP-A por el valor de índice.

27.7 La codificación EP-A será la codificación PER del bit de fijación de índice/utilización de índice, seguido por el índice, seguido por el siguiente tipo de secuencia para la cual se especifica AUTOMATIC TAGS (véanse 22.6 y 26.3 de la Rec. UIT-T X.680 | ISO/CEI 8824-1):

```
SEQUENCE {
    identification              CHOICE {
        syntaxes                    SEQUENCE {
            abstract                    OBJECT IDENTIFIER,
            transfer                    OBJECT IDENTIFIER },
        syntax                      OBJECT IDENTIFIER,
        presentation-context-id     INTEGER,
        context-negotiation         SEQUENCE {
            presentation-context-id     INTEGER,
            transfer-syntax             OBJECT IDENTIFIER },
        transfer-syntax             OBJECT IDENTIFIER,
        fixed                       NULL },
    string-value                OCTET STRING }
```

27.8 El valor de "valor de cadena" será la codificación del valor de datos abstractos que utiliza la sintaxis de transferencia identificada, y el valor de todos los otros campos se codificará utilizando la misma sintaxis de transferencia que los otros valores que aparecen en el valor de sintaxis abstracta.

NOTA – Ambas alternativas de "valor de datos" en la sintaxis abstracta se codifican idénticamente – como una cadena de octetos – en la sintaxis de transferencia.

28 Identificadores de objeto para las sintaxis de transferencia

28.1 Las reglas de codificación especificadas en la presente Recomendación | Norma Internacional pueden ser referenciadas y aplicadas siempre que sea necesario especificar una representación de cadena de bits inequívoca para todos los valores de un solo tipo ASN.1.

28.2 Se asignan los siguientes valores de identificadores de objeto y de descriptor de objeto para identificar y describir las reglas de codificación especificadas en la presente Recomendación | Norma Internacional:

For BASIC-PER, ALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) aligned (0)}

"Packed encoding of a single ASN.1 type (basic aligned)"

For BASIC-PER, UNALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) unaligned (1)}

"Packed encoding of a single ASN.1 type (basic unaligned)"

For CANONICAL-PER, ALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) aligned (0)}

"Packed encoding of a single ASN.1 type (canonical aligned)"

For CANONICAL-PER, UNALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) unaligned (1)}

"Packed encoding of a single ASN.1 type (canonical unaligned)"

28.3 Cuando una norma de aplicación define una sintaxis abstracta como un conjunto de valores abstractos, cada uno de los cuales es un valor de algún tipo ASN.1 específicamente denominado que utiliza la notación ASN.1, se pueden utilizar los valores de identificador de objeto especificados en 28.2 con el nombre de la sintaxis abstracta para identificar las sintaxis de transferencia que resultan de la aplicación de las reglas de codificación especificadas en la presente Recomendación | Norma Internacional para el tipo ASN.1 específicamente denominado, utilizado para definir la sintaxis abstracta.

NOTA – En particular, estas identificaciones de las reglas de codificación pueden aparecer en el campo "nombre de sintaxis de transferencia" del protocolo de presentación (Rec. UIT-T X.226 | ISO/CEI 8823-1).

28.4 Los nombres especificados en 28.2 no se utilizarán con un nombre de sintaxis abstracta para identificar una sintaxis de transferencia si no se cumplen las condiciones de 28.3 para la definición de la sintaxis abstracta.

Anexo A

Ejemplo de codificaciones

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

Este anexo ilustra la utilización de las reglas de codificación compactada (PER, *packed encoding rules*) especificadas en la presente Recomendación | Norma Internacional mostrando representaciones en octetos de una ficha de personal (hipotética) que se define mediante la notación ASN.1.

A.1 Ficha que no utiliza construcciones de subtipo

A.1.1 Descripción ASN.1 de la estructura de registro

La estructura de la ficha de personal hipotética se describe formalmente a continuación utilizando la notación ASN.1 especificada en la Rec. UIT-T X.680 | ISO/CEI 8824-1 para la definición de tipos. Este ejemplo es idéntico al presentado en el Anexo A de la Rec. UIT-T X.690 | ISO/CEI 8825-1.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        name            Name,
        title           [0] VisibleString,
        number          EmployeeNumber,
        dateOfHire      [1] Date,
        nameOfSpouse    [2] Name,
        children        [3] IMPLICIT
            SEQUENCE OF ChildInformation DEFAULT {} }

ChildInformation ::= SET
        { name          Name,
          dateOfBirth   [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
    {givenName          VisibleString,
     initial            VisibleString,
     familyName         VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString  -- YYYYMMDD
```

A.1.2 Descripción ASN.1 de un valor de registro

El valor de la ficha de personal de John Smith se describe formalmente a continuación utilizando la notación ASN.1.

