

# UIT-T

SECTOR DE NORMALIZACIÓN  
DE LAS TELECOMUNICACIONES  
DE LA UIT

# X.691

(07/2002)

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

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

RECOMENDACIONES UIT-T DE LA SERIE X  
**REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD**

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

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

**Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación compactada**

**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 Rec. UIT-T X.690 | ISO/CEI 8825-1).

**Orígenes**

La Recomendación UIT-T X.691 fue aprobada el 14 de julio de 2002 por la Comisión de Estudio 17 (2001-2004) del UIT-T por el procedimiento de la Recomendación UIT-T A.8. Se publica también un texto idéntico como Norma Internacional ISO/CEI 8825-2.

## PREFACIO

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

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

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

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

## NOTA

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

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

## PROPIEDAD INTELECTUAL

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

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

© UIT 2006

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

## ÍNDICE

*Página*

1	Alcance .....	1
2	Referencias normativas .....	1
2.1	Recomendaciones   Normas Internacionales idénticas .....	1
2.2	Pares de Recomendaciones   Normas internacionales de contenido técnico equivalente .....	1
2.3	Referencias adicionales .....	1
3	Definiciones .....	2
3.1	Especificación de la notación básica .....	2
3.2	Especificación de objetos de información .....	2
3.3	Especificación de restricciones .....	2
3.4	Parametrización de la especificación de ASN.1 .....	2
3.5	Reglas de codificación básica .....	2
3.6	Definiciones adicionales .....	2
4	Abreviaturas .....	5
5	Notación .....	5
6	Convenios .....	5
7	Reglas de codificación definidas en la presente Recomendación   Norma Internacional .....	5
8	Conformidad .....	6
9	Método de codificación utilizado para PER .....	7
9.1	Utilización de la notación de tipos .....	7
9.2	Utilización de rótulos para proporcionar un orden canónico .....	7
9.3	Restricciones visibles a PER .....	7
9.4	Modelo de tipos y valores utilizados para la codificación .....	9
9.5	Estructura de una codificación .....	9
9.6	Tipos que se han de codificar .....	10
10	Procedimientos de codificación .....	10
10.1	Producción de la codificación completa .....	10
10.2	Campos de tipo abierto .....	11
10.3	Codificación como un entero binario no negativo .....	11
10.4	Codificación como un entero binario complemento de dos .....	12
10.5	Codificación de un número entero restringido .....	12
10.6	Codificación de un número entero no negativo normalmente pequeño .....	13
10.7	Codificación de un número entero semirrestringido .....	13
10.8	Codificación de un número entero no restringido .....	14
10.9	Reglas generales para codificar un determinante de longitud .....	14
11	Codificación del tipo booleano .....	17
12	Codificación del tipo entero .....	17
13	Codificación del tipo enumerado .....	18
14	Codificación del tipo real .....	18
15	Codificación del tipo cadena de bits .....	18
16	Codificación del tipo cadena de octetos .....	19
17	Codificación del tipo nulo .....	20
18	Codificación del tipo secuencia .....	20
19	Codificación del tipo secuencia de .....	21
20	Codificación del tipo conjunto .....	21
21	Codificación del tipo conjunto de .....	22
22	Codificación del tipo elección .....	22
23	Codificación del tipo identificador de objeto .....	23

	<i>Página</i>
24 Codificación del tipo identificador de objeto relativo .....	23
25 Codificación del tipo PDV INCRUSTADO .....	23
26 Codificación de un valor del tipo externo .....	24
27 Codificación de los tipos cadenas de caracteres restringidas .....	25
28 Codificación del tipo cadena de caracteres no restringido .....	26
29 Identificadores de objeto para las sintaxis de transferencia .....	27
Anexo A – Ejemplos de codificaciones .....	28
A.1 Ficha sin restricciones de subtipo .....	28
A.2 Ficha con restricciones de subtipo .....	31
A.3 Registro que utiliza marcadores de extensión .....	33
A.4 Ficha con grupos de adición de extensión .....	38
Anexo B – Combinación de restricciones visibles a las reglas de codificación compactadas (PER) y restricciones no visibles .....	40
B.1 Consideraciones generales .....	40
B.2 Extensibilidad y visibilidad de las restricciones en PER .....	41
B.3 Ejemplos .....	43
Anexo C – Soporte de los algoritmos de las PER .....	46
Anexo D – Soporte de las reglas de extensibilidad de la ASN.1 .....	47
Anexo E – Anexo explicativo sobre la concatenación de codificaciones PER .....	48
Anexo F – Asignación de valores de identificador de objeto .....	49

## **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, 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**

Esta Recomendación | Norma Internacional especifica un conjunto de reglas de codificación compactada, 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 esta 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 | ISO/CEI 8824-1.

## **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 las Normas citadas a continuación. Los miembros de la CEI y de la ISO mantienen registros de las Normas Internacionales actualmente vigentes. La Oficina de Normalización de las Telecomunicaciones de la UIT mantiene una lista de las Recomendaciones UIT-T actualmente vigentes.

### **2.1 Recomendaciones | Normas Internacionales idénticas**

- Recomendación UIT-T X.680 (2002) | ISO/CEI 8824-1:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica.*
- Recomendación UIT-T X.681 (2002) | ISO/CEI 8824-2:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de objetos de información.*
- Recomendación UIT-T X.682 (2002) | ISO/CEI 8824-3:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones.*
- Recomendación UIT-T X.683 (2002) | ISO/CEI 8824-4:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Parametrización de las especificaciones de notación de sintaxis abstracta uno.*
- Recomendación UIT-T X.690 (2002) | ISO/CEI 8825-1:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación básica, de las reglas de codificación canónica y de las reglas de codificación distinguida.*

### **2.2 Pares de Recomendaciones | Normas internacionales de contenido técnico equivalente**

### **2.3 Referencias adicionales**

- ISO/CEI 646:1991, *Information technology – ISO 7-bit coded character set for information interchange.*
- ISO/CEI 2022:1994, *Information technology – Character code structure and extension techniques.*
- ISO 2375:1985, *Data processing – Procedure for registration of escape sequences.*

## ISO/CEI 8825-2:2002 (S)

- ISO 6093:1985, *Information processing – Representation of numerical values in character strings for information interchange.*
- *ISO International Register of Coded Character Sets to be Used with Escape Sequences.*
- ISO/CEI 10646-1:2000, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*

### 3 Definiciones

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

#### 3.1 Especificación de la notación básica

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

#### 3.2 Especificación de objetos de información

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

#### 3.3 Especificación de restricciones

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

- restricción de relación de componentes;
- restricción de tabla.

#### 3.4 Parametrización de la especificación de ASN.1

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

- restricción variable.

#### 3.5 Reglas de codificación básica

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

- conformidad dinámica;
- conformidad estática;
- valor de datos;
- codificación (de un valor de datos);
- emisor;
- receptor.

#### 3.6 Definiciones adicionales

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

**3.6.1 codificación de entero binario con complemento a dos:** Codificación de un número entero en un campo de bits (alineado a octeto en la variante alineado (ALIGNED)) de una longitud especificada, o en el número mínimo de octetos que acomodarán ese número entero codificado como un entero formado por complemento a dos, que permite representar números enteros que son iguales, mayores o menores que cero, como se especifica en 10.4.

NOTA 1 – El valor de un número binario complemento a 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 a 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.

NOTA 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.

**3.6.2 valor de sintaxis abstracta:** Valor de una sintaxis abstracta (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**.

**3.6.3 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.

NOTA – Si este término se utiliza seguido de "alineado a octeto en la variante ALIGNED", significa que el campo de bits ha de comenzar necesariamente en el límite de un octeto en la codificación completa en la variante alineada de las PER.

**3.6.4 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.

**3.6.5 tipo compuesto:** Una cadena de caracteres externa o no condicionada que constituye un conjunto, secuencia, conjunto parcial, secuencia parcial, selección o valor de datos de presentación insertado.

**3.6.6 valor compuesto:** Valor de un tipo compuesto.

**3.6.7 número entero restringido:** Número entero que tiene restricciones visibles a las reglas de codificación compactada para 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 restringidos 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 restricciones 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 restricciones visibles a las reglas de codificación compactada a un número máximo de componentes, el valor de un tipo entero que ha sido restringido por restricciones 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.

**3.6.8 restricción de tamaño efectiva (para un tipo de cadena restringido):** Una sola restricción 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 restringido.

NOTA 1 – Por ejemplo, lo siguiente tiene una restricción de tamaño efectiva:

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

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

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

mientras que lo siguiente no tiene restricción 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"))
```

NOTA 2 – La restricción de tamaño efectiva se utiliza únicamente para determinar la codificación en cuanto a la longitud de la cadena.

**3.6.9 restricción efectiva de alfabeto permitido (para un tipo de cadena de caracteres restringido):** Restricción 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.

NOTA 1 – Por ejemplo, en:

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

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

"Ax" tiene una restricción efectiva de alfabeto permitido de "ABCD". Bx tiene una restricción de alfabeto permitido efectiva que consiste en todo el alfabeto IA5String, porque no hay restricción de alfabeto permitido más pequeña que se aplique a todos los valores de Bx.

NOTA 2 – Se utiliza la restricción efectiva de alfabeto permitido solamente para determinar la codificación de caracteres.

**3.6.10 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".

**3.6.11 extensible para codificación de reglas de codificación compactada:** Propiedad de un tipo que requiere que las reglas de codificación compactada identifiquen la codificación de un valor como la de un valor raíz o como la de una adición de extensión.

NOTA – La codificación de los valores raíz normalmente es más eficiente que la codificación de adiciones de extensión.

**3.6.12 lista de campos:** Conjunto ordenado de campos de bits que se produce como resultado de la aplicación de estas reglas de codificación a componentes de un valor abstracto.

**3.6.13 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.

**3.6.14 tipo de longitud fija:** Un tipo para el que se puede determinar el valor del determinante de longitud más externo en la correspondiente codificación (utilizando los mecanismos especificados en la presente Recomendación | Norma Internacional) a partir de la notación de tipo (después de la aplicación de restricciones visibles a las reglas de codificación compactada solamente) y es el mismo para todos los posibles valores del tipo.

**3.6.15 valor fijo:** Valor para el que se puede determinar (utilizando los mecanismos especificados en la presente Recomendación | Norma Internacional) que es el único valor permitido (después de la aplicación de restricciones visibles a las reglas de codificación compactada solamente) del tipo que lo gobierna.

**3.6.16 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**.

**3.6.17 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.

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

**3.6.19 longitud normalmente pequeña:** Codificación de longitud que representa valores de una longitud no limitada, pero siendo más probable tener longitudes pequeñas que grandes.

**3.6.20 codificación de entero binario no negativo:** Codificación de un número entero restringido o semirrestringido en un campo de bits (alineado a octeto en la variante ALIGNED) de una longitud especificada, o en un campo de bits alineados a 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.

**3.6.21 tipo más extremo:** Un tipo ASN.1 cuya codificación se incluye en un portador no ASN.1 o como valor de otras construcciones ASN.1 (véase 10.1.1).

NOTA – Las codificaciones PER de un tipo más externo son siempre un entero múltiplo de 8 bits.

**3.6.22 restricción visible a las reglas de codificación compactada:** Caso de utilización de la notación de restricciones de notación de sintaxis abstracta uno que afecta a la codificación de reglas de codificación compactada de un valor.

**3.6.23 codificación de retransmisión segura:** Codificación completa de un valor en sintaxis abstracta que puede ser decodificado (incluidas cualesquiera codificaciones insertadas) sin conocer el entorno en el cual se realizó la codificación.

**3.6.24 número entero semirrestringido:** Número entero que tiene restricciones visibles a las reglas de codificación compactada para rebasar o ser igual a algún valor "lb" con el valor "lb" como un valor permitido, y que no es un número entero restringido.

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

**3.6.25 tipo simple:** Tipo que no es un tipo compuesto.

**3.6.26 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 conjuntos 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 9.3.5, la restricción sobre `Foo` no es visible a las reglas de codificación compactada porque `Bar` tiene más restricciones 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)
```

**3.6.27 número entero no restringido:** Número entero que no está restringido por restricciones visibles a las reglas de codificación compactada.

