Unión Internacional de Telecomunicaciones

UIT-T Z.104

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

(10/2004)

SERIE Z: LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

Técnicas de descripción formal – Lenguaje de especificación y descripción

Código de los datos del lenguaje de especificación y descripción

Recomendación UIT-T Z.104

Unión Internacional de Telecomunicaciones

RECOMENDACIONES UIT-T DE LA SERIE Z

LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

TÉCNICAS DE DESCRIPCIÓN FORMAL	
Lenguaje de especificación y descripción	**Z.100–Z.109**
Aplicación de técnicas de descripción formal	Z.110–Z.119
Gráficos de secuencias de mensajes	Z.120–Z.129
Lenguaje ampliado de definición de objetos	Z.130–Z.139
Notación de prueba y de control de prueba	Z.140–Z.149
Notación de requisitos de usuarios	Z.150–Z.159
LENGUAJES DE PROGRAMACIÓN	
CHILL: el lenguaje de alto nivel del UIT-T	Z.200–Z.209
LENGUAJE HOMBRE-MÁQUINA	
Principios generales	Z.300–Z.309
Sintaxis básica y procedimientos de diálogo	Z.310–Z.319
LHM ampliado para terminales con pantalla de visualización	Z.320–Z.329
Especificación de la interfaz hombre-máquina	Z.330–Z.349
Interfaces hombre-máquina orientadas a datos	Z.350–Z.359
Interfaces hombre-máquina para la gestión de las redes de telecomunicaciones	Z.360–Z.379
CALIDAD	
Calidad de soportes lógicos de telecomunicaciones	Z.400–Z.409
Aspectos de la calidad de las Recomendaciones relativas a los protocolos	Z.450–Z.459
MÉTODOS	
Métodos para validación y pruebas	Z.500–Z.519
SOPORTE INTERMEDIO	
Entorno del procesamiento distribuido	Z.600–Z.609

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

Recomendación UIT-T Z.104

Código de los datos del lenguaje de especificación y descripción

Resumen

La codificación de los datos del lenguaje de especificación y descripción (SDL, *specification and description language*) permite la comunicación de valores de datos entre partes del SDL de manera independiente de la implementación utilizada.

Los valores de datos pueden codificarse en el formato de texto que se define en la presente Recomendación. Alternativamente, cuando los datos están definidos en ASN.1, puede recurrirse a conjuntos de reglas de codificación ASN.1 normalizados.

Los resultados de la codificación serán utilizados internamente por el modelo SDL o para comunicar entre modelos SDL o entre elementos SDL y no SDL, como los entornos de prueba.

Orígenes

La Recomendación UIT-T Z.104 fue aprobada el 7 de octubre de 2004 por la Comisión de Estudio 17 (2001-2004) del UIT-T por el procedimiento de la Recomendación UIT-T A.8.

Palabras clave

ASN.1, código (codificación), datos, Rec. UIT-T Z.100, Rec. UIT-T Z.105, SDL.

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.

ÍNDICE

Introducción

La codificación define la manera en que los valores de datos se codifican cuando se comunica información entre partes del SDL.

En general, sólo pueden transmitirse los valores entre distintas partes del SDL y el SDL y su entorno. Por este motivo, la codificación definida por esta Recomendación sólo se aplica a los géneros valor de datos; no se soporta la codificación de géneros objeto de datos. Esto es compatible con la utilización de datos en la comunicación de señales SDL, donde se crean duplicados de datos de objeto (excepto en el caso en que una instancia de un proceso contiene tanto el emisor como el receptor).

Se prevé que la especificación de la codificación de los datos SDL se utilice normalmente en un canal que representa a una interfaz normativa, o para que los datos se encapsulen en otros elementos de datos para su tratamiento de manera transparente.

Cuando se especifica la codificación en un canal, ésta será transparente para el resto del modelo SDL, si los agentes receptores no utilizan la sintaxis adicional para las entradas que se dan en la Recomendación. Si ninguno de los extremos del canal conduce a un entorno SDL, la especificación de la codificación añade simplemente requisitos de codificación al resto del SDL y el código se elimina durante la entrada normal de las señales. Si el origen o el destino de la señal es un entorno SDL, la codificación establece un requisito de codificación de mensajes desde o hacia dicho entorno.

La encapsulación de datos en otro elemento de datos generalmente ocurre dentro de protocolos de capas, donde la comunicación entre dos capas puede corresponder a un canal entre dos agentes SDL. La codificación ofrece la posibilidad de simplificar los modelos SDL. En una capa, la señal de salida puede codificarse, pero cuando la señal es recibida desde el canal, puede ignorarse la codificación, de manera que pueda recibirse una señal de cualquiera de los tipos codificados. Los datos pueden almacenarse en la capa de receptor (o encapsularse para capas posteriores) y transmitirse a un par. Cuando los datos se pasan a la capa original, puede procederse a la decodificación y restauración de la señal original.

La codificación y la decodificación también se presentan como expresiones de datos, de manera que la encapsulación (y su procedimiento inverso) no han de formar parte del proceso de entrada y salida.

Se prevén otras utilizaciones de la codificación, principalmente para la codificación de texto, la cual podrá resultar útil para probar y validar modelos SDL, así como para disponer de una codificación independiente de ASN.1, aunque el conjunto de reglas de codificación de texto está previsto para transferir información en caracteres textuales a otras máquinas, y no para su lectura directa por humanos.

Dado que la codificación atañe principalmente a la comunicación, se relaciona con los datos transportados por conjuntos de señales así como con interfaces, puertas y canales. Se soporta específicamente la utilización del nombre ABSTRACT-SYNTAX ASN.1 en una interfaz para implicar un conjunto de señales basado en un tipo CHOICE de nivel superior en ABSTRACT-SYNTAX.

En esta Recomendación no se proporciona la especificación de la codificación de variables.

Recomendación UIT-T Z.104

Código de los datos del lenguaje de especificación y descripción

1 Alcance

Esta Recomendación amplía el SDL para proporcionar mecanismos de codificación de datos, de manera que los datos transmitidos entre distintas partes de un sistema SDL (o entre sistemas SDL) dispongan de una codificación independiente de la implementación.

Se proporciona un mecanismo que permite que la codificación se base en reglas distintas. Se define una codificación cuyo resultado es una cadena textual. Además, para los datos basados en definiciones ASN.1, o que pueden hacerse corresponder con definiciones ASN.1, puede utilizase el conjunto de reglas de codificación definido en la serie de Recs. UIT-T X.69x.

1.1 Utilización de codificación y decodificación

Dentro de una parte implementada independiente del SDL, no es necesario que se conozca la manera de representar los valores de datos y la implementación garantiza que los datos se comportan de la manera esperada.

Cuando la información se comunica entre partes del SDL implementadas separadamente, los valores de datos han de transmitirse utilizando una codificación que puedan procesar todas las partes del SDL que utilizan dicha información. Por ejemplo, si se transmite un valor booleano, éste puede transmitirse como un único bit y habrá de definirse si el bit cero representa verdadero o falso. Para que el valor booleano pueda comunicarse correctamente, tanto el emisor como el receptor habrán de aplicar el mismo conjunto de reglas de codificación. Antes de enviar el valor, o una vez que se ha recibido, puede haber buenos motivos para que la información se represente de distinta manera. Por ejemplo, en una parte de un sistema que soporta adecuadamente los enteros de 16 bits, y donde la capacidad de almacenamiento no es importante, pero el acceso a bit individuales es poco eficaz, el valor booleano puede representarse más adecuadamente con 16 bits, representando los 16 bits cero el valor falso y cualquier otro conjunto de bits, verdadero. En otra parte del sistema, el valor booleano puede aplicarse mejor en un único byte octeto.