```
{ name {givenName "John",initial "P",familyName "Smith"},
  title         "Director",
  number        51,
  dateOfHire            "19710917",
  nameOfSpouse          {givenName "Mary",initial "T",familyName "Smith"},
  children
      {{name {givenName "Ralph",initial "T",familyName "Smith"},
            dateOfBirth "19571111"},
      {name {givenName "Susan",initial "B",familyName "Jones"},
            dateOfBirth "19590717"}}}
```

A.1.3 Representación ALIGNED PER de este valor de registro

Más adelante se muestra la representación del valor de registro indicado más arriba (después de aplicar la variante ALIGNED PER de las reglas de codificación compactada definidas en esta Recomendación | Norma Internacional). La codificación se presenta en hexadecimal, seguida de una descripción comentada de la codificación presentada en binario.

La longitud de esta codificación es 94 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante UNALIGNED de PER ocupa 84 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 136 octetos, y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 161 octetos.

A.1.3.1 Presentación hexadecimal

```
80044A6F 686E0150 05536D69 74680133 08446972 6563746F 72083139 37313039
3137044D 61727901 5405536D 69746802 0552616C 70680154 05536D69 74680831
39353731 31313105 53757361 6E014205 4A6F6E65 73083139 35393037 3137
```

A.1.3.2 Presentación binaria

Para facilitar la lectura de la presentación binaria de los datos se utilizan líneas en blanco para agrupar campos que guardan una relación lógica entre sí (típicamente, los pares longitud/valor); se emplea una nueva línea para delimitar campos; se utiliza un espacio para delimitar caracteres en una cadena de caracteres; una "x" representa un bit de relleno de valor cero que se emplea en diversas ocasiones para alinear campos sobre un límite de octeto.

1xxxxxxx	Bitmap bit = 1 indica que "children" está presente
00000100	Longitud de name.givenName = 4
01001010 01101111 01101000 01101110	name.givenName = "John"
00000001	Longitud de name.initial = 1
01010000	name.initial = "P"
00000101	Longitud de name.familyName = 5
01010011 01101101 01101001 01110100 01101000	name.familyName = "Smith"
00000001	Longitud de (employee) number = 1
00110011	(employee) number = 51
00001000	Longitud de title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	title = "Director"
00001000	Longitud de dateOfHire = 8
00110001 00111001 00110111 00110001 00110000 00111001 00110001 00111111	dateOfHire = "19590717"
00000100	Longitud de nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001	nameOfSpouse.givenName = "Mary"
00000001	Longitud de nameOfSpouse.initial = 1
01010100	nameOfSpouse.initial = "T"
00000101	Longitud de nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000	nameOfSpouse.familyName = "Smith"
00000010	Número de hijos
00000101	Longitud de children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000	children[0].givenName = "Ralph"
00000001	Longitud de children[0].initial = 1
01010100	children[0].initial = "T"
00000101	Longitud de children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000	children[0].familyName = "Smith"
00001000	Longitud de children[0].dateOfBirth = 8
00110001 00111001 00110101 00110111 00110001 00110001 00110001 00110001	children[0].dateOfBirth = "19571111"
00000101	Longitud de children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110	children[1].givenName = "Susan"
00000001	Longitud de children[1].initial = 1
01000010	children[1].initial = "B"
00000101	Longitud de children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011	children[1].familyName = "Jones"
00001000	Longitud de children[1].dateOfBirth = 8
00110001 00111001 00110101 00111001 00110000 00110111 00110001 00110111	children[1].dateOfBirth = "19590717"

A.1.4 Representación ALIGNED PER de este valor de registro

A continuación se muestra la representación del valor de récord dado más arriba (después de aplicar la variante UNALIGNED de las reglas de codificación compactada definidas en esta Recomendación | Norma Internacional). La codificación se presenta en hexadecimal, seguida de una descripción comentada de la codificación presentada en binario. Obsérvese que en la variante UNALIGNED no hay bits de relleno y que los caracteres se codifican con el menor número posible de bits.

La longitud de esta codificación es 84 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante ALIGNED de PER ocupa 94 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 136 octetos, y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 161 octetos.

A.1.4.1 Presentación hexadecimal