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

## 4 Abreviaturas

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

ASN.1	Notación de sintaxis abstracta uno ( <i>abstract syntax notation one</i> )
BER	Reglas de codificación básica de ASN.1 ( <i>basic encoding rules of ASN.1</i> )
CER	Reglas de codificación canónica de ASN.1 ( <i>canonical encoding rules of ASN.1</i> )
DER	Reglas de codificación distinguida de ASN.1 ( <i>distinguished encoding rules</i> )
PER	Reglas de codificación compactada de ASN.1 ( <i>packed encoding rules of ASN.1</i> )
16K	16384
32K	32768
48K	49152
64K	65536

## 5 Notación

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

## 6 Convenios

**6.1** Esta 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 esta 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 esta 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 esta Recomendación | Norma Internacional.

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

**7.1** Esta 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, *packed encoding rules of ASN.1*)]. 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 **EXTERNO (EXTERNAL)**, **PDV INSERTADO (EMBEDDED PDV)** y **CADENA DE CARACTERES (CHARACTER STRING)** tengan una restricción de transporte de identificadores de contextos de presentación OSI.

**7.4** El algoritmo de regla de codificación más general especificado en esta 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 esta 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.

NOTA 1 – La codificación CANONICAL-PER produce codificaciones canónicas que tienen aplicaciones cuando hay que aplicar autenticadores a valores abstractos.

NOTA 2 – Cualquier implementación conforme a CANONICAL-PER para la codificación es conforme 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 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** Hay dos variantes de BASIC-PER y CANONICAL-PER: la variante **ALINEADA (ALIGNED)** y la variante **NO ALINEADA (UNALIGNED)**. 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 **CADENA DE OCTETOS (OCTET STRING)**, 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 **OCTET STRING**.

**7.10** Las reglas de esta 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 de la cláusula 9 en adelante.

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

NOTA – El anexo C 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 esta 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 esta 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.

NOTA 1 – En general no hay codificaciones alternativas definidas para la BASIC-PER explícitamente indicada en esta 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.

NOTA 2 – Cuando se utiliza CANONICAL-PER para proporcionar una codificación canónica, se recomienda que cualquier valor criptado 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).

## 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 un tipo es o no extensible para una codificación PER.

### 9.2 Utilización de rótulos para proporcionar un orden canónico

Esta 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 se determina según el rótulo más extremo de cada componente, como se especifica en 8.6 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

### 9.3 Restricciones visibles a PER

NOTA – El hecho de que algunas restricciones 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 restricciones en el tratamiento de errores detectados durante la codificación, ni supone que los valores que violan estas restricciones pueden ser transmitidos por un emisor conforme. No obstante, en esta Recomendación | Norma Internacional no se utilizan estas restricciones para especificar las codificaciones.

**9.3.1** Las restricciones 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 restricciones de variables no son visibles a PER (véanse 10.3 y 10.4 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

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

**9.3.4** Las restricciones de relación de componentes no son visibles a PER (véase 10.7 de la Rec. UIT-T X.682 | ISO/CEI 8824-3).

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

**9.3.6** Las restricciones de tipos de cadenas de caracteres restringidos que no son tipos de cadenas de caracteres de multiplicador conocido (véase la cláusula 37 de la Rec. UIT-T X.680 | ISO/CEI 8824-1) no son visibles a PER (véase 3.6.16).

**9.3.7** Las restricciones de patrón no son visibles a PER.

**9.3.8** De acuerdo con lo anterior, todas las restricciones de tamaño son visibles a PER.

**9.3.9** La restricción de tamaño efectiva para un tipo consiste en permitir un tamaño solamente si hay algún valor del tipo restringido que tiene ese tamaño (permitido).

**9.3.10** Las restricciones de alfabeto permitido en tipos de cadenas de caracteres de multiplicador conocido que no son extensibles tras la aplicación de 48.3 a 48.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, son visibles a PER. Las restricciones de alfabeto permitido que son extensibles no son visibles a PER.

**9.3.11** La restricción de alfabeto permitido efectiva para un tipo consiste en permitir un carácter solamente si hay algún valor del tipo restringido que contiene ese carácter. Si todos los caracteres del tipo restringido pueden estar presentes en algún valor del tipo restringido, entonces la restricción de alfabeto permitido efectiva es el conjunto de caracteres definido para el tipo no restringido.

**9.3.12** Las restricciones aplicadas a tipos reales no son visibles a PER.

**9.3.13** Una restricción 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 sólo valor, o cuando se utiliza, para restringir *identificación* a la alternativa *fija* (véanse las cláusulas 25 y 28).

**9.3.14** Las restricciones en estos tipos útiles no son visibles a PER.

**9.3.15** Las restricciones de subtipo de valor único aplicadas a un tipo cadena de caracteres no son visibles a PER.

**9.3.16** A reserva de lo anterior, todas las otras restricciones son visibles a PER única y exclusivamente si se aplican a un tipo entero o a tipos de cadenas de caracteres de multiplicador conocido.

**9.3.17** En general, restringir un tipo es aplicar distintas restricciones combinadas utilizando total o parcialmente la aritmética de conjuntos, restricciones de subtipos y la aplicación en serie de restricciones. En las siguientes cláusulas se especifican los efectos cuando algunos de los componentes de la restricción total son visibles a PER y otros no.

NOTA – Véase en el anexo B más detalles sobre el efecto de combinar restricciones visibles a PER y no visibles a PER.

**9.3.18** Si una restricción está formada por una aplicación en serie de restricciones, las restricciones que no son visibles a PER (en su caso), no afectan a las codificaciones PER, pero hacen que se elimine, como se especifica en 46.8 de la Rec. UIT-T X. 680 | ISO/CEI 8824-1, la extensibilidad (y las adiciones de extensión) que haya en cualquier restricción anterior.

NOTA 1 – Si la restricción final de una aplicación en serie no es visible a PER, el tipo no es extensible para las codificaciones PER y se codifica sin bit de extensión.

NOTA 2 – Por ejemplo:

```
A ::= IA5String(SIZE(1..4))(FROM("ABCD",...))
```

tiene una restricción de alfabeto permitido efectiva que consta de todo el alfabeto *IA5String*, ya que la restricción de alfabeto permitido extensible no es visible a PER. Sin embargo, tiene una restricción de tamaño efectiva que es "SIZE(1..4)".

De manera similar,

```
B ::= IA5String(A)
```

tiene la misma restricción de tamaño efectiva y la misma restricción de alfabeto permitido efectiva.

**9.3.19** Si una restricción visible a PER forma parte de una construcción *INTERSECTION*, la restricción resultante es visible a PER y está formada por la *INTERSECTION* de todas las partes visibles a PER (y se ignoran las partes no visibles a PER). Si una restricción no visible a PER es parte de una construcción *UNION*, la restricción resultante no es visible a PER. Si una restricción tiene una cláusula *EXCEPT*, el *EXCEPT* y el conjunto de valores siguiente se ignoran completamente, en todos los casos (que ese conjunto de valores que sigue a *EXCEPT* sea o no visible a PER).

NOTA – Por ejemplo:

```
A ::= IA5String(SIZE(1..4) INTERSECTION FROM("ABCD",...))
```

tiene una restricción de tamaño efectiva de 1..4, pero la restricción de alfabeto no es visible al ser extensible.

**9.3.20** Un tipo también es extensible para codificaciones PER (esté o no restringido posteriormente) si se ha producido de algunas de las formas siguientes:

- se deriva de un tipo *ENUMERADO* (*ENUMERATED*) (por subtipificación, referencia de tipo o rotulación) y hay un marcador de extensión en la producción "Enumeraciones" ("Enumerations"); o
- se deriva de un tipo *SECUENCIA* (*SEQUENCE*) (por subtipificación, referencia de tipo o rotulación) y hay un marcador de extensión en las producciones "Listas de tipos de componentes" ("ComponentTypeLists") o "tipo de secuencia" ("SequenceType"); o
- se deriva de un tipo *CONJUNTO* (*SET*) (por subtipificación, referencia de tipo o rotulación) y hay un marcador de extensión en las producciones "ComponentTypeLists" o "tipo de conjunto" ("SetType"); 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 "listas de tipos alternativos" ("AlternativeTypeLists").

**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 también permite la aplicación de restricciones y especificaciones de error. Las restricciones visibles a PER están presentes en el modelo como una restricción de los valores de un tipo. Otras restricciones 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.

**9.4.3** Cuando una restricción incluye un valor como una adición de extensión presente en la raíz, dicho valor siempre se codificará como un valor de la raíz, y no como un valor de adición de extensión.

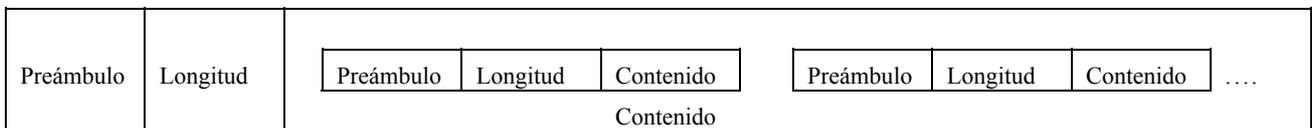
EJEMPLO

```
INTEGER (0..10, ..., 5)
-- El valor 5 se codifica como un valor raíz y no como una adición de
extensión.
```

**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).

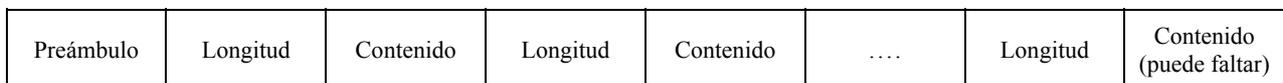


NOTA – Preámbulo, longitud y contenido son "campos" que, concatenados unos tras otros, forman una "lista de campos". La lista de campos de un tipo compuesto, salvo el tipo elección, puede estar constituida por los campos de varios valores concatenados unos tras otros. Pueden faltar, o bien el preámbulo,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

- a) puede consistir 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) puede consistir 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 a octeto en la variante ALIGNED), 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**

## ISO/CEI 8825-2:2002 (S)

**9.5.3** Cada una de las partes mencionadas en 9.5.2 genera:

- a) un campo nulo (nada); o
- b) un campo de bits (no alineado); o
- c) un campo de bits (alineados a octeto en la variante ALIGNED); o
- d) una lista de campos que puede contener campos de bits (no alineados), campos de bits (alineados a octeto en la variante ALIGNED), 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 no restringida (unrestricted character string).