La comunicación en el SDL se hace a través de canales, explícitos o implícitos. La comunicación de canal puede realizarse entre partes de un sistema SDL o entre partes de un sistema y su entorno.

La comunicación que tiene lugar en un canal corresponde a un protocolo o puede considerarse como tal. En general, el canal utilizado para la comunicación transporta distintas señales en cada sentido. Generalmente, las señales en un sentido difieren de las señales en el otro. Ocasionalmente, el canal transporta señales en un sentido únicamente, aunque no es lo más frecuente.

Un canal entre dos agentes transporta señales para un protocolo entre dichos dos agentes. Si se asume que puede recibirse cualquier tipo de señal en cualquier momento, los tipos de señal codificados en un sentido deben ser distintos de los demás tipos de señal en el mismo sentido, o el agente receptor no podrá distinguir una señal de otra. Dos instancias de una señal con los mismos datos de salida del mismo agente y en el mismo canal tendrán la misma codificación. Si el tipo de señal o los datos o el agente de origen difieren, la información codificada es distinta. Se incluye el agente de origen, dado que todas las instancias de señal transportan el pid del agente de origen como parámetro implícito. De otro modo, dos instancias de una señal con los mismos datos de salida en el mismo canal tendrían la misma codificación.

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al

efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

– 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.690 (2002) | ISO/CEI 8825-1:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de la regla de codificación básica, de las reglas de codificación canónica y de las reglas de codificación distinguida*.

– Recomendación UIT-T X.691 (2002) | ISO/CEI 8825-2:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación compactada*.

– Recomendación UIT-T X.693 (2001) | ISO/CEI 8825-4:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Reglas de codificación del lenguaje de marcaje extensible*.

– Recomendación UIT-T Z.100 (2002), *Lenguaje de especificación y descripción*.

– Recomendación UIT-T Z.105 (2003), *Lenguaje de especificación y descripción combinado con módulos de notación de sintaxis abstracta uno*.

– ISO/IEC 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set (UCS)*.

3 Definiciones

3.1 **decodificar** (proceso): Proceso de construcción de un valor de datos SDL a partir de una cadena textual o plantilla de bits que se supone es una codificación del valor de datos SDL que utiliza las mismas reglas para su codificación y decodificación.

3.2 **decodificación**: Resultado del proceso de decodificar.

3.3 **codificar** (proceso): El proceso de producción de una codificación.

3.4 **codificación**: Cadena de texto o plantilla de bits resultante de la aplicación de un conjunto de reglas de codificación a un valor de datos SDL.

3.5 **conjunto de reglas de codificación**: Uno de los conjuntos de reglas de codificación (ASN.1) definidos en las Recs. de la serie UIT-T X.69x o el conjunto de reglas de codificación de texto definido en la cláusula 10.7.1 y en el anexo A de la presente Recomendación; o el conjunto de reglas de codificación dependientes de la implementación/aplicación definido por el código de procedimiento invocado de acuerdo con esta Recomendación (véase 10.7).

4 Siglas

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

SDL Lenguaje de especificación y descripción (*specification and description language*)

5 Convenios

En esta Recomendación se utiliza la notación y los convenios de presentación de la Rec. UIT-T Z.100.

La presente Recomendación está organizada de manera que la numeración de las cláusulas 6 a 13 sigue la numeración de la Rec. UIT-T Z.100.

Cuando se define una regla de sintaxis abstracta o concreta en esta Recomendación con el mismo nombre de una regla de la Rec. UIT-T Z.100, la regla aquí prescrita sustituye la regla de la Rec. UIT-T Z.100. Cualquier condición de *Gramática*, *Semántica* o *Modelo* definida en una regla de la Rec. UIT-T Z.100, se aplica a la regla aquí redefinida, a menos que se especifique lo contrario.

Los caracteres se identifican por los nombres (en mayúsculas) que se les otorgan en ISO/CEI 10646.

6 Reglas generales

Esta Recomendación añade las palabras clave **encode** y **decode** adicionales a la regla léxica <keyword>.

6.1 Reglas léxicas

<keyword> ::=

abstract	**active**	**adding**
| **aggregation**	| **alternative**	| **and**
| **any**	| **as**	| **association**
| **atleast**	| **block**	| **break**
| **call**	| **channel**	| **choice**
| **comment**	| **composition**	| **connect**
| **connection**	| **constants**	| **continue**
| **create**	| **dcl**	| **decision**
| **decode**	| **default**	| **else**
| **encode**	| **endalternative**	| **endblock**
| **endchannel**	| **endconnection**	| **enddecision**
| **endexceptionhandler**	| **endinterface**	| **endmacro**
| **endmethod**	| **endobject**	| **endoperator**
| **endpackage**	| **endprocedure**	| **endprocess**
| **endselect**	| **endstate**	| **endsubstructure**
| **endsyntype**	| **endsystem**	| **endtype**
| **endvalue**	| **env**	| **exception**
| **exceptionhandler**	| **export**	| **exported**
| **external**	| **fi**	| **finalized**
| **from**	| **gate**	| **handle**
| **if**	| **import**	| **in**
| **inherits**	| **input**	| **interface**
| **join**	| **literals**	| **loop**
| **macro**	| **macrodefinition**	| **macroid**
| **method**	| **methods**	| **mod**
| **nameclass**	| **nextstate**	| **nodelay**
| **none**	| **not**	| **now**
| **object**	| **offspring**	| **onexception**
| **operator**	| **operators**	| **optional**
| **or**	| **ordered**	| **out**
| **output**	| **package**	| **parent**
| **priority**	| **private**	| **procedure**
| **protected**	| **process**	| **provided**
| **public**	| **raise**	| **redefined**
| **referenced**	| **rem**	| **remote**
| **reset**	| **return**	| **save**
| **select**	| **self**	| **sender**
| **set**	| **signal**	| **signallist**
| **signalset**	| **size**	| **spelling**
| **start**	| **state**	| **stop**
| **struct**	| **substructure**	| **synonym**
| **syntype**	| **system**	| **task**
| **then**	| **this**	| **timer**

to	**try**	**type**
use	**value**	**via**
virtual	**with**	**xor**

7 Organización de las especificaciones del SDL

7.1 Marco general

El marco general de las especificaciones del SDL es el definido en la Rec. UIT-T Z.100.

7.2 Lote

Modelo

Cuando se utiliza un módulo ASN.1 como lote y una <definition selection> en <package use clause> que se refiere al ASN.1, tiene la **interfaz** <selected entity kind> y el <name> es el nombre de un tipo de datos CHOICE, hay una interfaz implícita, que tiene el mismo nombre del CHOICE y define señales equivalentes a las alternativas CHOICE.

8 Conceptos estructurales

Los conceptos estructurales son los definidos en la Rec. UIT-T Z.100, además de la identificación opcional de un conjunto de reglas de codificación para las puertas y canales. En cualquier otro caso, los conceptos estructurales son los definidos en la Rec. UIT-T Z.100.

8.1 Tipos, instancias y puertas

Esta Recomendación añade la identificación opcional de un conjunto de reglas de codificación para la definición de puertas.

8.1.1 Definiciones de tipo estructural

Las definiciones de tipo estructural son las definidas en la Rec. UIT-T Z.100.

8.1.2 Expresión de tipo

Las expresiones de tipo son las definidas en la Rec. UIT-T Z.100.