```
824ADFA3  700D005A  7B74F4D0  02661113  4F2CB8FA  6FE410C5  CB762C1C  B16E0937
0F2F2035  0169EDD3  D340102D  2C3B3868  01A80B4F  6E9E9A02  18B96ADD  8B162C41
69F5E787  700C2059  5BF765E6  10C5CB57  2C1BB16E
```

A.1.4.2 Presentación binaria

Para facilitar la lectura de la presentación binaria de los datos se utilizan líneas en blanco para agrupar campos que guardan una relación lógica entre sí (típicamente, los pares longitud/valor); se emplea una nueva línea para delimitar campos; se utiliza un espacio para delimitar caracteres en una cadena de caracteres; se utiliza un punto (.) para señalar límites de octeto; una "x" representa un bit de relleno de valor cero que se emplea en diversas ocasiones para alinear campos sobre un límite de octeto.

1	Bitmap bit = 1 indica que "children" está presente
0000010.0 1001010 .1101111 1.101000 11.01110	Longitud de name.givenName = 4 name.givenName = "John"
000.00001 101.0000	Longitud de name.initial = 1 name.initial = "P"
0000.0101 1010.011 11011.01 110100.1 1110100 .1101000	Longitud de name.familyName = 5 name.familyName = "Smith"
0.0000001 0.0110011	Longitud de (employee) number = 1 (employee) number = 51
0.0001000 1.000100 11.01001 111.0010 1100.101 11000.11 111010.0 1101111 .1110010	Longitud de title = 8 title = "Director"
0.0001000 0.110001 01.11001 011.0111 0110.001 01100.00 011100.1 0110001 .0111111	Longitud de dateOfHire = 8 dateOfHire = "19590717"
0.0000100 1.001101 11.00001 111.0010 1111.001	Longitud de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
00000.001 10101.00	Longitud de nameOfSpouse.initial = 1 nameOfSpouse.initial = "T"
000001.01 101001.1 1101101 .1101001 1.110100 11.01000	Longitud de nameOfSpouse.familyName = 5 nameOfSpouse.familyName = "Smith"
000.00010	Número de hijos
000.00101 101.0010 1100.001 11011.00 111000.0 1101000	Longitud de children[0].givenName = 5 children[0].givenName = "Ralph"
.00000001 .1010100	Longitud de children[0].initial = 1 children[0].initial = "T"
0.0000101 1.010011 11.01101 110.1001 1110.100 11010.00	Longitud de children[0].familyName = 5 children[0].familyName = "Smith"
000010.00 011000.1 0111001 .0110101 0.110111 01.10001 011.0001 0110.001 01100.01	Longitud de children[0].dateOfBirth = 8 children[0].dateOfBirth = "19571111"
000001.01 101001.1 1110101 .1110011 1.100001 11.01110	Longitud de children[1].givenName = 5 children[1].givenName = "Susan"

```
000.00001                                                    Longitud de children[1].initial = 1
100.0010                                                     children[1].initial = "B"

0000.0101                                                    Longitud de children[1].familyName = 5
1001.100  11011.11  110111.0  1100101  .1110011             children[1].familyName = "Jones"

0.0001000                                                    Longitud de children[1].dateOfBirth = 8
0.110001  01.11001  011.0101  0111.001  01100.00  011011.1  0110001  .0110111x children[1].dateOfBirth = "19590717"
```

A.2 Registro que utiliza constricciones de subtipo

Este ejemplo es similar al de la cláusula A.1, del que sólo se diferencia en que utiliza la notación de subtipo para imponer constricciones sobre algunos ítems.

A.2.1 Descripción ASN.1 de la estructura de registro

La estructura del registro de personal hipotético se describe formalmente a continuación utilizando la notación ASN.1 especificada en la Rec. UIT-T X.680 | ISO/CEI 8824-1 para la definición de tipos.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        name            Name,
        title           [0] VisibleString,
        number          EmployeeNumber,
        dateOfHire      [1] Date,
        nameOfSpouse    [2] Name,
        children        [3] IMPLICIT
            SEQUENCE OF ChildInformation  DEFAULT  {} }

ChildInformation ::= SET
        { name          Name,
          dateOfBirth   [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
    {givenName          NameString,
     initial            NameString (SIZE(1)),
     familyName         NameString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString (FROM("0".."9") ^ SIZE(8)) -- YYYYMMDD

NameString ::= VisibleString (FROM("a".."z" | "A".."Z" | "-.") ^ SIZE(1..64))
```

A.2.2 Descripción ASN.1 de un valor de registro

El valor del registro de personal de John Smith se describe formalmente a continuación utilizando la notación ASN.1.

```
{ name {givenName "John",initial "P",familyName "Smith"},
  title         "Director",
  number        51,
  dateOfHire            "19710917",
  nameOfSpouse          {givenName "Mary",initial "T",familyName "Smith"},
  children
      {{name {givenName "Ralph",initial "T",familyName "Smith"},
             dateOfBirth "19571111"},
       {name {givenName "Susan",initial "B",familyName "Jones"},
             dateOfBirth "19590717"}}}
```

A.2.3 Representación ALIGNED PER de este valor de registro

A continuación se muestra la representación del valor de registro dado más arriba (después de aplicar la variante ALIGNED de las reglas de codificación compactada definidas en esta Recomendación | Norma Internacional). La codificación se presenta en hexadecimal, seguida de una descripción comentada de la codificación presentada en binario. En la presentación binaria se utiliza una "x" para representar bits de relleno que se codifican con valor cero y se utilizan en diversas ocasiones para alinear los campos.

La longitud de esta codificación es 74 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante UNALIGNED de DE REGLAS DE CODIFICACIÓN COMPACTADA ocupa 61 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 136 octetos, y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 161 octetos.

A.2.3.1 Presentación hexadecimal

```
864A6F68 6E501053 6D697468 01330844 69726563 746F7219 7109170C 4D617279
5410536D 69746802 1052616C 70685410 536D6974 68195711 11105375 73616E42
104A6F6E 65731959 0717
```

A.2.3.2 Presentación binaria

Para facilitar la lectura de la presentación binaria de los datos se utilizan líneas en blanco para agrupar campos que guardan una relación lógica entre sí (típicamente, los pares longitud/valor); se emplea una nueva línea para delimitar campos; se utiliza un espacio para delimitar caracteres en una cadena de caracteres; una "x" representa un bit de relleno de valor cero que se emplea en diversas ocasiones para alinear campos sobre un límite de octeto.