**9.6.2** La codificación ha de corresponder al tipo seleccionado.

**9.6.3** 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.

**9.6.4** Los siguientes "tipos útiles" se codificarán como si hubiesen sido sustituidos por sus definiciones dadas en la cláusula 41 de la Rec. UIT-T X.680 | ISO/CEI 8824-1:

- tiempo generalizado;
- tiempo universal;
- descriptor de objeto.

Las restricciones en estos tipos útiles no son visibles a PER. Las restricciones para codificación de los tipos de tiempo generalizado y tipos de tiempo universal de las cláusulas 11.7 y 11.8 de la Rec. UIT-T X.690 | ISO/CEI 8825-1, se han de aplicar en este caso.

**9.6.5** Un tipo definido utilizando una asignación de conjunto de valores se codificará como si el tipo se hubiese definido utilizando la producción especificada en la cláusula 15.8 de la Recomendación UIT-T X.690 | ISO/CEI 8825-1.

## 10 Procedimientos de codificación

### 10.1 Producción de la codificación completa

**10.1.1** Si un tipo ASN.1 se codifica utilizando cualquiera de las reglas de codificación identificadas por los identificadores de objeto enumerados en la cláusula 29.2 (o por referencia textual directa a esta Recomendación | Norma Internacional), y la codificación se incluye en:

- a) una cadena de bits ASN.1 o una cadena de octetos ASN.1 (con o sin restricción de contenido); o
- b) un tipo abierto ASN.1; o
- c) cualquier parte de un tipo pdv externo o insertado ASN.1; o
- d) cualquier protocolo de portadora no definido utilizando ASN.1,

el tipo ASN.1 se define como un tipo extremo para esta aplicación, y se aplicará la cláusula 10.1.2 a todas las codificaciones de sus valores.

NOTA 1 – Esto significa que todas las codificaciones PER completas (para todas las variantes) que se utilizan de esta manera siempre serán un entero múltiplo de ocho bits.

NOTA 2 – Es posible utilizar la notación de control de codificación (véase la Rec. UIT-T X.692 | ISO/CEI 8825-3) para especificar una variante de codificación PER cuando no se ha rellenado la codificación hasta el límite del octeto, como se especifica en 10.1.2. Muchas herramientas soportan esta opción.

NOTA 3 – Obsérvese que un protocolo de portadora no definido utilizando ANS.1 no ha de transportar explícitamente los bits cero adicionales de relleno (especificados en 10.1.2) pero sí puede implicar su presencia.

**10.1.2** La lista de campos producida como un resultado de la aplicación de esta Recomendación | Norma Internacional al valor abstracto de un tipo más externo se utilizará para producir la codificación completa de ese valor de sintaxis abstracta de esta forma: 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.

**10.1.3** 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 será sustituida 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, se le añadirán bits cero (cero a siete) para producir un múltiplo de ocho bits.

**10.1.4** 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 restringido a un solo valor.

**10.1.5** La cadena de bits resultante es la codificación completa del valor de sintaxis abstracta de un tipo más externo.

## 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" (por ejemplo).

**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 restringida de "n" (en unidades de octetos) y un campo de bits (alineado a octeto en la variante ALIGNED) 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) En esta subcláusula se limita el término "codificación de entero binario no negativo" a los casos de 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 subcláusulas se refieren a la generación de una codificación entera binaria no negativa de un número entero no negativo en un campo que es, o bien un campo de bits de longitud especificada, o un solo octeto, o un octeto doble, o el número mínimo de octetos para el valor. Esta subcláusula (10.3) 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 primer bit del campo de bits o como el bit más significativo del primer octeto de este campo, y el bit posterior del campo se define como el último bit del campo de bits o como el bit menos significativo del último octeto de este campo.

**10.3.3** 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.

**10.3.4** En la codificación de un entero binario no negativo, 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** Cuando un código se traduce (según se define anteriormente) en el valor que se está codificando, se trata de 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), sólo hay una codificación que se traduce en 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) En esta subcláusula se limita el término "codificación entera binaria complemento de dos" a los casos de 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 subcláusulas 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 la codificación de un entero binario 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** Cuando un código se traduce (como se define anteriormente) en el valor que se codifica se trata de 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 restringido

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 (10.5) especifica una correspondencia de un número entero restringido a un campo de bits (no alineado) o un campo de bits (alineado a octeto en la variante ALIGNED) y es invocada en cláusulas posteriores de esta Recomendación | Norma Internacional.

**10.5.2** Los procedimientos de esta subcláusula se invocan solamente si hay un número entero restringido 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 restricciones 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** Definase "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  $2^m < \text{"gama"} \leq 2^{m+1}$ , 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 subcláusula 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 (alineado a octeto en la variante ALIGNED) 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 el valor de la "gama" es 257 o mayor, y 64K o menor, el valor ("n" – "lb") se codificará en un campo de bits (alineados en octeto en la variante ALIGNED) 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 (alineado a octeto en la variante ALIGNED) 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 64 o mayor, 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 semirrestringido 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 semirrestringido

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 semirrestringido con un campo de bits (alineado a octeto en la variante ALIGNED) y se invoca en cláusulas ulteriores de esta Recomendación | Norma Internacional.

**10.7.2** Los procedimientos de esta subcláusula (10.7) se invocan solamente si se dispone de un número entero semirrestringido (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 restricciones 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 (alineado a octeto en la variante ALIGNED) 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.

## 10.8 Codificación de un número entero no restringido

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 (10.8) especifica una correspondencia de un número entero sin restricciones (digamos "n") con un campo de bits (alineado a octeto en la variante ALIGNED) 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 (alineado a octeto en la variante ALIGNED) 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

NOTA 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 restricciones 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).

NOTA 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 restringido se utiliza para la longitud. Para gamas suficientemente pequeñas, el resultado es un campo de bits. En los demás casos, la longitud no restringida (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.

NOTA 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 restringido para codificar la longitud en el número mínimo de bits necesarios para representar la gama. En los demás casos, la longitud no restringida (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 (restringido por restricciones 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);
- b) la longitud en bits de un campo asociado (las unidades son bits);
- 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 a 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;
- b) un número entero restringido con un límite inferior "lb" (mayor o igual que cero) y un límite superior "ub" menor que 64K; o
- c) un número entero semirrestringido con un límite inferior "lb" (mayor o igual que cero), o un número entero restringido 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 restringido con "ub" menor que 64K, se añadirá a la lista de campos la codificación del número entero restringido para el determinante de longitud como se especifica en 10.5. 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 1 – Por ejemplo:

```
A ::= IA5String (SIZE (3..6))           -- La longitud se codifica en un campo de bits de
                                         -- dos bits
B ::= IA5String (SIZE (40000..40254))   -- La longitud se codifica en un campo de bits de
                                         -- ocho bits
C ::= IA5String (SIZE (0..32000))       -- La longitud se codifica en un campo de bits
                                         -- (alineado a octeto en la variante ED de
                                         -- dos octetos)
D ::= IA5String (SIZE (64000))         -- La longitud no se codifica
```

NOTA 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 restringido, seguido por el campo asociado, de acuerdo con 10.9.3.5 a 10.9.3.8.4.

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

**10.9.3.5** En otro caso (longitud no restringida, o "ub" grande), "n" se codifica y se añade a la lista de campos, seguido de los campos asociados como se especifica más adelante.

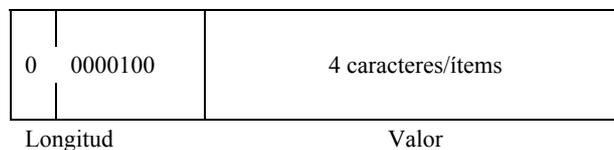
NOTA – La cota inferior, "lb", no afecta a las codificaciones de longitud especificadas en 10.9.3.6 a 10.9.3.8.4.