8.1.3 Definiciones basadas en tipos

Las definiciones basadas en tipos son las definidas en la Rec. UIT-T Z.100.

8.1.4 Tipo abstracto

Los tipos abstractos son los definidos en la Rec. UIT-T Z.100.

8.1.5 Puerta

Las puertas pueden tener un conjunto de reglas de codificación. La salida de una señal de un agente a través de la puerta se codifica como se especifica en el conjunto de reglas de codificación. La información recibida a través de la puerta se decodifica de acuerdo con el conjunto de reglas de codificación. Si no se establece un conjunto de reglas de codificación específico, la codificación no está definida en la especificación SDL.

Gramática abstracta

Gate-definition :: *Gate-name*
[*Encoding-rules*]
*In-signal-identifier-**set***
*Out-signal-identifier-**set***

Si hay un conjunto de *Encoding-rules*, el *In-signal-identifier-**set*** y el *Out-signal-identifier-**set*** no contendrán ninguna señal implícita para los procedimientos remotos o variables remotas.

Si un canal externo con un conjunto de *Encoding-rules* (reglas de codificación) está conectado a una puerta de un agente o estado compuesto, las *Encoding-rules* de la *Gate-definition* (definición de puerta) de la puerta serán idénticas a las *Encoding-rules* del canal. Habrá como mucho un canal de este tipo conectado a la puerta y las señales transportadas en un sentido para esta puerta serán idénticas a las señales transportadas en el sentido correspondiente en el canal.

Si un canal interno de un agente o un tipo de agente está conectado a la puerta del agente o el tipo de agente que tiene una *Gate-definition* con un conjunto de *Encoding-rules*, las *Encoding-rules* del canal serán idénticas. Las señales transportadas en un sentido para la puerta serán idénticas a las señales transportadas en el sentido correspondiente en el canal. Podrá haber, más de un canal interno de este tipo.

Gramática concreta

<gate definition> ::=
{ <gate symbol> | <inherited gate symbol> }
is associated with { <gate> [<encoding rules>]
[<signal list area>] [<signal list area>] } ***set***
[***is connected to*** <endpoint constraint>]

NOTA 1 – Cuando una puerta en un subtipo es una extensión de la puerta heredada de un supertipo, se utiliza <inherited gate symbol> en la sintaxis concreta.

NOTA 2 – Si es necesaria una codificación distinta en cada sentido, la comunicación deberá estar especificada por dos puertas, que transportan cada una de ellas señales en un sentido.

<interface gate definition> ::=
<gate symbol 1>
is associated with { <interface identifier> [<encoding rules>] }

Una especificación de <encoding rules> que está asociada con un <inherited gate symbol> especificará el mismo conjunto de *Encoding-rules* que las *Encoding-rules* de la definición de puerta correspondiente en el supertipo, si dicha puerta tiene *Encoding-rules*. Si no hay un conjunto de <encoding rules> asociado con un <inherited gate symbol>, y hay un conjunto de *Encoding-rules* para la puerta en el supertipo, la puerta heredada tiene este mismo conjunto de *Encoding-rules*. Si no hay un conjunto de *Encoding-rules* para la puerta en el supertipo, la presencia y valor del conjunto de *Encoding-rules* de la puerta heredada está determinado por la presencia y el valor de <encoding rules>.

Semántica

El conjunto de *Encoding-rules* de una *Gate-definition* de un tipo de agente o tipo de estado compuesto corresponde al conjunto de *Encoding-rules* del canal en el ámbito circundante de las (conjunto de) especificaciones de la instancia. La codificación utilizada en el comportamiento del tipo puede determinarse independientemente del canal real al que está conectada la puerta de la instancia.

Modelo

El conjunto de *Encoding-rules* de una *Gate-definition* de un tipo de agente implícito de un <agent diagram> (o el tipo de estado compuesto implícito de una <composite state area>) es idéntico al conjunto de *Encoding-rules* del canal externo (explícito o implícito) del que se deriva la *Gate-definition*. No hay *Encoding-rules* en esta *Gate-definition* si no las hay en el canal externo.

El conjunto de *Encoding-rules* de una *Gate-definition* implícita de un *Agent-type-definition* de un sistema (definido por un <system type> o implícito en un <system diagram>) es idéntico al conjunto de *Encoding-rules* de un canal interno (explícito o implícito) del que se deriva la *Gate-definition*. No hay *Encoding-rules* en esta *Gate-definition* si no las hay en el canal interno.

Si se da una <gate on diagram> explícita para un <system type diagram> o <system diagram>, el conjunto de *Encoding-rules* de la *Gate-definition* correspondiente de la *Agent-type-definition* está

determinado por las <encoding rules> de la <gate on diagram>, si hay <encoding rules>. Si no las hay, el conjunto de *Encoding-rules* se determina a partir del canal interno del mismo modo que la *Gate-definition* implícita.

8.2 Parámetros de contexto

Los parámetros de contexto (incluidos los parámetros de contexto de puerta) son los definidos en la Rec. UIT-T Z.100.

NOTA – Cuando una puerta en un tipo parametrizado está definida por un parámetro de contexto formal, el conjunto de reglas de codificación de la puerta en el tipo especializado utilizado para definir estas instancias estará definido por la definición real de puerta identificada por el parámetro de contexto real.

8.3 Especialización

La especialización es la definida en la Rec. UIT-T Z.100.

8.4 Referencias de tipo

Las referencias de tipo son las definidas en la Rec. UIT-T Z.100.

NOTA – Una <gate property area> en una referencia de tipo es una <gate definition> o una <interface gate definition>, por lo que puede contener una especificación de <encoding rules> que debe ser compatible con la definición dada en el tipo.

8.5 Asociaciones

Las asociaciones son las definidas en la Rec. UIT-T Z.100.

9 Agentes

Los agentes son los definidos en la Rec. UIT-T Z.100.

10 Comunicación

La codificación de datos amplía las definiciones de canales y conexiones.

Se define la gramática de las reglas de codificación.

10.1 Canal

Un canal que conecta dos agentes determina la codificación (de haberla) que ha de utilizarse para la comunicación entre ellos. Un canal conectado al entorno de un agente tiene la codificación definida (de haberla) para la puerta que conecta con el entorno.

Gramática abstracta

Channel-definition :: *Channel-name*
[*Encoding-rules*]
[**NODELAY**]
*Channel-path-**set***

La *Originating-gate* (puerta de origen) o la *Destination-gate* (puerta de destino) tendrán las mismas *Encoding-rules* que la *Channel-definition* (definición de canal). Si la *Channel-definition* no tiene *Encoding-rules*, ni la *Originating-gate* ni la *Destination-gate* tendrán *Encoding-rules*.

Gramática concreta

<channel definition area> ::=
<channel symbol>
is associated with
{ [<channel name> [<encoding rules>]]
{ [<signal list area>] [<signal list area>] } ***set*** }

is connected to {
{ <agent area> | <state partition area> | <gate on diagram> }
{ <agent area> | <state partition area> | <gate on diagram> } } ***set***

NOTA 1 – Si es necesaria una codificación distinta para cada sentido, la comunicación deberá estar especificada por dos canales que transporten, cada uno de ellos, señales en un sentido.

Semántica

Las *Encoding-rules* de una *Channel-definition* conectada a un conjunto de especificaciones de instancia se utiliza en el comportamiento de las instancias.

Modelo

Si se omiten las <encoding rules> y el <channel symbol> está conectado a una <gate on diagram> con <encoding rules>, la *Channel-definition* tendrá las mismas *Encoding-rules* que la *Gate-definition* de la <gate on diagram>. Si la *Gate-definition* no tiene *Encoding-rules,* la *Channel-definition* tampoco las tiene.