1	Bitmap bit = 1 indica que "children" está presente
000011x	Longitud de name.givenName = 4
01001010 01101111 01101000 01101110	name.givenName = "John"
01010000	name.initial = "P"
000100xx	Longitud de name.familyName = 5
01010011 01101101 01101001 01110100 01101000	name.familyName = "Smith"
00000001	Longitud de (employee) number = 1
00110011	(employee) number = 51
00001000	Longitud de title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	title = "Director"
0001 1001 0111 0001 0000 1001 0001 0111	dateOfHire = "19590717"
000011xx	Longitud de nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001	nameOfSpouse.givenName = "Mary"
01010100	nameOfSpouse.initial = "T"
000100xx	Longitud de nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000	nameOfSpouse.familyName = "Smith"
00000010	Número de hijos
000100xx	Longitud de children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000	children[0].givenName = "Ralph"
01010100	children[0].initial = "T"
000100xx	Longitud de children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000	children[0].familyName = "Smith"
0001 1001 0101 0111 0001 0001 0001 0001	children[0].dateOfBirth = "19571111"
000100xx	Longitud de children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110	children[1].givenName = "Susan"
01000010	children[1].initial = "B"
000100xx	Longitud de children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011	children[1].familyName = "Jones"
0001 1001 0101 1001 0000 0111 0001 0111	children[1].dateOfBirth = "19590717"

A.2.4 Representación UNALIGNED PER de este valor de registro

A continuación se muestra la representación del valor de registro dado más arriba (después de aplicar la variante UNALIGNED de las reglas de codificación compactada definidas en esta Recomendación | Norma Internacional). La codificación se presenta en hexadecimal, seguida de una descripción comentada de la codificación presentada en binario. Obsérvese que en la variante UNALIGNED no hay bits de relleno y que los caracteres se codifican en el menor número posible de bits.

La longitud de esta codificación es 61 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante PER ocupa 74 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 136 octetos, y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 161 octetos.

A.2.4.1 Presentación hexadecimal

```
865D51D2  888A5125  F1809984  44D3CB2E  3E9BF90C  B8848B86  7396E8A8  8A5125F1
81089B93  D71AA229  4497C632  AE222222  985CE521  885D54C1  70CAC838  B8
```

A.2.4.2 Presentación binaria

Para facilitar la lectura de la presentación binaria de los datos se utilizan líneas en blanco para agrupar campos que guardan una relación lógica entre sí (típicamente, los pares longitud/valor); se emplea una nueva línea para delimitar campos; se utiliza un espacio para delimitar caracteres en una cadena de caracteres; se utiliza un punto (.) para marcar los límites de octeto; una "x" representa un bit de relleno de valor cero que se emplea en diversas ocasiones para alinear campos sobre un límite de octeto.

1	Bitmap bit = 1 indica que "children" está presente
000011 0.01011 101.010 10001.1 101001	Longitud de name.givenName = 4 name.givenName = "John"
0.10001	name.initial = "P"
000.100 01010.0 101000 1.00100 101.111 10001.1	Longitud de name.familyName = 5 name.familyName = "Smith"
0000000.1 0011001.1	Longitud de (employee) number = 1 (employee) number = 51
0000100.0 1000100 .1101001 1.110010 11.00101 110.0011 1110.100 11011.11 111001.0	Longitud de title = 8 title = "Director"
0001 100.1 0111 000.1 0000 100.1 0001 011.1	dateOfHire = "19590717"
000011 0.01110 011.100 10110.1 110100	Longitud de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
0.10101	nameOfSpouse.initial = "T"
000.100 01010.0 101000 1.00100 101.111 10001.1	Longitud de nameOfSpouse.familyName = 5 nameOfSpouse.familyName = "Smith"
0000001.0	Número de hijos
000100 0.10011 011.100 10011.1 101011 1.00011	Longitud de children[0].givenName = 5 children[0].givenName = "Ralph"
010.101	children[0].initial = "T"
00010.0 010100 1.01000 100.100 10111.1 100011	Longitud de children[0].familyName = 5 children[0].familyName = "Smith"
0.001 1001 0.101 0111 0.001 0001 0001 0001 0.001 1001 0.101 0111 0.001 0001 0.001 0001	children[0].dateOfBirth = "19571111"
0.00100 010.100 11000.0 101110 0.11100 101.001	Longitud de children[1].givenName = 5 children[1].givenName = "Susan"
00001.1	children[1].initial = "B"
000100 0.01011 101.010 10100.1 100000 1.01110	Longitud de children[1].familyName = 5 children[1].familyName = "Jones"
000.1 1001 010.1 1001 000.0 0111 000.1 0111xxx	children[1].dateOfBirth = "19590717"

A.3 Registro que utiliza constricciones de subtipo

A.3.1 Descripción ASN.1 de la estructura de registro

La estructura del registro de personal hipotético se describe formalmente a continuación utilizando la notación ASN.1 especificada en la Rec. UIT-T X.680 | ISO/CEI 8824-1 para la definición de tipos que no son extensibles, y la especificada en la Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1 para la definición de tipos extensibles.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        name        Name,
        title       [0] VisibleString,
        number      EmployeeNumber,
```