**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 a octeto en la variante ALIGNED), 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 ::= SEQUENCE (SIZE (4..123456)) OF INTEGER
```

ambos valores se codifican de modo que el octeto de longitud ocupe un octeto y con el bit más significativo 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) entre el bit 6 del octeto uno (más significativo) al bit 1 del octeto dos (menos significativo) de un campo de bits (alineado a octeto en la variante ALIGNED) 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 bits más significativos (bits 8 y 7) del octeto puestos a 10 para indicar que la longitud es mayor que 127 y menor que 16K.

10	000000 10000010	130 caracteres/ítems
Longitud		Valor

**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 a octeto en la variante ALIGNED) 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.

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

NOTA 2 – Si en el ejemplo de 10.9.3.6 un valor de "B" tiene una longitud de 144K + 1 ítems (es decir, 64K + 64K + 16K + 1), el valor se fragmenta, con los bits más significativos (bits 8 y 7) 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 fragmento:

11	000100	ítems de 64K	11	000100	ítems de 64K	11	000001	ítems de 16K	0	0000001	ítem de 1
Longitud		Valor	Longitud		Valor	Longitud		Valor	Longitud		Valor

**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 (alineado a octeto en la variante ALIGNED) de "m" octetos que contiene los primeros "m" octetos del campo asociado, para unidades que son octetos; o
- b) un campo de bits (alineado a octeto en la variante ALIGNED) 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 (alineado a octeto en la variante ALIGNED) 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 semirrestringido 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 restringido con "ub" menor que 64K, ("n" – "lb") 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" ("ub" – "lb" + 1), a menos que "gama" sea 1, en cuyo caso no habrá codificación de longitud. 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  $2^m < \text{"gama"} \leq 2^{m+1}$ , 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 de longitud normalmente pequeña, o un número restringido con "ub" igual a 64K o mayor, o es un número entero semirrestringido, "n" se codificará como se especifica en 10.9.3.4 a 10.9.3.8.4.

NOTA – Así pues, si "ub" es igual a 64K o mayor, la codificación del determinante de longitud es la misma que si la longitud no estuviese restringida.

## 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

NOTA 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.

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

12.1 Si la especificación de restricciones del tipo entero 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 se pondrá a 1 si el valor que se ha de codificar no está dentro de la gama de la restricción 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 restringido, 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 la especificación de restricción de un tipo entero no tiene un marcador de extensión, se aplica lo siguiente.

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

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

12.2.4 Si no hay restricciones visibles a PER que fuercen el valor entero a ser un número entero restringido, o semirrestringido, deberá convertirse en un campo de acuerdo con los procedimientos de 10.8 (codificación de un número entero no restringido), 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 restringida "len" (establecido como se especifica en 10.5.7.4) si restricciones visibles a PER limitan el tipo con cotas superior e inferior finitas 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 con los dos bits menos significativos (bits 2 y 1) de un octeto).

- b) Un determinante de longitud no restringida igual a "len" (establecido como se especifica en 10.7 y 10.8), si restricciones 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 restringido cuya restricción 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 | ISO/CEI 8824-1 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 de acuerdo con 13.1 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 restricciones visibles a PER que pueden 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 CER y DER en la Rec. UIT-T X.690 | ISO/CEI 8825-1 para obtener un campo de bits (alineado a octeto en la variante **ALIGNED**) que constituye los octetos de contenido de la codificación CER/DER. Los octetos de contenido de esta codificación se componen de (por ejemplo) "n" octetos y se colocan en un campo de bits (alineado a octeto en la variante **ALIGNED**) de "n" octetos. Se invocarán los procedimientos de 10.9 para añadir este campo de bits (alineado a octeto en la variante **ALIGNED**) 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 restringidas a una longitud fija menor o igual que 16 bits no producen alineación de octetos. Las cadenas de bits más grandes están alineadas a octeto en la variante **ALIGNED**. Si la longitud se fija mediante restricciones 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 restricciones visibles a PER sólo pueden limitar la longitud de la cadena de bits.

**15.2** Cuando no hay restricciones visibles a PER y se aplica 21.7 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 restricción visible a PER y se aplica 21.7 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 posteriores, 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 restricción de tamaño efectiva.

**15.4** Sea "ub" el número máximo de bits en la cadena de bits (determinados por las restricciones 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 restricciones sobre el 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** Cuando un valor cadena de bits se coloque en un campo de bits como se especifica en 15.6 a 15.11, el primer bit de ese valor cadena bits ha de quedar en la posición del primer bit del campo de bits, y el último bit del valor cadena de bits en la posición del último bit del campo de bits.

**15.6** Si está presente un marcador de extensión en la especificación de restricción 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 15.11 para añadir la longitud como un número entero semirrestringido 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 el marcador de extensión no estuviese presente.

**15.7** Si no está presente un marcador de extensión en la especificación de restricciones del tipo cadena de bits, se aplican 15.8 a 15.11.

**15.8** Si la cadena de bits está restringida a 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.

**15.9** Si todos los valores de la cadena de bits están restringidos a la misma longitud ("ub" igual "lb") y esa longitud es menor o igual que 16 bits, la cadena de bits se colocará en un campo de bits de la longitud restringida "ub" que se añadirá a la lista de campos sin determinante de longitud, y se finalizan los procedimientos de esta cláusula.

**15.10** Si todos los valores de la cadena de bits están restringidos a la misma longitud ("ub" igual "lb") y esa longitud es mayor que 16 bits pero menor que 64K bits, la cadena de bits se colocará en un campo de bits (alineado a octeto en la variante ALIGNED) de la longitud restringida "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.

**15.11** Si no se da ninguna de las condiciones de 15.8 a 15.10, la cadena de bits se colocará en un campo de bits (alineado a octeto en la variante ALIGNED) de longitud "n" bits y se invocarán los procedimientos de 10.9 para añadir este campo de bits (alineado a octeto en la variante ALIGNED) de "n" bits a la lista de campos, precedido por un determinante de longitud igual a "n" bits como un número entero restringido si "ub" está fijado y es menor que 64K o como un número entero semirrestringido si "ub" no está fijado. "lb" es como se determina anteriormente.

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

## 16 Codificación del tipo cadena de octetos

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

**16.1** Las restricciones visibles a PER sólo pueden limitar la longitud de la cadena de octetos.

**16.2** Sea "ub" el número máximo de octetos en la cadena de octetos (determinado por las restricciones 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 restricciones sobre el 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 restricción 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 semirrestringido a la lista de campos, seguido por el valor de la 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 restricciones del tipo cadena de bits, se aplican 16.5 a 16.8.

**16.5** Si la cadena de octetos está restringida 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 restringidos 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 restringida "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 restringidos a 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 (alineado

a octeto en la variante ALIGNED) y se añadirán los octetos "ub" de longitud restringida a la lista de octetos sin determinante de longitud, y se finalizan los procedimientos de esta cláusula.

**16.8** Si no se da ninguna de las condiciones de 16.5 a 16.7, la cadena de bits se colocará en un campo de bits (alineado a octeto en la variante ALIGNED) de longitud "n" bits y se invocarán los procedimientos de 10.9 para añadir este campo de bits (alineado a octeto en la variante ALIGNED) de "n" bits a la lista de campos, precedido por un determinante de longitud igual a "n" bits como un número entero restringido si "ub" está fijado o como un número entero semirrestringido 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 una reserva de lugar, 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 hay adiciones de extensión, inmediatamente antes de que se codifique la primera, hay la codificación (como una longitud normalmente pequeña) 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** o **DEFAULT**, 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 restringido 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 3.6.25); en todos los demás casos la decisión de codificar o no queda a discreción del emisor.

**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, tomados en orden, indicarán la presencia o ausencia de una codificación de cada adición de extensión en el tipo que se codifica. Un valor de bit de uno indicará la

presencia de la codificación de la adición de extensión, y un valor de bit de cero indicará la ausencia de la codificación de la adición de extensión. El bit anterior en el campo de bits indicará la presencia o ausencia de la primera adición de extensión y el bit posterior indicará la presencia o ausencia de la última adición de extensión.

NOTA – Para indicar conformidad con una determinada versión de una especificación, el valor de "n" ha de ser igual al número de adiciones de extensión de esa versión.

**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, una tras otra. Cada adición de extensión que sea un "ComponentType" (es decir no un "ExtensionAdditionGroup") se codificará como si fuese el valor de un campo de tipo abierto, como se especifica en 10.2.1. Cada adición de extensión que sea un "ExtensionAdditionGroup" se codificará como un tipo secuencia conforme a lo especificado en 18.2 a 18.6, que se codifica entonces como si fuese el valor de un campo de tipo abierto como se especifica en 10.2.1. Si no hay todos los valores de componentes "ExtensionAdditionGroup", el "ExtensionAdditionGroup" deberá codificarse como una adición de extensión que falta (es decir, el bit correspondiente en el campo de bits descrito en 18.7 debe ser puesto a 0).

NOTA 1 – Si un "ExtensionAdditionGroup" contiene componentes marcados **OPTIONAL** o **DEFAULT**, el "ExtensionAdditionGroup" está precedido por un mapa de bits que indica la presencia/ausencia de valores para cada componente marcado **OPTIONAL** o **DEFAULT**.

NOTA 2 – Los componentes "RootComponentTypeList" que están definidos después del par marcador de extensión se codifican como si estuvieran definidos inmediatamente antes del par marcador de extensión.

## 19 Codificación del tipo secuencia de

**19.1** Las restricciones visibles a PER pueden limitar el número de componentes del tipo secuencia de.

**19.2** Sea "ub" el número máximo de componentes en la secuencia de (determinados por restricciones 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 restricciones 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 restricción 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 10.9 para añadir la longitud como un número entero semirrestringido 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 restringido si "ub" está fijado, y como un número entero semirrestringido si "ub" no está fijado. "lb" es como se determina anteriormente.

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

NOTA 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 8.6 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 "RootComponentTypeList" se codificarán como si ésta se hubiera declarado un tipo de secuencia. Los elementos

de conjunto que aparecen en la "ExtensionAdditionList" se codificarán como si fuesen componentes de un tipo de secuencia especificado en 18.9 (es decir, se codifican en el orden en el cual están definidos).

**EJEMPLO** – En lo que sigue, se supone un entorno rotulado de **IMPLICIT TAGS**:

```

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].

## 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 rellenas en sus extremos posteriores con hasta un máximo de siete bits cero para alcanzar un límite de octeto, y con octetos cero agregados al más corto si es necesario para que su longitud sea igual a la del más largo.

NOTA – Cualesquiera bits u octetos de relleno añadidos para la clasificación no aparecen 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. Ésta se codifica como para un entero restringido (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 restricciones 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 8.6 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. También se asigna un valor de índice "NamedType" en "ExtensionAdditionAlternativesList", empezando por 0 como en el caso de componentes de la raíz de extensión. Sea "n" el valor del índice mayor en la raíz.

NOTA – La cláusula 28.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 requiere que cada adición de extensión sucesiva tenga un valor de rótulo mayor que el de la última añadida a la "ExtensionAdditionAlternativesList".

**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 restricción de subtipo) restringido 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 la cláusula 12, 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.

NOTA – La indicación entre paréntesis de la versión de la definición de las adiciones de extensión de elección no afecta a la codificación de "ExtensionAdditionAlternatives".

## 23 Codificación del tipo identificador de objeto

NOTA – (Didáctica) Una codificación del 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 obtener un campo de bits (alineado a octeto en la variante ALIGNED) que es los octetos de contenido de la codificación BER. Los octetos de contenido de esta codificación BER consisten en (por ejemplo) "n" octetos y se colocan en un campo de bits (alineado a octeto en la variante ALIGNED) de "n" octetos. Se invocarán los procedimientos de 10.9 para añadir este campo de bits (alineado a octeto en la variante ALIGNED) a la lista de campos, precedido por un determinante de longitud igual a "n" como una cuenta de octetos de número entero semirrestringido.

## 24 Codificación del tipo identificador de objeto relativo

NOTA – (Didáctica) Una codificación del tipo identificador de objeto relativo utiliza los octetos de contenido de BER precedidos por un determinante de longitud que en la práctica será un solo octeto. El texto que sigue es idéntico al de la cláusula 23.

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

## 25 Codificación del tipo PDV INCRUSTADO

**25.1** Un tipo pdv incrustado se puede codificar de dos maneras:

- a) la alternativa **sintaxis** del tipo pdv incrustado tiene una restricción de tipo interno visible a PER a un solo valor, o **identificación** tiene una restricción de tipo interno visibles a PER a la alternativa **fija**, en cuyo caso sólo se codifica el **valor de datos**: este caso es llamado "predefinido";
- b) una restricción de tipo interno no se emplea para limitar la alternativa **sintaxis** a un solo valor, ni para limitar **identificación** a la alternativa **fijo**, en cuyo caso se codifica **identificación** y **valor de datos**: este caso es llamado "general".

**25.2** En el caso "predefinido", la codificación del valor del tipo pdv incrustado será la codificación PER del valor del tipo **OCTET STRING**. El valor de la **OCTET STRING** serán los octetos que forman la codificación completa del valor de datos referenciado en 33.3 a) de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

**25.3** En el caso "general", la codificación de un valor del tipo pdv incrustado será la codificación PER del tipo definido en 33.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 con la supresión del elemento **data-value-descriptor** (es decir, en la tabla de bits no habrá **OPTIONAL** en el encabezamiento de la codificación **SEQUENCE**). El valor del componente **data-value** del tipo **OCTET STRING** valor de datos serán los octetos que formarán la codificación completa de un solo valor referenciado en 33.3 a) de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

## 26 Codificación de un valor del tipo externo

**26.1** 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** con un valor especificado en las subcláusulas 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 difiere del especificado en la Rec. UIT-T X.680 | ISO/CEI 8824-1 por motivos históricos.