Los canales implícitos no tienen *Encoding-rules* si las puertas a las que están conectados no tienen *Encoding-rules*. En cualquier otro caso, las *Encoding-rules* de un canal implícito son idénticas a las *Encoding-rules* de cada una de las puertas.

NOTA 2 – Si el canal está conectado al borde del diagrama circundante y no hay <gate on diagram>, esto representa una conexión.

10.2 Conexión

Modelo

Las *Encoding-rules* de una puerta implícita de una conexión son idénticas al canal externo conectado. Un canal sin <encoding rules> (o un canal implícito) conectado a una puerta que tiene *Encoding-rules*, tiene las mismas *Encoding-rules*.

10.3 Señal

Las señales son las definidas en la Rec. UIT-T Z.100.

10.4 Definición de lista de señales

La lista de señales se define en la Rec. UIT-T Z.100.

10.5 Procedimientos remotos

Los procedimientos remotos son los definidos en la Rec. UIT-T Z.100.

10.6 Variables remotas

Las variables remotas son las definidas en la Rec. UIT-T Z.100.

10.7 Reglas de codificación del trayecto de comunicación, codificar y decodificar

El conjunto de reglas de codificación especifica qué conjunto de las mismas se utiliza para codificar y decodificar los datos transportados por un canal o puerta concreto.

Gramática abstracta

Encoding-rules	::	*Rules-identifier*
Encoding-expression	::	*Signal-identifier* [*Expression*]* *Encoding-path*
Encoding-path	::	{ *Channel-identifier* | *Gate-identifier* }

Decoding-expression :: *Expression*
Encoding-path

Rule-identifier :: *Literal-identifier*

El *Rule-identifier* será uno de los identificadores literales del tipo de datos `Encoding` (véase *semántica* más adelante). Si la regla real identificada corresponde a una codificación definida en las Recs. de la serie UIT-T X.69x, el conjunto de señales transportadas por un *Encoding-path* corresponderá a los elementos de un tipo CHOICE de ASN.1 que se transporta en el canal. El *Encoding-path* y el tipo CHOICE de ASN.1 corresponden en un sentido, si cada nombre de señal corresponde a un nombre CHOICE y para cada una de las señales el (único) parámetro de la señal es idéntico al del tipo de datos del CHOICE correspondiente.

El tipo de datos `Encoding` será el tipo de datos `Encoding` predefinido o un tipo de datos con el nombre `Encoding` que sea una especialización (directa o indirecta) del tipo de datos `Encoding` predefinido. La especialización sólo añadirá nombres literales al tipo de datos `Encoding` predefinido y no modificará otras propiedades.

Si el *Rule-identifier* corresponde (por *Nombre*) a un `Encoding` literal definido en el **`package`** (lote) `Predefined`, se recurrirá a los procedimientos incorporados implícitos en el conjunto de reglas de codificación normalizado (y se ignoran otros procedimientos, incluso si son visibles y tienen nombres correspondientes como se indica más adelante).

Si el *Rule-identifier* corresponde a un literal adicional añadido a la especialización del tipo de datos `Encoding` predefinido, habrá un procedimiento visible con la firma adecuada para cada invocación (implícita o explícita) de *Encoding-expression* o *Decoding-expression*.

El nombre del procedimiento para codificar es el nombre `encode` concatenado con el nombre de un literal `Encoding` adicional (por ejemplo, `encodemyprotocol`, donde el literal `Encoding` adicional es `myprotocol`). Este procedimiento tendrá un parámetro del tipo elección (choice) implícito para el trayecto correspondiente a la invocación (véase la *semántica* más adelante). Este procedimiento devolverá una cadena de caracteres (charstring), una cadena de bits (bitstring) o una cadena de octetos (octetstring).

El nombre del procedimiento para decodificar es el nombre `decode` concatenado con el nombre de un literal `Encoding` adicional (por ejemplo, `decodemyprotocol`, donde el literal `Encoding` adicional es `myprotocol`). Este procedimiento tendrá un parámetro del mismo tipo (cadena de caracteres, cadena de bits o cadena de octetos) que el procedimiento correspondiente para codificar, y devolverá el tipo elección para el trayecto pertinente a la invocación (véase la *semántica* más adelante).

Cada procedimiento codificar o decodificar será funcional (es decir, no contendrá estados y no modificará el valor de ninguna variable SDL externa al procedimiento cuando se interprete).

La longitud de la lista de *Expression*s opcional será idéntica al número de *Sort-reference-identifier*s en la *Signal-definition* indicada por *Signal-identifier*.

Cada *Expression* de una *Encoding-expression* será compatible con el género del *Sort-identifier-reference* correspondiente (por posición) en la *Signal-definition* indicada por el *Signal-identifier*.

Para un *Channel-identifier* de un *Encoding-path* de una *Encoding-expression* de un agente, el canal de este trayecto será alcanzable con el *Signal-identifier* del agente, y el *Channel-path* en el sentido desde el agente debe incluir el *Signal-identifier* en su conjunto de *Signal-identifier*s.

Para un *Gate-identifier* de un *Encoding-path* de una *Encoding-expression,* de un agente, la puerta deberá ser una puerta del agente o ser alcanzable a través de un canal con un *Signal-identifier* del agente, y el *Out-signal-identifier-**set*** de la puerta deberá incluir el *Signal-identifier*.

La *Expression* en una *Decoding-expression* será compatible con el género generado por una *Encoding-expression* utilizando el mismo *Encoding-path* en un contexto donde el contexto de la *Decoding-expression* es alcanzable a través del *Encoding-path*.

Gramática concreta

<encoding rules> ::=
encode <rules identifier>

<rules identifier> ::=
<literal>

<encoding expression> ::=
encode { <signal identifier>[(<actual parameters>)] | <expression> }
[<encoding path>]

<encoding path> ::=
as { <channel identifier> | <gate identifier> }

<decoding expression> ::=
decode <expression> [<encoding path>]

Se utiliza el conjunto de reglas de codificación del trayecto identificado por <channel identifier> o <gate identifier>.

Si una <encoding expression> contiene una <expression> (en vez de una señal), la señal real se deriva de la <expression> descrita en el modelo que se expone a continuación. El género de la <expression> será el género del tipo de datos de elección implícito correspondiente al conjunto de señales para <encoding path>.

Sólo se omitirá el <encoding path> de la <encoding expression> si hay exactamente un trayecto con codificación para salida de la señal en el contexto de la <encoding expression>, y en este caso se utiliza la codificación para dicho trayecto.

Sólo se omitirá el <encoding path> de la <decoding expression> si hay exactamente un trayecto con codificación para entrada en el contexto de la expresión de decodificación, y en este caso se utiliza la codificación para dicho trayecto.

Un <encoding path> utilizado como <sort> indica los datos implícitos definidos para el trayecto, como se define a continuación.

Semántica

El elemento *Encoding-rules* determina el conjunto de reglas utilizadas para modificar la codificación dependiente de la implementación para los datos internos

– a una codificación independiente de la implementación normalizada (para uno de los literales `Encoding` de tipo de datos `text` a `EXER`),
– o a una codificación de implementación o aplicación definida (para un literal añadido a una especialización del tipo de datos `Encoding`).