```
            dateOfHire      [1] Date,
            nameOfSpouse    [2] Name,
            children        [3] IMPLICIT
                SEQUENCE (SIZE(2, ...)) OF ChildInformation  OPTIONAL  {},
        .. }

ChildInformation ::= SET
            { name          Name,
              dateOfBirth   [0] Date,
              ...,
              sex           [1] IMPLICIT ENUMERATED {male(1), female(2), unknown(3)} OPTIONAL
            }

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
            {givenName      NameString,
             initial        NameString (SIZE(1)),
             familyName     NameString,
             ...}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER (0..9999, ...)

Date ::= [APPLICATION 3] IMPLICIT VisibleString (FROM("0".."9") ^ SIZE(8, ..., 9..20)) -- YYYYMMDD

NameString ::= VisibleString (FROM("a".."z" | "A".."Z" | "-.") ^ SIZE(1..64, ...))
```

A.3.2 Descripción ASN.1 de un valor de registro

El valor del registro de personal de John Smith se describe formalmente a continuación utilizando la notación ASN.1.

```
{ name {givenName "John",initial "P",familyName "Smith"},
  title         "Director",
  number        51,
  dateOfHire            "19710917",
  nameOfSpouse          {givenName "Mary",initial "T",familyName "Smith"},
  children
      {{name {givenName "Ralph",initial "T",familyName "Smith"},
             dateOfBirth "19571111"},
       {name {givenName "Susan",initial "B",familyName "Jones"},
             dateOfBirth "19590717", sex female}}}
```

A.3.3 Representación ALIGNED PER de este valor de registro

A continuación se muestra la representación del valor de registro dado más arriba (después de aplicar la variante ALIGNED de las reglas de codificación compactada definidas en esta Recomendación | Norma Internacional). La codificación se presenta en hexadecimal, seguida de una descripción comentada de la codificación presentada en binario. En la presentación binaria se utiliza una "x" para representar bits de relleno codificados con el valor cero y se emplean para alinear los campos en diversas ocasiones.

La longitud de esta codificación es 83 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante UNALIGNED de PER ocupa 65 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 139 octetos, y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 164 octetos.

A.3.3.1 Presentación hexadecimal

```
40C04A6F  686E5008  536D6974  68000033  08446972  6563746F  72001971  0917034D
61727954  08536D69  74680100  52616C70  68540853  6D697468  00195711  11820053
7573616E  42084A6F  6E657300  19590717  010140
```

A.3.3.2 Presentación binaria

Para facilitar la lectura de la presentación binaria de los datos se utilizan líneas en blanco para agrupar campos que guardan una relación lógica entre sí (típicamente, los pares longitud/valor); se emplea una nueva línea para delimitar campos; se utiliza un espacio para delimitar caracteres en una cadena de caracteres; una "x" representa un bit de relleno de valor cero que se emplea en diversas ocasiones para alinear campos sobre un límite de octeto.

```
0                                       No hay valores de extensión presentes en
                                        PersonnelRecord
1                                       Bitmap bit = 1 indica que "children" está presente
```

0	No hay valores de extensión presentes en "name"
0	La longitud está dentro de la gama de la raíz de extensión
0000 11xxxxxx	Longitud de name.givenName = 4
01001010 01101111 01101000 01101110	name.givenName = "John"
01010000	name.initial = "P"
0	La longitud está dentro de la gama de la raíz de extensión
000100x	Longitud de name.familyName = 5
01010011 01101101 01101001 01110100 01101000	name.familyName = "Smith"
0xxxxxxx	La longitud está dentro de la gama de la raíz de extensión
00000000 00110011	(employee) number = 51
00001000	Longitud de title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	title = "Director"
0xxxxxxx	La longitud está dentro de la gama de la raíz de extensión
0001 1001 0111 0001 0000 1001 0001 0111	dateOfHire = "19590717"
0	No hay valores de extensión presentes en nameOfSpouse
0	La longitud está dentro de la gama de la raíz de extensión
000011	Longitud de nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001	nameOfSpouse.givenName = "Mary"
01010100	nameOfSpouse.initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
000100x	Longitud de nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000	nameOfSpouse.familyName = "Smith"
0	El número de "children" está dentro de la gama de la raíz de extensión
0	No hay valores de extensión presentes en children[0]
0	No hay valores de extensión presentes en children[0].name
0	La longitud está dentro de la gama de la raíz de extensión
000100xx xxxx	Longitud de children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000	children[0].givenName = "Ralph"
01010100	children[0].initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
000100x	Longitud de children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000	children[0].familyName = "Smith"
0xxxxxxx	La longitud está dentro de la gama de la raíz de extensión
0001 1001 0101 0111 0001 0001 0001 0001	children[0].dateOfBirth = "19571111"
1	Valor(es) de extensión presente(s) en children[1]
0	No hay valores de extensión presentes en children[0].name
0	La longitud está dentro de la gama de la raíz de extensión
00010 0xxxxxxx	Longitud de children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110	children[1].givenName = "Susan"
01000010	children[1].initial = "B"

0	La longitud está dentro de la gama de la raíz de extensión
000100x	Longitud de children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011	children[1].familyName = "Jones"
0xxxxxxx	La longitud está dentro de la gama de la raíz de extensión
0001 1001 0101 1001 0000 0111 0001 0111	children[1].dateOfBirth = "19590717"
0000000	Longitud del bitmap de adición de extensión para children[1] = 1
1	Indica valor de extensión para "sex" está presente
00000001	Longitud de la codificación completa de "sex"
01xxxxxx	Codificación completa de "sex" = female

A.3.4 Representación UNALIGNED PER de este valor de registro

A continuación se muestra la representación del valor de registro dado más arriba (después de aplicar la variante UNALIGNED de las reglas de codificación compactada definidas en esta Recomendación | Norma Internacional). La codificación se presenta en hexadecimal, seguida de una descripción comentada de la codificación presentada en binario. Obsérvese que en la variante UNALIGNED no hay bits de relleno y que los caracteres se codifican en el menor número posible de bits.

La longitud de esta codificación es 65 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante UNALIGNED de PER ocupa 83 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 139 octetos, y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 164 octetos.

A.3.4.1 Presentación hexadecimal

```
40CBAA3A 5108A512 5F180330 889A7965 C7D37F20 CB8848B8 19CE5BA2 A114A24B
E3011372 7AE35422 94497C61 95711118 22985CE5 21842EAA 60B832B2 0E2E0202
80
```

A.3.4.2 Presentación binaria

Para facilitar la lectura de la presentación binaria de los datos se utilizan líneas en blanco para agrupar campos que guardan una relación lógica entre sí (típicamente, los pares longitud/valor); se emplea una nueva línea para delimitar campos; se utiliza un espacio para delimitar caracteres en una cadena de caracteres; se utiliza un punto (.) para marcar los límites de octeto; una "x" representa un bit de relleno de valor cero que se emplea para alinear campos sobre un límite de octeto.

0	No hay valores de extensión presentes en PersonnelRecord
1	Bitmap bit = 1 indica que "children" está presente
0	No hay valores de extensión presentes en "name"
0	La longitud está dentro de la gama de la raíz de extensión
0000.11	Longitud de name.givenName = 4
001011 .101010 10.0011 1010.01	name.givenName = "John"
010001	name.initial = "P"
.0	La longitud está dentro de la gama de la raíz de extensión
000100	Longitud de name.familyName = 5
0.10100 101.000 10010.0 101111 1.00011	name.familyName = "Smith"
0	La longitud está dentro de la gama de la raíz de extensión
00.00000011.0011	(employee) number = 51
0000.1000	Longitud de title = 8
1000.100 11010.01 111001.0 1100101 1100011 1.110100 11.01111 111.0010	title = "Director"
0	La longitud está dentro de la gama de la raíz de extensión
000.1 1001 011.1 0001 000.0 1001 000.1 0111	dateOfHire = "19590717"

0	No hay valores de extensión presentes en nameOfSpouse
0	La longitud está dentro de la gama de la raíz de extensión
0.00011	Longitud de nameOfSpouse.givenName = 4
001.110 01110.0 101101 1.10100	nameOfSpouse.givenName = "Mary"
010.101	nameOfSpouse.initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
0001.00	Longitud de nameOfSpouse.familyName = 5
010100 .101000 10.0100 1011.11 100011	nameOfSpouse.familyName = "Smith"
.0	El número de "children" está dentro de la gama de la raíz de extensión
0	No hay valores de extensión presentes en children[0]
0	No hay valores de extensión presentes en children[0].name
0	La longitud está dentro de la gama de la raíz de extensión
0001.00	Longitud de children[0].givenName = 5
010011 .011100 10.0111 1010.11 100011	children[0].givenName = "Ralph"
.010101	children[0].initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
0.00100	Longitud de children[0].familyName = 5
010.100 10100.0 100100 1.01111 100.011	children[0].familyName = "Smith"
0	La longitud está dentro de la gama de la raíz de extensión
0001 .1001 0101 .0111 0001 .0001 0001 .0001	children[0].dateOfBirth = "19571111"
1	Valor(es) de extensión presente(s) en children[1]
0	No hay valores de extensión presentes en children[0].name
0	La longitud está dentro de la gama de la raíz de extensión
0.00100	Longitud de children[1].givenName = 5
010.100 11000.0 101110 0.11100 101.001	children[1].givenName = "Susan"
00001.1	children[1].initial = "B"
0	La longitud está dentro de la gama de la raíz de extensión
000100	Longitud de children[1].familyName = 5
.001011 10.1010 1010.01 100000 .101110	children[1].familyName = "Jones"
0	La longitud está dentro de la gama de la raíz de extensión
0.001 1001 0.101 1001 0.000 0111 0.001 0111	children[1].dateOfBirth = "19590717"
0.000000	Longitud del bitmap de adición de extensión para children[1] = 1
1	Indica valor de extensión para "sex" está presente
0.0000001	Longitud de la codificación completa de "sex"
0.1xxxxxx	Codificación completa de "sex" = female
x	Bit de relleno para crear la codificación completa de PersonnelRecord

Anexo B

Observaciones sobre la combinación de constricciones visibles a las reglas de codificación compactada (PER)

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

Ciertas propiedades pueden observarse cuando se combinan elementos de subtipo, cada uno de los cuales, individualmente, puede ser visible a la PER. A continuación se describen esas propiedades por medio de ejemplos.

B.1 La constricción de tamaño efectiva para

```
A ::= IA5String (SIZE(1..4) | SIZE(9..10))
```

es

```
A ::= IA5String (SIZE(1..4 | 9..10))
```

B.2 Cuando una constricción de alfabeto permitido (PermittedAlphabet) se combina en la misma especificación de subtipo con otras constricciones de alfabeto permitido, no hay una constricción de alfabeto permitido efectiva que contenga un número de caracteres menor que la totalidad de los caracteres que aparecen en el tipo no constreñido, a menos que haya una especificación única de alfabeto permitido que sea un superconjunto de todas las demás especificaciones de alfabeto permitido en esa especificación de subtipo. Además, si en esa especificación de constricción se incluyen constricciones de tamaño, la especificación de alfabeto permitido que constituya dicho superconjunto deberá tener una constricción de tamaño efectiva, combinada con ella por medio de la operación INTERSECTION, de tal modo que la constricción de tamaño efectiva sea un superconjunto de todas las demás constricciones de tamaño impuestas al tipo. Por ejemplo,

```
B ::= IA5String ( FROM ("AB") ^ SIZE(1..2) |
                  FROM ("DE") ^ SIZE(3) |
                  FROM ("ABCDE") ^ (SIZE(1..5) )
```

tiene una constricción de tamaño efectiva y una constricción de alfabeto permitido efectiva de

```
B ::= IA5String (FROM ("ABCDE") ^ SIZE(1..5))
```

porque este es un superconjunto de la anterior expresión, más compleja, y es por tanto visible a la PER. Por otra parte, en lo sucesivo, la constricción de alfabeto permitido (PermittedAlphabet) será el juego completo de caracteres autorizados para IA5String, pues no hay una constricción de alfabeto permitido equivalente única que pueda escribirse. Por consiguiente, la constricción en lo siguiente no es visible a la PER:

```
C ::= IA5String (FROM("AB") | FROM("CD"))        -- Esto no es equivalente a (FROM("ABCD"))
```

B.3 Las constricciones de tamaño pueden combinarse libremente mientras no intervenga PermittedAlphabet. Por ejemplo, con

```
E ::= IA5String (SIZE(1..4) | SIZE(5..10) ^ FROM("ABCD") | SIZE(6..10))
```

no hay constricción de tamaño visible a PER (ya que el tamaño 5 no puede aparecer para todos los caracteres posibles), mientras que si tuviéramos

```
E ::= IA5String (SIZE(1..4) | SIZE(6..10) ^ FROM("ABCD") | SIZE(6..10))
```

habría una constricción de tamaño visible a PER de SIZE(1..4 | 6..10) aunque la constricción PermittedAlphabet efectiva fuera el conjunto de todos los caracteres IA5String. FROM("ABCD") en este caso no forma una constricción visible a PER porque esta constricción no se aplica a todos los valores posibles de E (por ejemplo, si la longitud de la cadena es 1, el carácter no está limitado a uno de "ABCD").

Anexo C

Soporte de los algoritmos de las PER

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

Una norma de aplicación, o un perfil normalizado internacional, puede especificar cuáles de las reglas de codificación compactada han de ser sustentadas y las sintaxis de transferencia correspondientes que se han de ofrecer o aceptar en la negociación.

Cuando es necesario utilizar codificaciones con retransmisión segura y/o canónicas dentro de EMBEDDED PDV (o EXTERNAL) o CHARACTER STRING, esto se debe indicar claramente.

El siguiente texto proporciona orientaciones que se pueden utilizar para elaborar texto normativo.

C.1 Una codificación canónica está destinada a ser utilizada cuando se están aplicando características de seguridad a la codificación (véase el Anexo D a la Rec. UIT-T X.690 | ISO/CEI 8825-1). La utilización de CANONICAL-PER (PER canónica) puede entrañar costes adicionales importantes de utilización de la CPU cuando el valor que se ha de codificar incluye un tipo set-of (conjunto de), y generalmente no se recomienda para protocolos a menos que se requieran características de seguridad.

C.2 Donde un valor de sintaxis abstracta contiene un material insertado que se ha codificado mediante una sintaxis de transferencia o una sintaxis abstracta diferentes de la asociada con el valor de sintaxis abstracta, se recomienda insistentemente que el material insertado se codifique de una manera segura para la retransmisión. Se requerirá una regla de codificación canónica si las características de seguridad son importantes. En este contexto se debe prestar una atención particular al nivel de ISO/CEI 10646-1 que se va a utilizar para el tipo BMPString o UniversalString, ya que sólo el nivel 1 de implementación de ISO/CEI 10646-1 garantiza la producción de una codificación canónica.

C.3 Cuando se establece un contexto de presentación para flujo de datos unidireccional, se recomienda decididamente que los datos se codifiquen de una manera segura para la retransmisión.

C.4 Cuando se establece un contexto de presentación para flujo de datos bidireccional, puede haber ventajas importantes de flexibilidad y economía si se emplea BASIC-PER (PER básica).

C.5 Se recomienda decididamente que todas las implementaciones que sustentan la decodificación de cualquier sintaxis de transferencia de la variante PER ALIGNED sustenten la decodificación de la variante BASIC-PER ALIGNED (y por tanto de la variante CANONICAL-PER ALIGNED) y que acepten contextos de presentación identificados con cualquiera de estas dos reglas de codificación a condición de que se establezcan los contextos para la recepción de datos por esa implementación. Se aplica lo mismo para la variante UNALIGNED.

C.6 Para facilitar el interfuncionamiento, se recomienda que todas las implementaciones de PER sustenten la variante ALIGNED y la variante UNALIGNED (la adición de complejidad de la implementación es pequeña). Las que se ofrecen en un caso de comunicación (cualquiera de las dos o ambas) es un asunto de gestión local, y la que se acepta si se ofrecen ambas es también un asunto de gestión local. Si sólo se ofrece una, ésta debe ser aceptada.

C.7 La aceptación de estas recomendaciones es particularmente importante para los vendedores de herramientas ("tools") para uso general. Cuando una implementación es específica de alguna aplicación particular, puede ser totalmente aceptable la sustentación de una sola sintaxis de transferencia PER (quizás especificada por el diseñador de esa aplicación).

Anexo D

Soporte de las reglas de extensibilidad de la ASN.1

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

D.1 Estas reglas de codificación compactada (*packed encoding rules*) dependen de la definición completa del tipo a que se aplican. En general, si en la definición de tipo se introducen cambios que no sean puramente sintácticos, la codificación para todos los valores que utilizan esa parte de la especificación será afectada. En particular, la adición de ulteriores componentes facultativos a una secuencia, la conversión de un componente en un CHOICE de ese componente y algún otro tipo, y la mitigación o agravación de constricciones sobre algunos componentes, probablemente cambien la codificación de valores del tipo en cuestión.

D.2 Sin embargo, estas reglas de codificación se han diseñado para asegurar que se cumplan los requisitos de las reglas de codificación especificados en el modelo ASN.1 de extensión de tipo (véase la Rec. UIT-T X.680/Enm. 1 | ISO/CEI 8824-1/Enm. 1).

D.3 Cuando un tipo no forma parte de una secuencia de extensión (no hay marcador de extensión presente), se aplica el texto que precede a este anexo: PER no proporciona soporte para la extensibilidad de ese tipo. Cuando un tipo secuencia o conjunto tenga un marcador de extensión pero no haya adiciones de extensión, hay una tara de un bit (que puede convertirse en un octeto debido al relleno en las variantes ALIGNED), en comparación con el mismo tipo sin marcador de extensión. Cuando hay adiciones presentes en el tipo y dichas adiciones se transmiten efectivamente en una determinada comunicación, hay una tara suplementaria de aproximadamente un octeto, más un campo de longitud adicional para cada adición de extensión que se transmita, en comparación con el mismo tipo con el marcador de extensión suprimido.

D.4 Es importante observar que tanto la adición como la supresión de un marcador de extensión cambia los bits en la línea, lo que generalmente requerirá un cambio del número de la versión del protocolo.

D.5 La inclusión de un marcador de extensión en un conjunto de objetos de información o la adición o supresión de especificaciones de excepciones no entrañan cambios en la codificación; sin embargo, es evidente que pueden representar cambios en el comportamiento requerido de una implementación, y podrían, de todas formas, requerir un cambio del número de la versión del protocolo.

Anexo E

Anexo explicativo sobre la concatenación de codificaciones PER

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

E.1 Las codificaciones PER son autodelimitadoras, dado el conocimiento de las reglas de codificación y el tipo de la codificación. Las codificaciones completas para las variantes ALIGNED y UNALIGNED son siempre un múltiplo de ocho bits.

E.2 A los efectos de transportar codificaciones PER en el protocolo de capa de presentación de OSI, las codificaciones de las variantes ALIGNED y UNALIGNED se pueden concatenar en la opción de cadena de octetos.

Anexo F

Asignación de valores de identificador de objeto

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

En la presente Recomendación | Norma Internacional se han asignado los siguientes valores de identificador de objeto y de descriptor de objeto:

For BASIC-PER, ALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) aligned (0)}

"Packed encoding of a single ASN.1 type (basic aligned)"

For BASIC-PER, UNALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) unaligned (1)}

"Packed encoding of a single ASN.1 type (basic unaligned)"

For CANONICAL-PER, ALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) aligned (0)}

"Packed encoding of a single ASN.1 type (canonical aligned)"

For CANONICAL-PER, UNALIGNED variant:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) unaligned (1)}

"Packed encoding of a single ASN.1 type (canonical unaligned)"