**26.2** El valor de los componentes depende del valor abstracto que se transmite, que es un valor del tipo especificado en 33.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

**26.3** 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.

**26.4** 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 33.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, con los componentes del tipo externo **referencia directa** y **referencia indirecta** definidos en 26.1.

**Cuadro 1 – Codificaciones alternativas para "identificación"**

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

**26.5** 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.

**26.6** Si el valor de datos es el valor de un tipo de datos ASN.1 simple (véase la nota de 26.7), y si las reglas de codificación para este valor de datos son las especificados en esta Recomendación | Norma Internacional, la implementación emisora utilizará la alternativa **tipo ASN.1 simple**.

**26.7** En cualquier otro caso, si la codificación, convenida o negociada, del valor de datos es un número integral de octetos, la implementación emisora codificará como **alineado a 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 26.6.

**26.8** En cualquier otro caso, si la codificación, convenida o negociada, del valor de datos no es un número integral de octetos, la elección de **codificación** será **arbitraria**.

**26.9** Si la elección de **codificación** es **tipo ASN.1 simple**, el tipo ASN.1 se codificará como se especifica en 10.2, 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**.

**26.10** 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.

**26.11** 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.

## 27 Codificación de los tipos cadenas de caracteres restringidas

NOTA 1 – (Explicación de la variante ALIGNED) Las cadenas de caracteres de longitud fija menor o igual que dos octetos no están alineadas a octeto. Las cadenas de caracteres de longitud variable que están limitadas a una longitud máxima menor que dos octetos no están alineadas en octetos. Todas las demás cadenas de caracteres están alineadas en octeto en la variante ALIGNED. Las cadenas de caracteres de longitud fija se codifican sin octetos de longitud si son más cortas que 64K. Para cadenas de caracteres no restringidas o restringidas a una longitud mayor 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 restricción de alfabeto permitido efectiva.

NOTA 2 – (Explicación de la variante UNALIGNED) Las cadenas de caracteres no están alineadas a octeto. 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 restringidas o restringidas a una longitud mayor 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 restricción alfabeto permitido efectiva.

NOTA 3 – (Explicación del tamaño de cada carácter codificado) La codificación de cada carácter depende de la restricción alfabeto permitido efectiva (véase 9.3.11) que define el alfabeto en uso para el tipo. Supóngase que este alfabeto consiste (por ejemplo) en un juego de caracteres ALPHA. Para cada uno de los tipos cadena de caracteres de multiplicador conocido (véase 3.6.16) 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 (por ejemplo) 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.

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

**27.2** La notación de restricción 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.

**27.3** La notación de restricción 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 restricciones 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.

**27.4** Si el tipo es extensible para codificaciones PER (véase 9.3.16), 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 se hará como si no hubiese ninguna restricción de tamaño efectiva y se aplicará una restricción de alfabeto permitido efectiva formada por un conjunto de caracteres del tipo no restringido.

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.

**27.5** Esta subcláusula se aplica a cadenas de caracteres de multiplicador conocido. La codificación de otros tipos de cadenas de caracteres restringidas se especifican en 27.6.

**27.5.1** El alfabeto permitido efectivo será el que indica la restricción de alfabeto permitido por la restricción alfabeto permitido o todo el alfabeto del tipo incorporado si no hay ninguna restricción de alfabeto permitido.

**27.5.2** 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.

## ISO/CEI 8825-2:2002 (S)

**27.5.3** Se asocia un valor numérico "v" con cada carácter por referencia a la cláusula 39 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 39.3 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 (el valor está en la gama 0 a  $2^{32} - 1$ ). Para **BMPString**, el valor es el utilizado para determinar el orden canónico en 39.3 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 (el valor está en la gama 0 a  $2^{16} - 1$ ). Para **NumericString** y **PrintableString** y **VisibleString** y **IA5String**, el valor es el definido para la codificación según ISO/CEI 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.)

**27.5.4** 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 39 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 – El apartado a) no se puede aplicar nunca a un carácter **NumericString** restringido o no restringido, que siempre se codifica en cuatro bits o menos utilizando b).

**27.5.5** 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.

**27.5.6** 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 a octeto en la variante ALIGNED) si el producto de "aub" por "b" es mayor que 16, pero en los demás casos se añadirá como un campo de bits que no está alineado a octeto. Esto completa los procedimientos de esta subcláusula.

**27.5.7** 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 a octeto en la variante ALIGNED) si el producto de "aub" por "b" es mayor o igual que 16, pero en los demás casos se añadirá como un campo de bits que no está alineado a octeto. Esto completa los procedimientos de esta subcláusula.

NOTA – Ni 27.5.6 ni 27.5.7 especifican la alineación si el producto de "aub" por "b" es inferior a 16, y sí especifican la alineación si el producto es superior a 16. Para un valor igual a 16, 27.5.6 especifica la no alineación y 27.5.7 especifica la alineación.

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

**27.6.1** Para BASIC-PER, la referencia siguiente a "codificación básica" significa los octetos de contenido de una codificación BER. Para CANONICAL-PER significa los octetos de contenido de la codificación especificada para CER y DER en la Rec. UIT-T X.690 | ISO/CEI 8825-1.

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

**27.6.3** Se invocará 10.9 para añadir un determinante de longitud no restringido con "n" como una cuenta en octetos y el campo de "n" octetos se añadirá como un campo de bits (alineado a octeto en la variante ALIGNED), lo que termina los procedimientos de esta subcláusula.

## 28 Codificación del tipo cadena de caracteres no restringido

**28.1** Hay dos maneras de codificar un tipo cadena de caracteres no restringido:

- a) la alternativa **sintaxis** del tipo cadena de caracteres no restringido tiene una restricción de tipo interno visible a PER a un solo valor, o **identificación** tiene una restricción de tipo interno visible a PER a la alternativa **fija**, en cuyo caso sólo se codifica **valor de cadena**, éste es llamado caso "predefinido";
- b) no hay restricción de tipo interno para la alternativa **sintaxis** a un solo valor, ni para **identificación** a la alternativa **fijo**, en cuyo caso se codifica **identificación** y **valor de cadena**; éste es el llamado caso "general".

**28.2** Para el caso "predefinido", la codificación del valor del tipo OCTET STRING será la codificación PER de un valor del tipo OCTET STRING. El valor de la OCTET STRING serán los octetos que forman la codificación completa del valor cadena de caracteres referenciado en 40.3 a) de la Rec. UIT-T X.680 | ISO/CEI 8824-1

**28.3** En el caso "general" la codificación de un valor de tipo cadena de caracteres no restringido será la codificación PER de tipo definido en 40.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 con la supresión del componente **data-value-descriptor** (esto es, en la tabla de bits no habrá OPTIONAL en el encabezamiento de la codificación SEQUENCE). El valor del componente **string-value** del tipo OCTET STRING serán los octetos que forman la codificación de un esto valor de cadena referenciado en 40.3 a) de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

## 29 Identificadores de objeto para las sintaxis de transferencia

**29.1** Las reglas de codificación especificadas en esta 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.

**29.2** Se asignan los siguientes valores de identificadores de objeto y de descriptor de objeto para identificar y describir las reglas de codificación especificadas en esta 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)"
```

**29.3** 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 29.2 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 esta Recomendación | Norma Internacional para el tipo ASN.1 específicamente denominado, utilizado para definir la sintaxis abstracta.

**29.4** Los nombres especificados en 29.2 no se utilizarán con un nombre de sintaxis abstracta para identificar una sintaxis de transferencia si no se cumplen las condiciones de 29.3 para la definición de la sintaxis abstracta.

## Anexo A

## Ejemplos 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 esta Recomendación | Norma Internacional mostrando representaciones en octetos de una ficha de personal (hipotética) que se define mediante la notación ASN.1.

## A.1 Ficha sin restricciones de subtipo

## A.1.1 Descripción ASN.1 de la estructura de la ficha

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 la ficha

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 la ficha

A continuación se muestra la representación del valor mencionado (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 por 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
0000100 01001010 01101111 01101000 01101110	Longitud de name.givenName = 4 name.givenName = "John"
0000001 01010000	Longitud de name.initial = 1 name.initial = "P"
0000101 01010011 01101101 01101001 01110100 01101000	Longitud de name.familyName = 5 name.familyName = "Smith"
0000001 00110011	Longitud de (employee) number = 1 (employee) number = 51
0000100 01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	Longitud de title = 8 title = "Director"
0000100 00110001 00111001 00110111 00110001 00110000 00111001 00110001 00110111	Longitud de dateOfHire = 8 dateOfHire = "19710917"
0000100 01001101 01100001 01110010 01111001	Longitud de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
0000001 01010100	Longitud de nameOfSpouse.initial = 1 nameOfSpouse.initial = "T"
0000101 01010011 01101101 01101001 01110100 01101000	Longitud de nameOfSpouse.familyName = 5 nameOfSpouse.familyName = "Smith"
00000010	Número de hijos
0000101 01010010 01100001 01101100 01110000 01101000	Longitud de children[0].givenName = 5 children[0].givenName = "Ralph"
0000001 01010100	Longitud de children[0].initial = 1 children[0].initial = "T"
0000101 01010011 01101101 01101001 01110100 01101000	Longitud de children[0].familyName = 5 children[0].familyName = "Smith"
0000100 00110001 00111001 00110101 00110111 00110001 00110001 00110001 00110001	Longitud de children[0].dateOfBirth = 8 children[0].dateOfBirth = "19571111"
0000101 01010011 01110101 01110011 01100001 01101110	Longitud de children[1].givenName = 5 children[1].givenName = "Susan"
0000001 01000010	Longitud de children[1].initial = 1 children[1].initial = "B"
0000101 01001010 01101111 01101110 01100101 01110011	Longitud de children[1].familyName = 5 children[1].familyName = "Jones"
0000100 00110001 00111001 00110101 00111001 00110000 00110111 00110001 00110111	Longitud de children[1].dateOfBirth = 8 children[1].dateOfBirth = "19590717"

**A.1.4 Representación UNALIGNED PER de este valor de la ficha**

A continuación se muestra la representación del valor mencionado (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 por 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.

## ISO/CEI 8825-2:2002 (S)

La longitud de esta codificación es 84 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante UNALIGNED 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 octetos y una 'x' representa un bit cero utilizado para rellenar el octeto final hasta un límite de octeto.

1	Bitmap bit = 1 indica que "children" está presente
000010.0 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 .0110111	Longitud de dateOfHire = 8 dateOfHire = "19710917"
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 100.0010	Longitud de children[1].initial = 1 children[1].initial = "B"
0000.0101 1001.100 11011.11 110111.0 1100101 .1110011	Longitud de children[1].familyName = 5 children[1].familyName = "Jones"
0.0001000 0.110001 01.11001 011.0101 0111.001 01100.00 011011.1 0110001 .0110111x	Longitud de children[1].dateOfBirth = 8 children[1].dateOfBirth = "19590717"

## A.2 Ficha con restricciones 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 restricciones sobre algunos ítems.

### A.2.1 Descripción ASN.1 de la estructura de la ficha

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.