Se recurre a la codificación cuando una señal sale a través de un trayecto con un conjunto de reglas de codificación especificado y se invoca el procedimiento codificar correspondiente. Cuando una señal entra de dicho trayecto, se invoca el procedimiento de decodificación de los datos en la codificación interna recurriendo al procedimiento correspondiente. Por tanto, los procedimientos codificar y decodificar no repercuten en la semántica del SDL, como se define en la Rec. UIT-T Z.100, pero requieren una codificación específica de las señales para trayectos específicos y, así, se permite la implementación separada de distintas partes del sistema.

Cuando se interpreta una *Encoding-expression* o *Decoding-expression* se invoca el procedimiento correspondiente.

Se recurre al procedimiento codificar relativo a un conjunto de reglas de codificación para un trayecto en una *Encoding-expression* para obtener una cadena de caracteres, una cadena de bits o

una cadena de octetos, dependiendo del contexto y del conjunto de reglas de codificación utilizado. La cadena de caracteres, cadena de bits o cadena de octetos obtenida al codificar se decodifica gracias a la *Decoding-expression* utilizando el mismo conjunto de reglas de codificación para el mismo trayecto. Se utiliza el conjunto de reglas de codificación del trayecto de codificación identificado por *Channel-identifier* o *Gate-identifier* o *Interface-identifier*.

Para una *Encoding-expression* los datos se codifican como si fueran a salir por dicho trayecto. El resultado es un tipo de datos correspondiente al conjunto de reglas de codificación de dicho trayecto.

Para una *Decoding-expression*, los datos se decodifican como si hubiesen llegado por el trayecto especificado. El resultado es la expresión correspondiente al tipo de datos implícito para entrada por dicho canal en el contexto de decodificación, como se indica más abajo. Si falla la decodificación, se genera la excepción InvalidReference.

Para un trayecto con codificación, se define implícitamente el tipo de datos que corresponde a SDL:

```
value type Implicitname /* an implicit and unique name */
{ choice
         signal1 value
         { struct
            1 Sort11 optional;
            2 Sort12 optional;
            3 Sort13 optional;
            /* ... and so on for each parameter of signal1 */
          } ;
         signal2 value
         { struct
            1 Sort21 optional;
            2 Sort22 optional;
            3 Sort23 optional;
            /* ... and so on for each parameter of the signal2 */
          } ;
         signal3 NULL;  /* no parameters */
         /* ... and so on for each signal */
}
```

donde,

`signal1`, `signal2`, etc., son los nombres de las señales transportadas por el trayecto, y

`Sort11`, `Sort12`, etc., son tipos de datos correspondientes a los parámetros de `signal1`, y

`Sort21`, `Sort22`, etc., son los tipos de datos correspondientes a los parámetros de `signal2`.

Para una señal sin parámetros, el tipo de datos es `NULL` predefinido.

Si el trayecto es bidireccional y se transporta la señal en ambos sentidos, sólo hay una posibilidad de tipo de datos implícito para dicha señal.

El identificador implícito de este tipo de datos se denomina como el **as** <channel identifier> o <gate identifier> para el trayecto, de manera que la declaración de variable legal es:

```
dcl message as user_input;
```

donde `user_input` es el nombre del canal o la puerta con codificación y la asignación válida es:

```
message := decode encoded_value as user_input;
```

Los siguientes tipos de datos enumerados para el conjunto de reglas de codificación normalizado se añadirán al lote predefinido, como se define en D.3/Z.100 para soportar la codificación de datos SDL:

```
value type Encoding
     { literals text, BER, CER, DER, APER, UPER, CAPER, CUPER, BXER, CXER, EXER }
```

que se utiliza para denominar el conjunto de reglas de codificación requerido, de la siguiente manera:

`text` para el conjunto de reglas de codificación de texto de esta Recomendación, que produce una cadena de caracteres;

`BER` para el conjunto de reglas de codificación básica de ASN.1, que produce una cadena de octetos;

`CER` para el conjunto de reglas de codificación canónica de ASN.1, que produce una cadena de octetos;

`DER` para el conjunto de reglas de codificación distinguida de ASN.1, que produce una cadena de octetos;

`APER` para la variante alineada básica de las reglas de codificación compactada de ASN.1, que produce una cadena de octetos;

`UPER` para la variante no alineada básica de las reglas de codificación compactada de ASN.1, que produce una cadena de bits;

`CAPER` para la variante alineada canónica de las reglas de codificación compactada de ASN.1, que produce una cadena de octetos;

`CUPER` para la variante no alineada canónica de las reglas de codificación compactada de ASN.1, que produce una cadena de bits;

`BXER` para la variante básica de las reglas de codificación XML de ASN.1, que produce una cadena de caracteres;

`CXER` para la variante canónica de las reglas de codificación XML de ASN.1, que produce una cadena de caracteres;

`EXER` para la variante extendida de las reglas de codificación XML de ASN.1, que produce una cadena de caracteres.

Se añade el sinónimo PER al **package** predefinido como alternativa para APER, de la siguiente manera:

```
synonym PER Encoding = APER;
```

El operador `last` del tipo de datos `Encoding` se redefine como un nombre desconocido, por lo que no puede accederse a él, de manera que la aplicación e implementación pueden extender el tipo de datos `Encoding` sin cambio alguno únicamente cuando se utilicen las normas antes mencionadas. El tipo de datos `Encoding` puede especializarse añadiendo literales adicionales.

Modelo

Si una <encoding expression> contiene una <expression>, la elección actual se determina a partir de la expresión y la señal con el mismo nombre con que la elección sale con los valores de los parámetros de señal establecidos en la expresión.

10.7.1 Conjunto de reglas de codificación de texto

El conjunto de reglas de codificación de texto se presenta de manera que la información pueda transmitirse en trayectos de comunicación mediante cadenas de texto entre elementos en el sistema y entre el sistema y su entorno. No se define la codificación de los caracteres. Aunque generalmente las cadenas de texto son legibles por las personas, no es éste el objetivo de las reglas de codificación de texto.

Semántica

Si el conjunto de reglas de codificación se especifica como `text`, el resultado de una codificación es una `Charstring`.

La cadena real se determina de la siguiente manera:

LOS CORCHETES DE APERTURA y DE CIERRE ({ }) delimitan los valores de los tipos de datos y muestran dónde empiezan y terminan los valores, excepto cuando van incluidos en la codificación de una `Charstring`, en cuyo caso representan los caracteres <left curly bracket> o <right curly bracket> reales de la Rec. UIT-T Z.100;

La COMA (,) se utiliza para delimitar los elementos de una lista (por ejemplo en una codificación **struct**);

En cualquier otro caso, la cadena de caracteres real va determinada por cada tipo de datos, como se define a continuación, y se ilustra como 'Generated `CharString`' en los ejemplos.

Una señal completa se codifica como una lista de valores y se trata como una codificación **choice** del tipo de datos implícito del trayecto con codificación.

En el anexo A pueden encontrarse los detalles de la codificación de texto.

10.7.2 Conjuntos de reglas de codificación normalizada de las Recs. de la serie X.69x

La utilización de uno de los nombres `BER`, `CER`, `DER`, `APER`, `UPER`, `CAPER`, `CUPER`, `BXER`, `CXER` o `EXER` (correspondiente a los conjuntos de reglas de codificación de las Recs. de la serie X.69x, véase la cláusula 10.7 sobre *Semántica* anterior), tan sólo se especificará si las señales transportadas por el trayecto se definen como un tipo de datos CHOICE de ASN.1. Por consiguiente, los valores se codifican de acuerdo con este tipo de datos tratado como un ABSTRACT-SYNTAX de ASN.1.

11 Comportamiento

11.1 Arranque

El arranque es definido en la Rec. UIT-T Z.100.

11.2 Estado

Si durante la consideración de las señales en el puerto de entrada, la señal que se considera no corresponde a ninguna de las señales válidas para cualquiera de los estados del agente (por ejemplo, porque el mensaje recibido no puede decodificarse a una señal válida), se produce la excepción InvalidReference.

11.3 Entrada

Si se establece una <encoded input> para un trayecto (un canal o puerta), los mensajes que pueden recibirse a través de este trayecto se almacenan en la variable prevista en su forma codificada. Estas señales no pueden especificarse en ninguna otra entrada o guardarse para el mismo estado. Si las mismas señales pueden recibirse desde otro trayecto, se siguen asignando a la variable, cuyo género ha de ser compatible con la codificación del trayecto. Aunque el modelo que se presenta a continuación describe en primer lugar la decodificación de las señales y luego su codificación, se supone que las implementaciones reales optimicen este proceso copiando el valor codificado en la variable dada en la <encoded input>.

Gramática concreta

<input list> ::=
<stimulus> { , <stimulus> }*
| <asterisk input list>
| <encoded input>

<encoded input> ::=
encode <variable> [<encoding path>]

El <encoding path> sólo se omitirá de la <encoded input> si hay exactamente un trayecto con codificación que lleva al contexto de la entrada que tiene el conjunto de reglas de codificación que produce el género (cadena de caracteres, cadena de octetos o cadena de bits) de la <variable>.

Semántica

Si la señal especificada en la entrada se recibe a través de un canal que tiene un conjunto de reglas de codificación especificado, la señal se decodifica de acuerdo con dicho conjunto de reglas de codificación.

Modelo

Para cada una de las señales que puedan recibirse de un canal o puerta especificado en la <encoded input>, hay una entrada implícita que es equivalente a la asignación de valores transportados por la señal hacia las variables locales implícitas de los tipos adecuados, seguida de una tarea implícita. Ésta asigna a la <variable> el valor de la expresión de codificación para la señal y los valores recibidos de las variables implícitas utilizando el conjunto de reglas de codificación para el trayecto de la <encoded input>. Después de la tarea implícita, se interpreta la transición que sigue a la entrada codificada.

11.4 Entrada prioritaria

Semántica

Si la señal especificada en la entrada prioritaria se recibe a través de un canal que tiene especificado un conjunto de reglas de codificación, la señal se decodifica de conformidad con dicho conjunto.

11.5 Señal continua

La señal continua es la definida en la Rec. UIT-T Z.100.

11.6 Condición habilitadora

Semántica

Si la señal especificada se recibe a través de un canal que tiene especificado un conjunto de reglas de codificación, la señal se decodifica de conformidad con dicho conjunto.

11.7 Conservación

La conservación es la definida en la Rec. UIT-T Z.100.

11.8 Transición implícita

La transición implícita es la definida en la Rec. UIT-T Z.100.

11.9 Transición espontánea

La transición espontánea es la definida en la Rec. UIT-T Z.100.

11.10 Etiqueta

La etiqueta es la definida en la Rec. UIT-T Z.100.

11.11 Máquina de estado y estado compuesto

La máquina de estado y el estado compuesto son los definidos en la Rec. UIT-T Z.100.

11.12 Transición

La transición es la definida en la Rec. UIT-T Z.100.

11.13 Acción

11.13.1 Tarea

La tarea es la definida en la Rec. UIT-T Z.100.

11.13.2 Creación

La creación es la definida en la Rec. UIT-T Z.100.

11.13.3 Llamada de procedimiento

La llamada de procedimiento es la definida en la Rec. UIT-T Z.100.

11.13.4 Salida

Si se da una <expression output> para un trayecto (un canal o puerta), la expresión es un valor elección utilizado para derivar la señal que sale. El género elección corresponde al conjunto de señales para el trayecto.

Si se da una <encoded output> para un trayecto (un canal o puerta), la expresión dada se utiliza como señal que sale. En el modelo que se muestra a continuación, la expresión se decodifica para verificar la señal que se envía.

Gramática concreta

<output body> ::=
<output body item> {, <output body item> }*
<communication constraints>

<output body item> ::=
<signal identifier> [<actual parameters>]
<expression output>
| <encoded output>

<expression output> ::=
<expression>

<encoded output> ::=
encode <expression>

Cuando se utiliza una <expression output>, habrá exactamente un <via path> en <communication constraints> y el género de la <expression> de la <expression output> será el género del tipo de datos elección implícito correspondiente al conjunto de señales para este trayecto.

Cuando se utiliza una <encoded output>, habrá exactamente un <via path> en <communication constraints> y este <via path> especificará la puerta o canal que tiene el conjunto de reglas de codificación correspondiente al género (cadena de caracteres, cadena de octetos o cadena de bits) de la <expression> de la <encoding output>.

Semántica

Si la señal sale a través de un trayecto que tiene especificado un conjunto de reglas de codificación, la señal se codifica de conformidad con dicho conjunto de reglas de codificación.

Modelo

Si el <output body item> es una <expression output>, la elección presente está determinada por la expresión y la señal con el mismo nombre con que la elección sale con los valores de los parámetros de señal de dicha expresión.

Si el <output body item> es una <encoded output>, la señal que sale es la obtenida de la decodificación de los contenidos de la expresión de acuerdo con la regla de decodificación para señales recibidas que se envían por el trayecto especificado. La señal seleccionada es aquella con el mismo nombre que la elección presente en la decodificación. Si la señal no es válida para el

trayecto o falla la decodificación por cualquier motivo (por ejemplo, si la cadena de la expresión no corresponde de manera válida a una de las señales del trayecto), se genera la excepción OutOfRange y no se envía ninguna señal; en cualquier otro caso, el valor de la decodificación sale como una señal. Dado que la expresión ya está codificada como una cadena para el trayecto, el valor de cadena puede enviarse sin conversión alguna.

11.13.5 Decisión

La decisión es la definida en la Rec. UIT-T Z.100.

11.14 Lista de enunciados

La lista de enunciados es la definida en la Rec. UIT-T Z.100.

11.15 Temporizador

El temporizador es el definido en la Rec. UIT-T Z.100.

11.16 Excepción

La excepción es la definida en la Rec. UIT-T Z.100.

12 Datos

12.1 Definiciones de datos

Las definiciones de datos son las mismas de la Rec. UIT-T Z.100, excepto que <sort> se extiende con <encoding path>.

Gramática concreta

<sort> ::=
<basic sort> [(<range condition>)]
| <anchored sort>
| <expanded sort>
| <reference sort>
| <pid sort>
| <inline data type definition>
| <inline syntype definition>
| <encoding path>

12.2 Utilización pasiva de los datos

La sintaxis abstracta de las expresiones se amplía para incluir la codificación y la decodificación; en cualquier otro caso, la utilización pasiva de los datos es la definida en la Rec. UIT-T Z.100.

12.2.1 Expresiones

La codificación y la decodificación se añaden como expresiones.

Gramática abstracta

Active-expression = *Variable-access*
| *Conditional-expression*
| *Operation-application*
| *Equality-expression*
| *Imperative-expression*
| *Range-check-expression*
| *Value-returning-call-node*
| *State-expression*
| *Encoding-expression*
| *Decoding-expression*

Gramática concreta

<expression0> ::=
<operand>
| <create expression>
| <value returning procedure call>
| <encoding expression>
| <decoding expression>

12.3 Utilización activa de datos

La sintaxis abstracta de las expresiones se amplía para incluir la codificación y la decodificación; en cualquier otro caso, la utilización activa de los datos es la definida en la Rec. UIT-T Z.100.