```

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 la ficha

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.2.3 Representación ALIGNED PER de este valor de la ficha

A continuación se muestra la representación del valor mencionado (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 por 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 que se emplean para alinear los campos en diversas ocasiones.

La longitud de esta codificación es 74 octetos. Para comparación, el mismo valor PersonnelRecord codificado mediante la variante UNALIGNED de PER 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

```

## ISO/CEI 8825-2:2002 (S)

### 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 y 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 01001010 01101111 01101000 01101110	Longitud de name.givenName = 4 name.givenName = "John"
01010000	name.initial = "P"
000100xx 01010011 01101101 01101001 01110100 01101000	Longitud de name.familyName = 5 name.familyName = "Smith"
00000001 00110011	Longitud de (employee) number = 1 (employee) number = 51
00001000 01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	Longitud de title = 8 title = "Director"
0001 1001 0111 0001 0000 1001 0001 0111	dateOfHire = "19710917"
000011xx 01001101 01100001 01110010 01111001	Longitud de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
01010100	nameOfSpouse.initial = "T"
000100xx 01010011 01101101 01101001 01110100 01101000	Longitud de nameOfSpouse.familyName = 5 nameOfSpouse.familyName = "Smith"
00000010	Número de hijos
000100xx 01010010 01100001 01101100 01110000 01101000	Longitud de children[0].givenName = 5 children[0].givenName = "Ralph"
01010100	children[0].initial = "T"
000100xx 01010011 01101101 01101001 01110100 01101000	Longitud de children[0].familyName = 5 children[0].familyName = "Smith"
0001 1001 0101 0111 0001 0001 0001 0001	children[0].dateOfBirth = "19571111"
000100xx 01010011 01110101 01110011 01100001 01101110	Longitud de children[1].givenName = 5 children[1].givenName = "Susan"
01000010	children[1].initial = "B"
000100xx 01001010 01101111 01101110 01100101 01110011	Longitud de children[1].familyName = 5 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 la ficha

A continuación se muestra la representación del valor mencionado (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 por 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 UNALIGNED de 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 señalar límites de octetos y una 'x' representa un bit cero utilizado para rellenar el octeto final hasta 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 = "19710917"
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 0.001 1001 0.101 0111 0.001 0001 0.001 0001	Longitud de children[0].familyName = 5 children[0].familyName = "Smith" 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 marcadores de extensión

### A.3.1 Descripción ASN.1 de la estructura de la ficha

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.

```

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 la ficha

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", sex female}}}

```

### A.3.3 Representación ALIGNED PER de este valor de la ficha

A continuación se muestra la representación del valor mencionado (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 por 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 que 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 y 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
1                                     PersonnelRecord
1                                     Bitmap bit = 1 indica que "children" está
0                                     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 01001010 01101111 01101000 01101110	Longitud de name.givenName = 4 name.givenName = "John"
01010000	name.initial = "P"
0	La longitud está dentro de la gama de la raíz de extensión
000100x 01010011 01101101 01101001 01110100 01101000	Longitud de name.familyName = 5 name.familyName = "Smith"
0xxxxxxx	La longitud está dentro de la gama de la raíz de extensión
00000000 00110011	(employee) number = 51
00001000 01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	Longitud de title = 8 title = "Director"
0xxxxxxx	La longitud está dentro de la gama de la raíz de extensión
0001 1001 0111 0001 0000 1001 0001 0111	dateOfHire = "19710917"
0	No hay valores de extensión presentes en nameOfSpouse
0	La longitud está dentro de la gama de la raíz de extensión
000011 01001101 01100001 01110010 01111001	Longitud de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
01010100	nameOfSpouse.initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
000100x 01010011 01101101 01101001 01110100 01101000	Longitud de nameOfSpouse.familyName = 5 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 01010010 01100001 01101100 01110000 01101000	Longitud de children[0].givenName = 5 children[0].givenName = "Ralph"
01010100	children[0].initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
000100x 01010011 01101101 01101001 01110100 01101000	Longitud de children[0].familyName = 5 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[1].name
0	La longitud está dentro de la gama de la raíz de extensión
00010 0xxxxxxx 01010011 01110101 01110011 01100001 01101110	Longitud de children[1].givenName = 5 children[1].givenName = "Susan"
01000010	children[1].initial = "B"

## ISO/CEI 8825-2:2002 (S)

0	La longitud está dentro de la gama de la raíz de extensión
000100x 01001010 01101111 01101110 01100101 01110011	Longitud de children[1].familyName = 5 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 la ficha

A continuación se muestra la representación del valor mencionado (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 por 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 ALIGNED 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 señalar límites de octetos y una 'x' representa un bit cero utilizado para rellenar el octeto final hasta 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 001011 .101010 10.0011 1010.01	Longitud de name.givenName = 4 name.givenName = "John"
010001	name.initial = "P"
.0	La longitud está dentro de la gama de la raíz de extensión
000100 0.10100 101.000 10010.0 101111 1.00011	Longitud de name.familyName = 5 name.familyName = "Smith"
0	El valor está dentro de la gama de la raíz de extensión
00.00000011.0011	(employee) number = 51
0000.1000 1000.100 11010.01 111001.0 1100101 1100011 1.110100 11.01111 111.0010	Longitud de title = 8 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 = "19710917"

0	No hay valores de extensión presentes en nameOfSpouse
0	La longitud está dentro de la gama de la raíz de extensión
0.00011 001.110 01110.0 101101 1.10100	Longitud de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
010.101	nameOfSpouse.initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
0001.00 010100 .101000 10.0100 1011.11 100011	Longitud de nameOfSpouse.familyName = 5 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 010011 .011100 10.0111 1010.11 100011	Longitud de children[0].givenName = 5 children[0].givenName = "Ralph"
.010101	children[0].initial = "T"
0	La longitud está dentro de la gama de la raíz de extensión
0.00100 010.100 10100.0 100100 1.01111 100.011	Longitud de children[0].familyName = 5 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[1].name
0	La longitud está dentro de la gama de la raíz de extensión
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"
0	La longitud está dentro de la gama de la raíz de extensión
000100 .001011 10.1010 1010.01 100000 .101110	Longitud de children[1].familyName = 5 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 0.1xxxxxx x	Longitud de la codificación completa de "sex" Codificación completa de "sex" = female Bit de relleno para crear la codificación completa de PersonnelRecord

## A.4 Ficha con grupos de adición de extensión

### A.4.1 Descripción ASN.1 de la estructura de la ficha

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. Se supone **AUTOMATIC TAGS**.

```

Ax ::= SEQUENCE {
    a    INTEGER (250..253),
    b    BOOLEAN,
    c    CHOICE {
        d    INTEGER,
        ...
        [[
            e    BOOLEAN,
            f    IA5String
        ]],
        ...
    },
    ...
    [[
        g    NumericString (SIZE(3)),
        h    BOOLEAN OPTIONAL
    ]],
    ...
    i    BMPString OPTIONAL,
    j    PrintableString OPTIONAL
}

```

### A.4.2 Descripción ASN.1 de un valor de la ficha

El valor de **Ax** se describe formalmente a continuación mediante la ASN.1.

```
{ a 253, b TRUE, c e : TRUE, g "123", h TRUE }
```

### A.4.3 Representación ALIGNED PER de este valor de la ficha

A continuación se muestra la representación del valor mencionado (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 por 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 que se emplean para alinear los campos en diversas ocasiones.

La longitud de esta codificación es 8 octetos. Para comparación, el mismo valor codificado mediante la variante UNALIGNED de PER ocupa 8 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 22 octetos y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 26 octetos.

#### A.4.3.1 Presentación hexadecimal