13 Definición de sistema genérico

La definición de sistema genérico es la misma de la Rec. UIT-T Z.100.

Anexo A

Especificación del conjunto de reglas de codificación de texto

Los tipos de datos se presentan a continuación con los tipos de datos del anexo D/Z.100 en el orden en que ocurren, seguidos de otros tipos de datos.

A.1 Booleano

Los valores `Boolean` falso y verdadero se codificarán como LETRA MAYÚSCULA F y LETRA MAYÚSCULA T, respectivamente.

Ejemplo

```
dcl Var_Boolean Boolean;
task Var_Boolean := true;
```

Generated `CharString`: T

A.2 Carácter

Los valores `Caracter` se codificarán como un carácter real a excepción del carácter `ESC`, que se codifica con dos caracteres ESCAPE. Un valor de carácter no definido u omitido se codificará como un carácter ESCAPE seguido de un carácter NULL.

NOTA – La `CharString` generada puede contener caracteres no imprimibles.

Ejemplo

```
dcl Var_Character Character;
task Var_Character := 'M';
```

Generated `CharString`: M

A.3 Cadena

`String` tiene un parámetro para el género de elemento y se codificará como una lista de valores del género de elemento contenida entre un CORCHETE DE APERTURA y UNO DE CIERRE, separados por una COMA.

Ejemplo

```
value type IntString inherits String <Integer>;
dcl str IntString;

task str:= ''//mkstring(6)//mkstring(9)//mkstring(1948);
```

Generated CharString: {6,9,1948}

A.4 Cadena de caracteres, cadena IA5S, cadena numérica, cadena imprimible, cadena visible

Aunque Charstring se basa en una cadena al ser un tipo de datos predefinido que se utiliza comúnmente, se le otorga una codificación especial. IA5String y Charstring cubren la misma gama de caracteres y tienen la misma codificación. NumericString, PrintableString y VisibleString son subconjuntos de IA5String.

Estos tipos de cadenas se codificarán como una cadena de caracteres contenida entre caracteres APÓSTROFES (') sin cambios, excepto el <apostrophe> (') que se codificará como dos APÓSTROFES (').

NOTA 1 – Dentro de la cadena de caracteres <comma>s, <left curly brackets> y <right curly bracket>s se tratan como caracteres normales.

NOTA 2 – La Generated CharString podrá, por tanto, contener caracteres no imprimibles, incluyendo los caracteres fin de fila o fin de registro.

Ejemplo

```
dcl Var_Charstring Charstring;

task Var_Charstring := 'Fred''s world;
```

Generated CharString: 'Fred's world'

A.5 Entero

Integer se codificará como una notación de enteros decimales sin CEROS (0) al inicio y donde los números negativos van inmediatamente precedidos de un signo menos (–) sin CEROS (0) al inicio. El cero nunca se tratará como un número negativo.

Ejemplo

```
dcl i Integer;

task i := 2-7;
```

Generated CharString: –5

A.6 Natural

Natural es un **syntype** de Integer por lo que tendrá la misma codificación que Integer.

A.7 Real

El valor 0.0 se codificará como el NÚMERO CERO seguido de un PUNTO y de un NÚMERO CERO.

Los valores negativos se codificarán como un SIGNO MENOS inmediatamente seguido de la codificación del valor negativo (un valor real positivo).

Un valor real positivo se codificará como un único dígito entre 1 y 9 seguido de un PUNTO y de, al menos, 1 y hasta 11 fracciones decimales, seguido de la LETRA MINÚSCULA E ′e′, seguido de un exponente. El exponente es la codificación entera del valor exponencial: la potencia de base 10 que se aplica a la primera parte del número. El exponente puede ser negativo.

Pueden omitirse los ceros fraccionales de la cola. Algunos valores reales no pueden codificarse con precisión y es posible que valores reales distintos con ligeras diferencias entre ellos tenga la misma codificación. En cualquier caso, la exactitud de un valor real en una aplicación dependerá de su implementación. Por ejemplo, si el valor real 2.0/7.0 se almacena en realidad como la relación de dos enteros (2 y 7), la precisión es absoluta, mientras que una codificación más convencional podría ser 0.2857142857, que sólo es correcta hasta el décimo decimal.

Ejemplos

```
dcl p, q, r1, r2 Real := 2000.0, 7.0, 0, 0;

task r1:= p/q;
task r2:= q/p;
```

Generated `CharString` for r1: 2.85714285714e2

Generated `CharString` for r2: 3.5e-3

A.8 Array

`Array` tiene dos parámetros: un género índice y un género componente. Los géneros pueden ser, en principio, cualquier género, pero normalmente el género índice tiene un número finito de valores.

Cuando el género índice tiene un número finito de valores ordenados, un `Array` puede codificarse como una lista de valores de codificación de elementos, uno para cada elemento, separados por COMAS dentro de un único par de CORCHETES DE APERTURA Y DE CIERRE.

Ejemplo

```
value type ABC {literals A, B, C};
value type A1 inherits Array <ABC, Integer>;
dcl avalue, bvalue A1;

task avalue:= (. 3 .);
task bvalue[A]:= 3;
task bvalue[B]:= 5;
task bvalue[C]:= 7;
```

Generated `CharString` for avalue:{3,3,3}

Generated `CharString` for bvalue:{3,5,7}

Es posible que el índice no esté ordenado (es decir, que el operador "<" no esté definido, o que el género índice sea **structure** o **choice**). También es posible que el género índice no tenga un número finito de valores (es decir, el número posible de valores no es infinito, por ejemplo `CharString` o `Real`). Cuando el género índice no está ordenado o es infinito, el `Array` se codificará como el valor más frecuente seguido de pares de valores para cada elemento. Si dos o más valores ocurren con mucha frecuencia, se elige uno de ellos arbitrariamente como valor más frecuente. Cada par irá entre un CORCHETE DE APERTURA y uno DE CIERRE separados por una COMA. El primer valor es el valor de índice y el segundo, el valor de elemento. En los pares no se repetirán los valores de índice.

Ejemplo

```
value type Dehashing inherits Array <CharString, CharString> := (.''.);
/* note that the default value is an empty string */
dcl hashtable Dehashing;

task htable ('ac'):= 'action';
task htable ('ab'):= 'ability';
task htable ('zzzz'):= 'end of document';
```

Generated `CharString` for `htable`: {'',{'ab','ability'},{'ac','action'},{'zzzz','end of document'}}

A.9 Vector

El vector es un caso especial de array que siempre tiene un género índice que es un subconjunto de Natural en la gama de 1 a un valor máximo especificado y cualquier género de elemento. El vector, por tanto, se codificará de la misma manera que el array con un número finito de valores índice ordenados, es decir, una lista de valores de codificación de elementos, uno para cada elemento, separados por COMAS dentro de un CORCHETE DE APERTURA y uno DE CIERRE.

A.10 Conjuntista

El `Powerset` de un género representa un conjunto matemático cuyos elementos son todos miembros de dicho género. Cuando el género tiene un número finito de valores ordenados, el `Powerset` del género se codificará como una cadena de bits (una cadena de CEROS (0) y UNOS (1)) comprendidos entre APÓSTROFES ('). Cada posición de bit en esta cadena de bits indica si un valor del género está presente o ausente en el conjunto. Los UNOS (1) indican que el valor está presente y los CEROS (0) que está ausente. El bit más a la izquierda representa el valor más pequeño del género elemento y cada uno de los demás bits representa un valor del género elemento superior a los valores representados por los bits a su izquierda.

Ejemplo

```
value type Shortalpha {literals a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v};
/* note Shortalpha has 22 values */
value type Psa inherits Powerset<Shortalpha>;
dcl letters_used Psa;

task letters_used := (. h, c, u, r .);
```

Generated `CharString`: '00100001000000000010010'

Cuando el género no tiene un número finito de valores ordenados, el `Powerset` del género se codificará como la codificación de cada valor del elemento presente separado por COMAS entre un CORCHETE DE APERTURA y uno DE CIERRE. Los valores de elemento podrán estar en cualquier orden.

Ejemplo

```
value type Pchrstr inherits Powerset<Charstring>;
dcl strings_used Pchrstr;

task strings_used := (. 'me', 'you', 'us', 'me', 'again', 'hey', 'you' .);
```

Generated `CharString`: {'again','hey','you','me','us'}

A.11 Duración

`Duration` se utiliza para indicar "un intervalo de tiempo". A menos que se especifique lo contrario, el valor por defecto de la unidad de `Duration` es un segundo.

Los valores de duración se codificarán como un par de enteros separados por una COMA entre un CORCHETE DE APERTURA y uno DE CIERRE. El primer entero es el número de unidades y el segundo cualquier fracción en nano unidades (10^{-9}). Por defecto, se trata de segundos y nanosegundos. El valor puede ser negativo, en cuyo caso la codificación del valor negativo se utilizará insertando un SIGNO MENOS (–) inmediatamente antes del primer entero.

Ejemplo

```
dcl dvar Duration;

task dvar := -17.00000007;
```

Generated `CharString`: {-17,700}

A.12 Tiempo

`Time` se utiliza para indicar "un punto en el tiempo". El valor de la unidad de `Time` será idéntico al valor de la unidad de `Duration`. El origen de las unidades `Time` no está especificado en esta Recomendación, pero corresponde al punto cero del reloj del sistema, es decir, AHORA da el valor 0.0. Es posible que los valores de `Time` sean negativos.

Los valores de tiempo se codificarán de la misma manera que los de duración

Ejemplo

```
dcl dvar Duration;

task dvar := -17.00000007;
```

Generated `CharString`: {-17,700}

A.13 Bolsa

`Bag` se codificará de la misma manera que `Powerset` con un género elemento que no tenga un conjunto de valores ordenado finito, pero cada valor de elemento irá precedido por un entero seguido de DOS PUNTOS (:). El entero es el número de veces que el género elemento ocurre en el valor `Bag`.

Ejemplo

```
value type B1 inherits Bag <Integer>;

dcl Var_Bag B1;

task Var_Bag := (. 7, 4, 7 .);
```

Generated `CharString`: {2:7,1:4}

A.14 Bit, cadena de bit

`Bit` se codificará como un único CERO (0) o UNO (1) que representan a los bits cero y uno, respectivamente.

Ejemplo

```
dcl Var_Bit Bit;

task Var_Bit := 1;
```

Generated `CharString`: 1

`Bitstring` se codificará como una secuencia de bits representada por los caracteres CERO (0) y UNO (1) entre APÓSTROFES (').

Ejemplo

```
dcl Var_Bit Bit_String;

task Var_Bit := '01011'B;
```

Generated `CharString`: '01011'

A.15 Octeto, cadena de octetos

`Octet` se codificará como dos caracteres correspondientes a la notación hexadecimal de `Octet`. Los caracteres alfabéticos se representarán en letras minúsculas (es decir, de la LETRA MINÚSCULA A (a) a la LETRA MINÚSCULA F (f)).

Ejemplo

```
dcl oct Octet;

task oct := 62;
```

Generated `CharString`: 3e

`Octetstring` es una secuencia de valores `Octet` que se codificará como una cadena de pares de caracteres hexadecimales entre APÓSTROFES ('). Los caracteres alfabéticos se representarán en letras minúsculas.

Ejemplo

```
dcl os Octetstring;

task os:= '12B32D'H;
```

Generated `CharString`: '12b32d'

A.16 Pid, género pid

Para permitir la flexibilidad entre aplicaciones, los valores pid se codificarán como un valor CHOICE correspondiente a la definición de elección:

```
{ choice
      0 ApplicationDefined;
      1 Integer;
      2 OctetString;
      3 BitString;
      4 CharString;
      5 { struct
           identity CharString;
           instance Natural;
        };
}
```

Un valor de género pid se codificará como el correspondiente valor de género Pid.

Todos los valores pid se codificarán utilizando el mismo campo elección como se define anteriormente, es decir, si un valor pid en un modelo SDL está codificado utilizando el entero elección 1 (choice 1), todos los demás valores pid del mismo modelo se codificarán utilizando el entero de elección 1. En cualquier caso, la selección de las elecciones anteriores depende de la aplicación. La primera elección muestra la codificación definida por la aplicación que debe utilizarse. No se define en esta Recomendación el genero `ApplicationDefined`.

La misma instancia de agente de un modelo tendrá siempre la misma codificación de valor pid. Dos instancias de agente distintas tendrán siempre distintos valores pid codificados. En cualquier caso, la derivación de los valores pid codificados no se define en esta Recomendación.

Ejemplo (utilizando la elección 5)

```
dcl agent_id Pid; /* in second instance of process IPS */

task agent_id := self;
```

Generated `CharString`: {5, {IPS, 2}}

A.17 Nulo

El valor Nulo (véase la Rec. UIT-T Z.105) se codificará con un carácter CERO (0).

Ejemplo

```
dcl Var_Null Null;

task Var_Null := Null;
```

Generated `CharString`: 0

A.18 Enumerado (lista literal)

Un tipo enumerado definido por una lista literal o como un tipo ENUMERATED de ASN.1 se codificará como el valor natural dado por la aplicación del operador `num` del tipo de datos al literal que representa el valor que se codifica.

Ejemplo

```
value type Enum
{ literals e1, e2, e3;
};

dcl evar Enum;

task evar := e2;
```

Generated `CharString`: 1

A.19 Estructuras

El valor de tipo de datos estructura se codificará como una lista de valores para los campos dentro de un CORCHETE DE APERTURA y uno DE CIERRE separados por COMA (,).

Ejemplo

```
value type Record { struct
f1 Integer;
f2 Charstring;
f3 Integer;};
dcl locat Record;
task locat:= (. 17, 'mid-field', 230125 .);
```

Generated `CharString`: {1, 'mid-field', 230125}

A.20 Elección

Un valor elección se codificará como un par de valores separados por una COMA, entre un CORCHETE DE APERTURA y uno DE CIERRE. El primer valor es el campo seleccionado indicado por el nombre de la elección. El segundo valor es la `CharString` para el valor del campo de acuerdo con el tipo del campo.

Ejemplo

```
value type C {choice
cs CharString;
cb Boolean;
};

dcl ChoiceVar C;

task ChoiceVar.cb := true;
```

Generated `CharString`: {cb,T}

A.21 Herencia y syntype

Un **syntype** define un nuevo nombre para un tipo de datos con una constricción opcional en los valores fijados. La codificación de texto es, por tanto, idéntica a la del tipo de datos.

Un tipo de datos que hereda de otro tipo de datos tiene la misma codificación que el tipo de datos progenitor, aunque, si se añaden literales o campos adicionales, éstos se añaden a la codificación.

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
Serie X	Redes de datos, comunicaciones de sistemas abiertos y seguridad
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet y Redes de la próxima generación
Serie Z	**Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación**

Impreso en Suiza
Ginebra, 2005