```
9E000180 010291A4
```

#### A.4.3.2 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 y 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	Valores de adición de extensión presentes en Ax
00	Bitmap bits = 0 indica campos opcionales (i y j) ausentes
11	a = 253
1	b = VERDADERO (TRUE)
1	Valor de elección de c es un valor de adición de extensión
0000000 xx	Índice de elección selecciona c.e
00000001	Longitud de c.e
1xxxxxxx	c.e = VERDADERO (TRUE)

0000000	Número de adiciones de extensión definidas en Ax = 1
1	Primera adición de extensión presente
00000010	Longitud de la codificación de adición de extensión = 2
1	Bitmap = 1 indica 'h' presente
0010 0011 0100	g = "123"
1xx	h = VERDADERO (TRUE)

#### A.4.4 Representación UNALIGNED PER de este valor de la ficha

A continuación se muestra la representación del valor mencionado (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 por una descripción comentada de la codificación presentada en binario. Obsérvese que los bits de relleno no ocurren en la variante UNALIGNED, salvo posiblemente al final de la codificación del valor más externo y, por lo tanto, implícitamente al final del valor transportado por un tipo abierto.

La longitud de esta codificación es 8 octetos. Para comparación, el mismo valor codificado mediante la variante ALIGNED de PER ocupa 8 octetos, codificado mediante BER con la forma de longitud definida ocupa por lo menos 22 octetos y codificado mediante BER con la forma de longitud indefinida ocupa por lo menos 26 octetos.

##### A.4.4.1 Presentación hexadecimal

9E000600 040A4690

##### A.4.4.2 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 octetos y una 'x' representa un bit cero utilizado para rellenar el octeto final hasta un límite de octeto.

1	Valores de adición de extensión presentes en Ax
00	Bitmap bits = 0 indica campos opcionales (i y j) ausentes
11	a = 253
1	b = VERDADERO (TRUE)
1	Valor de elección de c es un valor de adición de extensión
0.000000	Índice de elección selecciona c.e
00.000001	Longitud de c.e
1x.xxxxxx	c.e = VERDADERO (TRUE)
00.00000	Número de adiciones de extensión definidas en Ax = 1
1	Primera adición de extensión presente
00.000010	Longitud de la codificación de adición de extensión = 2
1	Bitmap = 1 indica 'h' presente
0.010 0011 0.100	g = "123"
1xxxx	h = VERDADERO (TRUE)

## Anexo B

### Combinación de restricciones visibles a las reglas de codificación compactadas (PER) y restricciones no visibles

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

#### B.1 Consideraciones generales

**B.1.1** La correcta determinación de la extensibilidad de las PER es fundamental para el interfuncionamiento de las aplicaciones. También es importante que las distintas aplicaciones determinen de la misma forma los valores que han de codificarse según las PER como valores raíz y los valores que se han de codificar como adiciones de extensión en el caso de un tipo extensible.

**B.1.2** Los usuarios suelen escribir códigos simples y la codificación PER es intuitiva, pero en el caso de construcciones complicadas, las interacciones entre la visibilidad a PER, la extensibilidad de PER y la aritmética de conjunto han de debatirse más a fondo y es el contenido de la presente cláusula.

**B.1.3** Dado que algunas restricciones no son visibles a PER (véase 9.3) un tipo puede definirse como extensible según lo dispuesto en la Rec. UIT-T X.680 | ISO/CEI 8824-1, pero considerarse no extensible (con restricciones laxas que pueden abarcar todas las extensiones posibles) para la codificación PER.

**B.1.4** Si un tipo se considera extensible en ambos casos, el conjunto de valores raíz para la codificación PER no es siempre idéntico al conjunto de valores que se considerarían valores raíz según las definiciones de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

**B.1.5** En la mayor parte de los casos reales, se trata de determinaciones sencillas y directas.

**B.1.6** No obstante, la notación ASN.1 es muy completa y adaptable para aplicar restricciones complejas resultantes de la aritmética de conjuntos y/o la aplicación en serie de restricciones simples o complejas.

**B.1.7** Es poco probable que las especificaciones de usuario definan construcciones ASN.1 tan complejas como las que tratamos en el presente anexo, pero los implementadores han de saber qué código producir en el caso de estas restricciones.

**B.1.8** Las normas para restricciones muy complejas (que conlleven quizás varios nombres de referencia de tipo) no son siempre intuitivas, pero se han diseñado para simplificar la aplicación de las herramientas así como la complejidad de la especificación ASN.1.

**B.1.9** Para **SEQUENCE**, **SET**, **CHOICE** y **ENUMERATED**, un tipo siempre es extensible si contiene el marcador de extensión (la elipsis "..."), incluso si está restringido (véase 9.3.20). Un valor es un valor raíz única y exclusivamente si el valor no incluye ningún elemento (o alternativas de **CHOICE** y enumeraciones para **ENUMERATED**) detrás de la elipsis. Un **SEQUENCE**, **SET**, **CHOICE** o **ENUMERATED** no extensible puede ser un tipo progenitor al que se aplica una restricción extensible y da como resultado un tipo secuencia, conjunto, elección o enumerado extensible. No obstante, las restricciones para estos tipos nunca son visibles a PER y los tipos resultantes se codifican sin el bit de extensibilidad en PER. Estos tipos no se tratan más a fondo en el presente anexo, que aborda solamente la extensibilidad que surge del uso de restricciones extensibles en tipos enteros y cadena de caracteres de multiplicador conocido restringido. (Las restricciones para otros tipos no afectan a las codificaciones PER, excepto las restricciones de tamaño en tipos cadena de octetos y cadena de bits, que son similares a las restricciones de tamaño en los tipos cadena de caracteres y no se tratan más en este anexo.)

**B.1.10** En la parte normativa del texto se especifican las normas propiamente dichas, y este anexo didáctico sólo pretende explicarlas mejor a los proveedores de herramientas.

**B.1.11** Para simplificar la explicación, los valores de un tipo no extensible o una restricción se describen más adelante como valores raíz, aunque este término estrictamente sólo se aplica a los tipos o restricciones extensibles.

**B.1.12** La cláusula G.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 proporciona información didáctica sobre la combinación de las restricciones cuando todas ellas son visibles a PER, como se especifica en dicha Recomendación | Norma Internacional, y conviene leerla en paralelo con este anexo. Si hay restricciones que no son visibles a PER, o se aplican restricciones a tipos cadena de caracteres, es necesario añadir algunas normas. Estas normas adicionales son las que se tratan en B.2.

## B.2 Extensibilidad y visibilidad de las restricciones en PER

### B.2.1 Consideraciones generales

**B.2.1.1** En BER, la codificación de los valores es idéntica para los valores raíz y para las adiciones de extensión, por lo que la extensibilidad no tiene repercusión alguna sobre ella. En PER, los valores abstractos generalmente se codifican de manera eficaz si forman parte del conjunto de valores raíz, que casi siempre es un conjunto finito, pero no necesariamente, y de manera menos eficaz si se trata de adiciones de extensión.

**B.2.1.2** No obstante, para muchas codificaciones PER, hay valores en las adiciones de extensión de un tipo (como se determina en la Rec. UIT-T X.680 | ISO/CEI 8824-1) que se codifican según PER como si fueran valores raíz y no adiciones de extensión. La identificación exacta de estos valores se realiza señalando que algunas restricciones "no son visibles a PER".

**B.2.1.3** El concepto de visibilidad a PER se introdujo en esta Recomendación | Norma Internacional para facilitar la tarea de los codificadores a la hora de determinar si un valor que ha de codificarse está o no en la raíz de un tipo extensible. Si el codificador no puede tratar eficazmente una restricción, se considerará que ésta "no es visible" a la codificación PER (es decir, que no tienen efectos sobre ella).

**B.2.1.4** Con una excepción, la visibilidad de una restricción simple depende únicamente del tipo restringido y/o de los aspectos de la restricción que no están relacionados con la extensibilidad. Por ejemplo, es necesario determinar si una restricción depende textualmente de una restricción de tabla o si es una restricción variable (dependiente textualmente de un parámetro de la sintaxis abstracta).

**B.2.1.5** Si se trata de una restricción variable o dependiente textualmente de una restricción de tabla, no será nunca visible a PER, independientemente de a qué tipo se aplique.

**B.2.1.6** Además, las restricciones nunca son visibles a PER a menos que se apliquen a un tipo entero o a un tipo cadena de caracteres de multiplicador conocido restringido (o se trata de restricciones de tamaño aplicadas a una cadena de bits o una cadena de octetos).

**B.2.1.7** La excepción es la restricción de alfabeto permitido aplicada a un tipo cadena de caracteres de multiplicador conocido restringido. Esta restricción es visible a PER única y exclusivamente si no es extensible.

**B.2.1.8** También es importante señalar que las restricciones de subtipo de valor único aplicadas a los tipos cadena de caracteres no son visibles a PER.

**B.2.1.9** En PER, las restricciones aplicadas a tipos cadena de caracteres tienen dos variantes independientes: las restricciones de tamaño de la cadena; y las restricciones de alfabeto permitido. Las primeras afectan a la presencia y a la forma de un campo longitud en la codificación, y las segundas afectan al número de bits utilizados para codificar cada carácter. En la utilización simple, está claro que una restricción especifica una u otra de estas características, por lo que:

```
A1 ::= VisibleString (SIZE (20))
-- A size constraint

A2 ::= VisibleString (FROM ("A".."F"))
-- A permitted-alphabet constraint

A3 ::= VisibleString (SIZE (2))(FROM ("A".."F"))
-- Both a size and a permitted-alphabet constraint
```

**B.2.1.10** No obstante, hay que tener en cuenta que:

```
B ::= VisibleString (SIZE (20) INTERSECTION FROM ("A".."F")
UNION
SIZE (3) INTERSECTION FROM ("F".."K") )
```

**B.2.1.11** Para especificar la codificación de los tipos con restricciones complejas de este tipo, PER introduce el concepto de restricción de tamaño *efectiva* y de restricción de alfabeto permitido *efectiva*. Estas dos restricciones permiten todos los valores abstractos en la raíz de la restricción propiamente dicha y, normalmente, algunos valores abstractos adicionales. En el ejemplo anterior, la restricción de tamaño efectiva es 3..20, y la restricción de alfabeto permitido efectiva es FROM("A".."K").

**B.2.1.12** Para tratar la extensibilidad, esta Recomendación | Norma Internacional especifica además que la restricción de tamaño efectiva o la restricción de alfabeto permitido efectiva, o ambas, pueden ser extensibles (la restricción de alfabeto permitido efectiva no será visible a PER y se ignorará al determinar las codificaciones), y es necesario considerar la repercusión de la (no) visibilidad a PER de las restricciones de alfabeto permitido extensibles en las restricciones efectivas aplicadas a un tipo.

**B.2.1.1.13** En las siguientes cláusulas se abordan las siguientes cuestiones principales: los efectos de la visibilidad a PER y el cálculo de las restricciones efectivas para la aplicación en serie de restricciones y para la aritmética de conjunto.

## B.2.2 Visibilidad a PER de las restricciones

**B.2.2.1** En B.2.2.10 se indica cuándo es visible a PER una restricción (compleja) completa y cuándo no, pero antes nos referiremos a la aplicación en serie de restricciones, siendo cada una de ellas (en su conjunto) visible o no visible a PER.

**B.2.2.2** La regla es muy simple: si una restricción completa en la aplicación en serie de restricciones no es visible a PER, simplemente no se tiene en cuenta a los efectos de las codificaciones PER.

NOTA – Si las restricciones no visibles se eliminan para definir las codificaciones PER, no implica que las aplicaciones puedan transmitir valores abstractos adicionales de forma válida. Las restricciones originales siguen aplicándose a los valores que pueden transmitirse, aunque generalmente los codificadores utilizarán únicamente restricciones visibles a PER para realizar comprobaciones y diagnósticos.

**B.2.2.3** Es importante saber que la eliminación de restricciones no visibles puede tener efectos graves en los casos complejos, y es siempre importante considerar la extensibilidad (y cuáles son los valores raíz) tras la eliminación de restricciones aplicadas en serie que no son visibles a PER. (Si ninguna de las restricciones aplicadas en serie es visible a PER, el tipo no está restringido, y no es extensible, a los efectos de las codificaciones PER.)

**B.2.2.4** Un tipo extensible de acuerdo con la Rec. UIT-T X.680 | ISO/CEI 8824-1 puede no ser extensible para PER.

**B.2.2.5** Incluso cuando los efectos no sean tan graves, los valores que son adiciones de extensión, de acuerdo con la Rec. UIT-T X.680 | ISO/CEI 8824-1, pueden ser parte de los valores raíz al eliminarse algunas restricciones, y, entonces, se codificarán en PER como valores raíz y no como adiciones de extensión.

NOTA – Esto significa que las codificaciones PER son más verbosas de lo teóricamente posible, pero siguen teniendo una codificación única para todos los valores abstractos en el tipo que se codifica.

**B.2.2.6** Hay tres factores principales en relación con la visibilidad de una restricción compleja cuando se aplica en serie.

**B.2.2.7** El primer factor que se debe considerar es si la restricción es variable (depende textualmente de un parámetro de la sintaxis abstracta), o si depende textualmente de una restricción de tabla. En estos casos, toda la restricción que se aplica en serie no es visible a PER y se descarta.

**B.2.2.8** El segundo factor vale únicamente para restricciones aplicadas a los tipos cadena de caracteres. Las restricciones de subtipo de valor único aplicadas a estos tipos no son visibles a PER, pero su presencia no implica que toda la restricción aplicada en serie no sea visible, si se incluye en ella una aritmética de conjunto.

**B.2.2.9** Las normas para determinar la visibilidad a PER en este caso se especifican en 9.3.19 y se resumen a continuación. "V" indicará la visibilidad a PER e "I" indicará la no visibilidad (invisibilidad).

**B.2.2.10** Dado que **UNION** e **INTERSECTION** son ambas conmutativas, la norma para el resultado se aplica únicamente al primer caso V. Cuando todos los componentes son V, se aplican las reglas normales de la Rec. UIT-T X.680 | ISO/CEI 8824-1, y no se trata más este tema. Los casos donde todos los componentes son I siempre dan I, y tampoco se tratan más adelante. Estas normas son:

```
V UNION I => I
V INTERSECTION I => V
-- The resulting V is just the V part of the intersection
V EXCEPT I => V
-- The resulting V is just the V without the set difference
I EXCEPT V => I
V, ..., I => I
I, ..., V => I
```

**B.2.2.11** Hay una consecuencia importante de eliminar así restricciones de subtipo de valor único (y las cláusulas **EXCEPT**). Significa que todas las restricciones "atómicas" que pueden aplicarse a un tipo cadena de caracteres son únicamente restricciones de tamaño o únicamente restricciones de alfabeto permitido. La restricción total está formada (únicamente) de intersecciones, uniones y adiciones de extensión (arbitrariamente complicadas) con esas unidades "atómicas".

**B.2.2.12** De este modo se simplifica en gran medida el cálculo de lo que en PER se denominan "restricciones efectivas" aplicadas a los tipos cadena de caracteres.

**B.2.2.13** El tercer factor principal es determinar si una restricción de alfabeto permitido es extensible. Estas restricciones no son visibles a PER, pero se les otorga un tratamiento distinto del expuesto, pues su presencia no afecta

a la visibilidad de ninguna de las restricciones de tamaño que puedan estar presentes. Este punto se debate más a fondo en B.2.3.

### B.2.3 Restricciones efectivas

**B.2.3.1** Todas las restricciones aplicadas a un tipo cadena de caracteres de multiplicador conocido equivalen a un par de restricciones efectivas: una restricción de alfabeto permitido efectiva y una restricción de tamaño efectiva. Cualquiera de ellas, o ambas, pueden ser extensibles o nulas (restricción no efectiva).

**B.2.3.2** En la aplicación en serie, la última restricción es la única en que una de estas dos puede ser extensible, de conformidad con las normas de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

**B.2.3.3** Las definiciones de restricción de tamaño efectiva y de alfabeto permitido efectiva se encuentran en 3.6.8 y 3.6.9 y no se repiten aquí, pero se aplican a un tipo en el que se han suprimido las restricciones "invisibles", como se especifica en B.2.2.9 y B.2.2.10.

**B.2.3.4** Al igual que con la eliminación de restricciones no visibles a PER, la sustitución de una restricción efectiva por una aplicación en serie de una restricción de tamaño efectiva y una restricción de alfabeto permitido efectiva añade nuevos valores abstractos para las codificaciones PER (se incluye cualquier valor cuyo tamaño esté dentro de la restricción de tamaño efectiva y que utiliza únicamente el alfabeto permitido efectivo). No obstante, estos valores nunca serán transmitidos por una aplicación conforme a las normas y el efecto que se obtiene es simplemente que la codificación PER es menos eficaz de lo que teóricamente debería ser.

#### B.2.3.5 Ejemplo

```
A ::= VisibleString ( SIZE(10) INTERSECTION FROM("A")
                     UNION
                     SIZE(20) INTERSECTION FROM("B") )
```

tiene sólo dos valores, por lo que una codificación de un bit es teóricamente posible, pero las codificaciones PER utilizan restricciones efectivas y pueden codificar (aproximadamente) un millón de valores en:

```
B ::= VisibleString ( SIZE (10 UNION 20)
                     INTERSECTION
                     FROM ("AB") )
```

**B.2.3.6** Las restricciones efectivas aplicadas a la unión de dos conjuntos de valores da como resultado siempre la unión de las restricciones efectivas aplicadas a cada uno de los conjuntos de valores, pero en el caso general (si todas las restricciones son visibles a PER) esta regla simple no abarca la intersección.

**B.2.3.7** No obstante, la eliminación de las restricciones de subtipo de valor único y de las cláusulas **EXCEPT** es importante en este caso. Cuando todas las restricciones "atómicas" son simplemente restricciones de tamaño o simplemente restricciones de alfabeto permitido (posiblemente extendido), las restricciones efectivas pueden calcularse para la aritmética de conjunto arbitrario (sin cláusulas **EXCEPT**) de manera simple.

**B.2.3.8** Supóngase que {S, A} representa el conjunto de todos los valores permitidos por una restricción de tamaño S aplicada en serie con una restricción de alfabeto permitido A. (Obsérvese aquí también que la unión y la intersección son conmutativas.) En este caso se obtiene que:

$$\begin{aligned} \{S1, A1\} \text{ INTERSECTION } \{S2, A2\} &=> \{S1 \text{ INTERSECTION } S2, \\ & \quad A1 \text{ INTERSECTION } A2\} \\ \{S1, A1\} \text{ UNION } \{S2, A2\} &=> \{S1 \text{ UNION } S2, \\ & \quad A1 \text{ UNION } A2\} \\ \{S1, A1\}, \dots &=> \{S1, \dots\} \end{aligned}$$

**B.2.3.9** Este último caso necesita una aclaración. Una restricción de alfabeto permitido extensible no tiene repercusiones sobre la codificación, puesto que PER no soporta un número distinto de bits para los caracteres que son necesarios para valores raíz y valores de adición de extensión. Así, si una restricción de alfabeto permitido (efectiva) es extensible, ya no se trata de una restricción, pues todos los caracteres han de tener una representación. El efecto de "..." en el último caso es que tanto el alfabeto permitido como el tamaño son extensibles, pero sólo el tamaño extensible sigue siendo una restricción. Esto se expresa en el texto normativo indicando que las restricciones de alfabeto permitido efectivas extensibles no son visibles a PER.

## B.3 Ejemplos

En esta cláusula se presentan algunos ejemplos a título ilustrativo.

```
A ::= INTEGER (MIN .. MAX, ..., 1..10)
-- A is extensible, but the root is unconstrained and the
-- extensibility bit is always set to zero
```

```

A1 ::= INTEGER (1..32, ..., 33..128)
-- A1 is extensible, and contains values 1 to 128 with 1 to 32 in the
-- root and 33 to 128 as extension additions

A2 ::= INTEGER (1..32, ..., 33..128) (1..128)
-- This is illegal, as 128 is not in the root of the parent
-- (see ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 46)

A3 ::= INTEGER ( (1..32, ..., 33..128) ^ (1..128) )
-- This is legal. A3 is extensible, and contains 1 to 32 in the root
-- and 33 to 128 as extensions

A4 ::= INTEGER (1..32) (MIN .. 63)
-- MIN is 1, and 63 is illegal

A5 ::= INTEGER ( (1..32) ^ (MIN..63) )
-- This is legal. MIN is minus infinity and A3 contains 1 to 32

A6 ::= INTEGER ( (1..64, ... , B) ^ (1..32) )
-- A6 always contains (only) the values 1..32, no matter what values
-- B contains, but is nonetheless formally extensible and PER will
-- encode all values in 5 bits, with an extensibility bit (always) set
-- to zero

A7 ::= INTEGER (1..32, ... , B) (1..256)
-- A7 is illegal, as the parent for (1..256) can never contain more
-- than 1 to 32 no matter what B contains

A8 ::= IA5String (SIZE(3..4) | SIZE(9..10))
-- A8 has an effective size constraint of SIZE(3..4|9..10)
-- PER will encode as if it were SIZE(3..10), using three bits
-- to encode the length field

A9 ::= IA5String (FROM ("AB") ^ SIZE(1..2) |
                 FROM ("DE") ^ SIZE(3) |
                 FROM ("AXE") ^ (SIZE(1..5) )
-- A9 has an effective size constraint of SIZE(1..5), and PER will
-- encode the length in three bits. It has an effective alphabet
-- constraint of FROM("ABDEX") and PER will encode each character
-- using three bits

A10 ::= IA5String (SIZE(1..4) | SIZE(5..10) ^
                 FROM("ABCD") | SIZE(6..10))
-- A10 has an effective size constraint of SIZE(1..10), but
-- the permitted alphabet consists of the entire IA5String alphabet

A11 ::= IA5String (SIZE(1..10) | FROM("A".."D"))
-- No size constraint, alphabet is the entire IA5String alphabet

A12 ::= IA5String (SIZE(1..10) ^ FROM("A".."D"), ...)
-- A12 has an extensible effective size constraint of SIZE(1..10,...)
-- and the alphabet is the entire IA5String alphabet

A13 ::= IA5String (SIZE(1..10, ...) ^ FROM("A".."D"))
-- A13 has an extensible effective size constraint of SIZE(1..10,...)
-- and an effective alphabet constraint of FROM("A".."D")

A14 ::= IA5String (SIZE(1..10, ...,29)) (FROM("A".."D"))
-- An effective size constraint of SIZE(1..10), not extensible,
-- because of the serial application of the FROM. Effective
-- alphabet constraint is FROM("A".."D")

A15 ::= IA5String (SIZE(1..10, ...) | FROM("A".."D"), ...)
-- An extensible effective size constraint, but from MIN to MAX, with
-- all values in the root, encoding with an extensibility bit always
-- set to zero. The alphabet is the entire IA5String alphabet

A16 ::= IA5String (FROM("A".."D") ^ SIZE(1..10), ...)
-- The effective size constraint is SIZE(1..10,...), extensible
-- The alphabet is the entire IA5String alphabet

A17 ::= IA5String (FROM("A".."D"), ...) (SIZE(1..10))
-- An effective alphabet constraint of FROM("A".."D"), not extensible,
-- because of the serial application of the SIZE. Effective size
-- constraint is SIZE(1..10)

```

## Anexo C

### Soporte de los algoritmos de las PER

(Este anexo no es parte integrante de esta 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** La codificación canónica se utiliza cuando se están aplicando características de seguridad a la codificación. La utilización de CANONICAL-PER puede complicar bastante el trabajo del procesador central si el valor que se ha de codificar incluye un tipo conjunto de, y generalmente no se recomienda para protocolos a menos que se requieran características de seguridad.

**C.2** Si 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** Se recomienda insistentemente que todas las implementaciones que soportan la decodificación de cualquier sintaxis de transferencia de la variante PER ALIGNED soporten la decodificación de la variante BASIC-PER ALIGNED (y por tanto de la variante CANONICAL-PER ALIGNED). Se aplica lo mismo para la variante UNALIGNED.

**C.4** Para facilitar el interfuncionamiento, se recomienda que todas las implementaciones de PER soporten la variante ALIGNED y la variante UNALIGNED (la implementación no es mucho más compleja). Es un asunto de gestión local ofrecer una u otra, o las dos, y también cuál de las dos variantes cuando se ofrecen ambas. Si sólo se ofrece una, ésta debe ser aceptada.

**C.5** La aceptación de estas recomendaciones es particularmente importante para los proveedores de herramientas para uso general. Cuando una implementación es específica de alguna aplicación particular, puede ser totalmente aceptable el soporte 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 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 el relajamiento o el refuerzo de restricciones para algunos componentes, probablemente cambien la codificación de valores del tipo en cuestión.

**D.2** Sin embargo, el objeto de estas reglas de codificación es garantizar el cumplimiento de las normas del modelo ASN.1 de extensión de tipo (véase la Rec. UIT-T X.680 | ISO/CEI 8824-1).

**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 esta Recomendación | Norma Internacional)

**E.1** Las codificaciones PER son autodelimitadoras, dado el conocimiento de las reglas de codificación y el tipo de 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 esta 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)"
```



## SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
<b>Serie X</b>	<b>Redes de datos, comunicaciones de sistemas abiertos y seguridad</b>
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet y Redes de la próxima generación
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación