NORMA
BRASILEÑA

ABNT NBR 15606-3

Segunda edición
29.11.2010

Válida a partir de
29.12.2010

Televisión digital terrestre Codificación de datos y especificaciones de transmisión para radiodifusión digital
Parte 3: Especificación

ICS 33.160.01

ISBN 978-85-07-02598-6

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referencia
ABNT NBR 15606-3:2010
86 páginas

ABNT
Av.Treze de Maio, 13 - 28º andar
20031-901 - Rio de Janeiro - RJ
Tel.: + 55 21 3974-2300
Fax: + 55 21 2220-1762
abnt@abnt.org.br
www.abnt.org.br

Índice

Página

Prefacio

La Associação Brasileira de Normas Técnicas (ABNT) es el Fórum Nacional de Normalización. Las Normas Brasileñas, cuyo contenido es responsabilidad de los Comités Brasileños (ABNT/CB), de los Organismos de Normalización Sectorial (ABNT/ONS) y de las Comisiones de Estudios Especiales (ABNT/CEE), son elaboradas por Comisiones de Estudio (CE), formadas por representantes de sus sectores implicados de los que forman parte: productores, consumidores y neutrales (universidades, laboratorios y otros).

Los Documentos Técnicos ABNT se elaboran de acuerdo con las reglas de Directivas ABNT, Parte 2.

La Associação Brasileira de Normas Técnicas (ABNT) llama la atención sobre la posibilidad de que algunos de los elementos de este documento pueden ser objeto de derechos de patente. La ABNT no debe ser considerada responsable por la identificación de cualesquiera derechos de patente.

La ABNT NBR 15606-3 fue elaborada por la Comisión de Estudio Especial de Televisión Digital (ABNT/CEE-85). El Proyecto circuló en Consulta Nacional según Edicto nº 09, de 06.09.2007 a 05.11.2007, con el número de Proyecto 00:001.85-006/3. El Proyecto circuló en Consulta Nacional según Edicto nº 09, de 03.09.2010 a 03.11.2010, con el número Proyecto de Enmienda ABNT NBR 15606-3.

En caso que surja cualquier duda con relación a la interpretación de la versión en español siempre deben prevalecer las prescripciones de la versión en portugués

Esta Norma está basada en los trabajos del Fórum del Sistema Brasileiro de Televisão Digital Terrestre, según establece el Decreto Presidencial nº 5.820, de 29/06/2006.

La ABNT NBR 15606, bajo el título general “Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital”, está previsto que contenga las siguientes partes:

— Parte 1: Codificación de datos;

— Parte 2: Ginga-NCL para receptores fijos y móviles – Lenguaje de aplicación XML para codificación de aplicaciones;

— Parte 3: Especificación de transmisión de datos;

— Parte 4: Ginga-J – Ambiente para la ejecución de aplicaciones procedurales;

— Parte 5: Ginga-NCL para receptores portátiles – Lenguaje de aplicación XML para codificación de aplicaciones.

Esta segunda edición incorpora la Enmienda 1 de 29.11.2010 y cancela y sustituye la edición anterior (ABNT NBR 15606-3:2007).

Esta versión en español es equivalente a la ABNT NBR 15606-3:2010, de 29.11.2010.

Esta versión en español fue publicada en 02.02.2011.

Televisión digital terrestre Codificación de datos y especificaciones de transmisión para radiodifusión digital
Parte 3: Especificación

1 Alcance

Esta parte de la ABNT NBR 15606 suministra una especificación de codificación y transmisión de datos para el esquema de transmisión digital.

Esta parte de la ABNT NBR 15606 se aplica a la transmisión de datos realizada como parte de la transmisión digital de datos.

2 Referencias normativas

Los documentos indicados a continuación son indispensables para la aplicación de este documento. Para las referencias fechadas, se aplican solamente las ediciones citadas. Para las referencias sin fecha, se aplican las ediciones más recientes del documento citado (incluyendo enmiendas).

ABNT NBR 15603-1, *Televisión digital terrestre – Multiplexación y servicios de información (SI) – Parte 1: SI del sistema de radiodifusión*

ABNT NBR 15603-2:2007, *Televisión digital terrestre – Multiplexación y servicios de información (SI) – Parte 2: Estructura de datos y definiciones de la información básica de SI*

ABNT NBR 15603-3, *Televisión digital terrestre – Multiplexación y servicios de información (SI) – Parte 3: Sintaxis y definiciones de información extendida del SI*

ABNT NBR 15606-1, *Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital - Parte 1: Codificación de datos*

ABNT NBR 15606-2:2007, *Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital – Parte 2: Ginga-NCL para receptores fijos y móviles – Lenguaje de aplicación XML para codificación de aplicaciones*

ABNT NBR 15602-3, *Televisión digital terrestre — Codificación de video, audio y multiplexación — Parte 3: Sistemas de multiplexación de señales*

ABNT NBR 15606-4:2010, *Televisión digital terrestre — Codificación de datos y especificaciones de transmisión para radiodifusión digital - Parte 4: Ginga-J - Ambiente para la ejecución de aplicaciones procedurales*

ABNT NBR 15607, *Televisión digital terrestre — Canal de interactividad — Parte 1: Protocolos, interfaces físicas e interfaces de software*

ISO 639-2, *Codes for the representation of names of languages – Part 2: Alpha-3 code*

ISO 8859-1, *Information processing - 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet Nº 1*

ISO/IEC TR 8802-1, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements - Part 1: Overview of Local Area Network Standards*

ISO/IEC 8802-2, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks - Specific requirements – Part 2: Logical link control*

ISO/IEC 8859-15, *Information technology - 8-bit single-byte coded graphic character sets — Part 15: Latin alphabet Nº 9*

ISO/IEC 13818-1, *Information technology – Generic coding of moving pictures and associated audio information: Systems*

ISO/IEC 13818-6:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSM-CC*

EN 300 468:2005, *Digital video broadcasting (DVB); specification for service information (SI) in DVB systems*

EN 301 192, *Digital video broadcasting (DVB); DVB specification for data broadcasting*

EN 301 193, *Digital Video Broadcasting (DVB); Interaction channel through the Digital Enhanced Cordless Telecommunications (DECT)*

EN 301 195, *Digital Video Broadcasting (DVB); Interaction channel through the Global System for Mobile communications (GSM)*

EN 301 199, *Digital Video Broadcasting (DVB); Interaction channel for Local Multi-point Distribution Systems (LMDS)*

EN 301 790, *Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems*

ARIB STD-B10:2007, *Service information for digital broadcasting system*

ARIB STD-B23:2004, *Application execution engine platform for digital broadcasting*

ARIB STD-B24:2007, *Presentation engine platform for digital broadcasting*

ETS 300 800, *Digital Video Broadcasting (DVB); Interaction channel for Cable TV distribution systems (CATV)*

ETS 300 801, *Digital Video Broadcasting (DVB); Interaction channel through Public Switched Telecommunications Network (PSTN) / Integrated Services Digital Networks (ISDN)*

ETSI TR 101 162, *Digital Video Broadcasting (DVB); Allocation of service information (SI), codes for DVB systems*

ETSI TR 101 201, *Digital Video Broadcasting (DVB); Interaction channel for Satellite Master Antenna TV (SMATV) distribution systems; Guidelines for versions based on satellite and coaxial sections*

ETSI TR 101 202, *Digital Video Broadcasting (DVB); Implementation guidelines for data broadcasting*

ETSI TS 101 812:2003, *Multimedia home platform – MHP specification 1.03*

GEM 1.0:2005, *Globally executable MHP Version 1.02*

GEM 1.1:2006, *Globally executable MHP (GEM) Especification 1.1*

IEEE 802:2001, *IEEE Standard for Local and Metropolitan Area Networks: Overviem and Architecture*

RFC 768, *User Datagram Protocol*

RFC 791, DARPA *Internet program protocol specification*

RFC 793:1981, *Transmission Control Protocol Darpa Internet Program Protocol Specification*

RFC 1034, *Domain names - concepts and facilities*

RFC 1035, *Domain names - implementation and specification*

RFC 1112, *Host extensions for IP multicasting*

RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*

RFC 1521, *Borenstein N., and N. Freed, MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message*

RFC 1661, *The Point-to-Point Protocol (PPP)*

RFC 1717, *The PPP Multilink Protocol (MP)*

RFC 1877, *PPP Internet Protocol Control Protocol Extensions for Name Server Addresses*

RFC 1945, *Hypertext Transfer Protocol - HTTP/1.0*

RFC 1950, *ZLIB Compressed data format specification version 3.3*

RFC 1982, *Serial Number Arithmetic*

RFC 2181, *Clarifications to the DNS Specification*

RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1*

RFC 2818, *HTTP Over TLS*

RFC 2396, *URI Generic Syntax*

CORBA/IIOP, *Common Object Request Broker Architecture Specification, Internet Inter-ORB Protocol*

3 Términos y definiciones

Para los efectos de esta parte de la ABNT NBR 15606, se aplican los siguientes términos y definiciones.

3.1
acceso múltiple por división de código de múltiple portador
MC-CDMA
esquema de acceso múltiple por división de código (CDMA) que emplea sistemas de múltiples portadores

3.2
***buffer* de transporte**
buffer que decodifica un paquete de *stream* de transporte en un decodificador-meta de un *stream* de transporte MPEG2

3.3
encabezamiento del paquete PES
campo que comprende la primera parte de un paquete PES

3.4
carrusel de datos
método que envía cualquier conjunto de datos repetidamente para que los datos puedan ser bajados vía transmisión siempre que sea necesario

NOTA Este método se especifica en la ISO/IEC 13818-6.

3.5
comando y control de almacenamiento de medios digital
DSMCC
método que brinda soporte al acceso a archivos y *streams* en servicio digital interactivo

3.6
comité para sistema de televisión avanzado
ATSC
comité con el propósito de estandarizar el sistema de transmisión digital en EUA

3.7
consejo audiovisual digital
DAVIC
Consorcio industrial para estandarización del servicio de multimedia interactiva

3.8
contenido
grupo de datos transmitidos por medio de un programa de transmisión de datos o servicio de comunicaciones bidireccionales que es generado por el programa de transmisión de datos para servir como parte del programa de transmisión de datos

NOTA Como un término en el contexto de transmisión, "contenido" indica un conjunto de *streams* en el programa que envía el grupo de datos. Un único programa de transmisión puede estar compuesto por múltiples contenidos.

3.9
contenido local
parte del contenido de transmisión de datos contenido en un único evento de datos

NOTA Generalmente, un contenido local es un grupo de datos agrupados con base en el contexto o para una mayor conveniencia de producción de programa.

3.10
DSM-CC U-U
Digital Storage Media – Command and Control User-to-User Interface

3.11
dirección
protocolo utilizado para definir un nombre de servidor utilizando el PPP

NOTA Esta definición está de acuerdo con la RFC 1877.

3.12
ethernet
estándar LAN que define una red con base en barramento empleando el CSMA/CD (acceso múltiple de sentido de portador/detección de colisión) para control de acceso

NOTA Esta definición está de acuerdo con la IEEE 802.

3.13
evento de datos
conjunto de *streams* de transmisión de datos que representa un grupo de contenido de transmisión de datos a ser distribuido con los tiempos de inicio y finalización preconfigurados

NOTA El concepto de evento de datos *(data event)* es introducido para permitir que un grupo de contenido de transmisión de datos sea alternado para otro, si están o no en el mismo programa, conforme sea necesario. En otras palabras, un evento de datos es independiente de un evento.

3.14
formato de adaptación
formato utilizado en un encabezamiento DSM-CC y que es una forma de información inserida en un área de adaptación que codifica la información para atender a una solicitación, dependiendo de la red de distribución

3.15
hash function
función embrollo
función matemática que mapea un área amplia (inmensa en algunos casos) dentro de una pequeña banda

NOTA Una *hash function* es de dirección única y libre de colisión.

3.16
host
máquina
dispositivo de punto de acceso o dispositivo de servidor, necesario para servicios de transmisión bidireccional

3.17
hypertext transfer protocol
HTTP
capa de aplicación para transmitir datos a través de la World Wide Web

NOTA Esta definición está de acuerdo con la RFC 1954.

3.18
identificador de paquete
PID
identificador de paquete de un *stream* de transporte MPEG-2

3.19
información de servicio
SI
datos digitales que describen un arreglo de programas, un sistema de distribución para transmisión de *streams* de datos, descripción de programas, informaciones de parrilla de programación/tiempo de duración

NOTA Estos datos también transportan MPEG-2 PSI (Informaciones Específicas del Programa), así como partes de la extensión definidas de forma independiente.

3.20
información específica del programa
PSI
información de control de transmisión, que suministra la información necesaria para permitir a un receptor automáticamente demultiplexar y decodificar varios *streams* de programa que fueron multiplexados

3.21
MIME
protocolo de capa de aplicación que suministra una arquitectura de contenido que permite que datos multimedia, como archivos de texto, audio e imágenes, en formato que no sea US-ASCII, sean transmitidos vía *e-mail*

3.22
paquete PES
formato de datos usado para transmitir *streams* básicos que consiste en un encabezamiento de paquete PES y una carga útil PES inmediatamente siguiendo el encabezamiento

3.23
private_stream_1
tipo de *stream* transmitido usando PES y que se utiliza para transmitir un *stream* privado sincronizado con otros *streams*

3.24
private_stream_2
tipo de *stream* transmitido usando PES y que se utiliza para transmitir un *stream* privado que no necesita ser sincronizado con otros *streams*

3.25
procedimiento de control de conexión de datos de alto nivel
procedimiento HDLC
procedimiento de control de transmisión con alta confiabilidad, usado para comunicación entre computadoras, principalmente en LAN e *internet*

3.26
protocolo datagrama del usuario
UDP
protocolo de capa de transporte que promueve entrega de datos sin conexión entre dos máquinas

NOTA 1 Aunque el UDP no soporte mensajes de reconocimiento, minimiza el protocolo elevado para mayor eficiencia en servicios de transmisión.

NOTA 2 Esta definición está de acuerdo con la RFC 768.

3.27
protocolo de autenticación de contraseña NNTP
protocolo de capa de aplicación utilizado para distribuir, enviar y recuperar noticias *(Net News)* en Internet

NOTA Esta definición está de acuerdo con la RFC 1334.

3.28
protocolo de autenticación de contraseña PAP
componente del protocolo de punto a punto (PPP) que soporta la autenticación

NOTA 1 Este protocolo no hace identificación de usuario y contraseña al enviar.

NOTA 2 Esta definición ésta de acuerdo con la RFC 1334.

3.29
protocolo de control de transmisión
TCP
protocolo de capa de transporte que promueve distribución de datos altamente confiable, de punta la punta, orientada por conexión, utilizando un mecanismo de detección y corrección de error
NOTA Esta definición está de acuerdo con la RFC 793.

3.30
protocolo de control IP
IPCP
protocolo utilizado para establecer varias configuraciones exigidas para utilizar IP en la fase de protocolo de capa de red

NOTA Esta definición está de acuerdo con la RFC 1332.

3.31
protocolo de internet
IP
protocolo de capa de red que define el mecanismo de encaminamiento en Internet para permitir que los datos sean transmitidos

NOTA Esta definición está de acuerdo con la RFC 791.

3.32
protocolo de resolución de dirección
ARP
Protocolo utilizado en una red TCP/IP para obtener la dirección física del nudo Ethernet basado en su dirección de IP

3.33
protocolo de transmisión de datos en modo básico
protocolo de comunicaciones desarrollados para transmisión básica de datos entre un *host* y un terminal

NOTA El protocolo emplea un método para minimizar los errores de transmisión.

3.34
red digital de servicios integrados
ISDN
red digital de servicios integrados

3.35
reservado
término que, cuando se utiliza en sentencias que definen el stream de bit codificado, indica que el valor se puede usar en el futuro para extensiones definidas por la ISO

NOTA Los bits reservados se configuran en 1.

3.36
reserved_future_use
término que, cuando se utiliza en sentencias definiendo el *stream* de bit codificado, indica que el valor se puede utilizar en extensiones definidas por la ISO en el futuro

NOTA Los bits reservados se configuran en 1.

3.37
sección
Estructura sintáctica utilizada para mapear las informaciones de servicio y otros datos dentro de un paquete de *stream* de transporte

3.38
servicio de nombre de dominio
DNS
protocolo utilizado por el servicio que mapea un nombre de máquina en una red dentro de su dirección de IP

NOTA Esta definición está de acuerdo con las RFC 1034 y RFC 1035.

3.39
tabla de información de evento
EIT
tabla de información de evento que contiene datos relacionados a un evento y un programa como un nombre de evento (programa), hora de inicio y un período

3.40
tabla de mapeo de programa
PMT
tabla que forma parte del PSI

3.41
transmisión de video digital
DVB
proyecto para estandarización del sistema de transmisión digital en Europa

4 Tipos de especificación de transmisión de datos

Los tipos de especificaciones para transmisión de datos e identificación de tipo de *stream* contenidas en un PMT se presentan en la Tabla 1.

Tabla 1 — Tipos de especificación de transmisión

Especificación de la transmisión	Función mayoritaria y facilidad de uso	Identificación del tipo de *stream*
PES independiente	Utilizado para *streams* de datos sincronizados a asíncronos para servicios de radiodifusión	0x06
Carrusel de datos/objetos	Utilizado para transferencias de datos en general: Sincronizados y asíncronos para servicios de radiodifusión. Aplicado a la transmisión de datos para servicios de *download* y servicios multimedia	0x0B, 0X0D [a]
Mensaje de eventos	Utilizado para notificaciones sincronizadas y asíncronas referentes a las aplicaciones en el TA partiendo de la estaca de *broadcast*. Utilizado para servicios de multimedia	0x0C, 0X0D [b]
Protocolos de canal de interactividad	Protocolo de transmisión utilizado en redes fijas como redes PSTN/ISDN y redes de celular, incluyendo red celular/PHS con comunicaciones bidireccionales[d]	—
Encapsulado multiprotocolo	Datagramas son encapsulados en *datagram_sections* Que son compatibles con el formato *DSMCC_section* para datos privados	0x0A [c]
Data piping	Protocolo que permite insertar datos de una red de radiodifusión directamente en el *payload* del paquete MPEG-2	0x7E

[a] Cuando un *stream* no contiene datos DSM-CC, sino un carrusel de datos, se utiliza 0x0B ó 0x0D y, cuando también tiene otros datos se utiliza DSM-CC, 0x0D.

[b] Cuando un *stream* no contiene datos DSM-CC, sino como mensaje de evento, se utiliza 0x0C ó 0x0D y, cuando también tiene otros datos DSM-CC, se utiliza 0x0D.

[c] Cuando un *stream* no contiene datos DSM-CC, sino datos de encapsulado de multiprotocolos (MPE), se utiliza 0x0A y, cuando también tiene otros datos DSM-CC, se utiliza 0x0D.

[d] PSTN: Red Telefónica conmutada pública.

5 Especificación de transmisión del carrusel de datos

5.1 Transmisión con carrusel de datos DSM-CC

La especificación de transmisión del carrusel de datos se destina a implementar la transmisión general sincronizada o asíncrona sin la necesidad de datos *streaming,* tales como *download* de datos para una unidad receptora o transmisión de contenidos para servicios de multimedia. La especificación de transmisión del carrusel de datos definida en esta Norma se basa en la especificación del carrusel de datos DSM-CC establecida en la ISO/I EC 13818-6.

La transmisión repetida de datos, como es definida en la especificación del carrusel de datos DSC-CC, permite a la unidad receptora obtener datos por demanda en cualquier momento durante un período de transmisión.

Los datos se transmiten en una unidad modular formada por bloques donde todos los bloques, excepto los que están al final del módulo, tienen el mismo tamaño y cada bloque se transmite en secciones.

En la transmisión de estos datos se utilizan el mensaje *download* de bloque de datos (referida como mensaje DDB) y mensaje de indicación de información *download* (referida como mensaje DII). Ambos mensajes son componentes del protocolo de *download* del usuario de la red especificado en la ISO/IEC 13818-6. El cuerpo de datos se transmite por el mensaje DDB con cada módulo dividido dentro de los bloques. Para informaciones adicionales relacionadas a la transmisión PSI/SI, ver el Anexo B.

5.2 Mensaje de control DSM-CC

5.2.1 Mensaje de indicación de información de *download* (DII)

Un mensaje DII forma parte de un mensaje de control DSM-CC. Así, el mensaje DII transmite el contenido del mensaje reteniéndolo en el *userNetworkMessage()* en la sección DSM-CC.

La versión del mensaje DII es indicada por el *transaction_number* (número de la transacción) en el campo *transaction_id* (identificación de transacción) del *dsmccMessageHeader.* Este número de versión es común a todos los mensajes DII del carrusel de datos y el número de la versión se incrementa en uno cuando el contenido de uno o más mensajes DII se altera.

5.2.2 Sintaxis y semántica del mensaje DII

La estructura de los datos del mensaje DII se presenta en la Tabla 2.

Tabla 2 — Estructura de los datos del mensaje de indicación de información de *download*

Sintaxis	Número de bits	Mnemónico
DownloadInfoIndication() {		
dsmccMessageHeader()		
downloadId	32	*uimsbf*
blockSize	16	*uimsbf*
windowSize	8	*uimsbf*
ackPeriod	8	*uimsbf*
tCDownloadWindow	32	*uimsbf*
tCDownloadScenario	32	*uimsbf*
compatibilityDescriptor()		
numberOfModules	16	*uimsbf*
for(i=0;i< numberOfModules;i++) {		
moduleId	16	*uimsbf*
moduleSize	32	*uimsbf*
moduleVersion	8	*uimsbf*
moduleInfoLength	8	*uimsbf*
for(i=0;i< moduleInfoLength;i++){		
moduleInfoByte	8	*uimsbf*
}		
}		
privateDataLength	16	*uimsbf*
for(i=0;i<privateDataLength;i++){		
privateDataByte	8	*uimsbf*
}		
}		

La semántica de los campos DII debe ser la siguiente:

— ***dsmccMessageHeader ()* (encabezamiento del mensaje DSM-CC)**: tal como especificado en 5.3;

— ***downloadId (identificador de download)**:* campo de 32 bits que sirve como un rótulo para la identificación única del carrusel de datos. En el caso de evento de operación de datos, *data_event_id* (identificación del evento de datos) debe ser insertado en los bits 28-31 del *downloadId* (identificador de *download).* En caso contrario, la banda y los valores para asegurar la unicidad se especifica en un estándar operativo;

— ***windowSize**:* campo de 8 bits que no se utiliza para transmisión del carrusel de datos y el valor debe ser ajustado en 0;

— ***ackPeriod**:* campo de 8 bits que no se utiliza en la transmisión del carrusel de datos y el valor debe ser ajustado en 0;

— ***tCDownloadWindow**:* campo de 32 bits que no se utiliza en la transmisión del carrusel de datos y el valor debe ser ajustado en 0;

— ***tCDownloadScenario**:* campo de 32 bits que indica el período de límite de tiempo en el que se presume que el *download* está completo en microsegundos;

— ***compatibilityDescriptor():*** estructura del descriptor de compatibilidad *(compatibilityDescriptor())* que se especifica en la ISO/IEC 13818-6 y que debe ser configurada en este campo. Cuando el contenido de la estructura del *compatibilityDescriptor()* no es necesaria, el *descriptorCount* se debe configurar en 0x0000 y, así, la extensión del campo debe ser de 4 bytes;

— ***numberOfModules*** **(número de módulo)**: campo de 16 bits que indica el número de módulos descritos en el enlace siguiente en este mensaje DII;

— ***moduleId*** **(identificador de módulo)**: campo de 16 bits que indica la identificación del módulo descrito en los siguientes campos: *ModuleSize, module Version* y *moduleInfoByte;*

— ***moduleSize*** **(extensión del módulo)**: campo de 32 bits que indica la extensión byte del módulo. Cuando la extensión del byte del módulo no es conocida, debe ser configurada en 0;

— ***moduleVersion****:* campo de 8 bits que indica la versión de este módulo;

— ***moduleInfoLength*** **(extensión de la información del módulo)**: campo de 8 bits que indica la extensión byte del área de información del módulo;

— ***moduleInfoByte*** **(información del módulo)**: campo de unidad de 8 bits que se puede usar para insertar descriptores relacionados al módulo. Estos descriptores se definen en 5.4;

NOTA Los valores de *tag* de los descriptores a ser insertados se definen en la Tabla 5.

— ***privateDataLength:*** campo de 16 bits que indica la extensión byte del campo *PrivateDataByte;*

— ***privateDataByte*** **(datos privados):** campo de unidad de 8 bits que se puede usar para contener una estructura de datos en un formato de descriptor. La estructura de datos es definida con base en un formato de codificación de datos o por un operador de servicio.

La semántica de los valores de *tag* de los descriptores para este campo es definida en la Tabla 3. Los descriptores posibles para este campo son los definidos en 5.4 y por un formato de codificación de datos.

Tabla 3 — Semántica de los *tags* de los descriptores del área de información de módulo y área privada en el DII

Valor de *tag* del descriptor	Semántica
0x01 – 0x7F	Valores de *tag* reservados de descriptores compatibles que se insertarán en el área de información del módulo y área privada (ver 5.4)
0x80 - 0xBF	Valores de *tag* disponibles de descriptores definidos por un operador de servicio
0xC0 – 0 xEF	Valores de *tag* reservados de descriptores a ser insertados en el área de información del módulo y área privada (ver 5.4)
0xF0 – 0 xFE	Valores de *tag* reservados de descriptores definidos con base en un formato de codificación de datos

5.3 Sintaxis y semántica del *dsmccMessageHeader()*

La estructura de datos del *dsmccMessageHeader()* se define en la Tabla 4.

Tabla 4 — Estructura de datos del *dsmccMessageHeader*

Sintaxis	Número de bits	Mnemónico
dsmccMessageHeader() {		
protocolDiscriminator	8	*uimsbf*
dsmccType	8	*uimsbf*
messageID	16	*uimsbf*
transaction_id	32	*uimsbf*
Reserved	8	*bslbf*
adaptationLength	8	*uimsbf*
messageLength	16	*uimsbf*
if(adaptationLength>0){		
dsmccAdaptationHeader()	16	*uimsbf*
}		
}		

La semántica del *dsmccMessageHeader()* debe ser la siguiente:

— ***protocolDiscriminator:*** campo de 8 bits que se debe configurar en 0 x 11 e indica que este mensaje es del tipo MPEG-2 DSM-CC;

— ***dsmccType* (tipo DSM-CC):** campo de 8 bits que indica el tipo del mensaje MPEG-2 DSM-CC. En un mensaje DII para transmisión del carrusel de datos, se debe configurar en 0x03 (mensaje *download* U-N*);*

— ***messageId* (identificador del tipo del mensaje):** campo de 16 bits que identifica el tipo del mensaje DSM-CC. En un mensaje DII, se debe configurar en 0x1002;

— ***transaction_id* (identificador de transacción):** campo de 32 bits que identifica el mensaje y tiene la función de controlar la versión. El formato de la *transaction_id* se muestra en la Figura 1. El campo *Transaction Number* en los bits 0-29 se debe usar para identificar la versión de la DII, como especificado en la ISO/IEC 13818-6. El valor de bits 30-31 se debe configurar en '10' *(TransactionId* asignado por la red) conforme definido en el *Transaction id Originator,* como especificado en la ISO/IEC 13818-6;

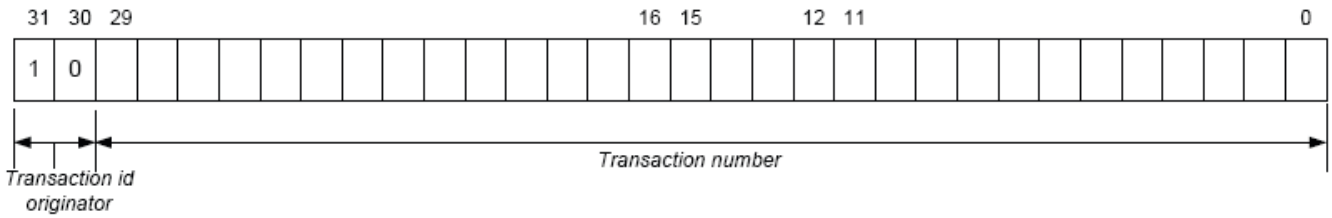


Figura 1 — Formato de la *transaction_id*

— ***adaptationLength:*** campo de 8 bits que indica el número de bytes del campo *dsmccAdaptationHeader();*

— ***messageLength:*** Campo de 16 bits que indica el número de bytes del mensaje inmediatamente después de este campo. Es decir, el valor es una suma de la extensión *payload* y de la extensión *dsmccAdaptationHeader();*

— ***dsmccAdaptationHeader():*** La estructura de datos de este campo está definida en 5.5.3.

5.4 Descriptores del área de información del módulo y área privada

5.4.1 Tipos de descriptores

Los tipos de descriptores utilizados en un área de información del módulo y en un área privada son mostrados en la Tabla 5. Cualquiera de estos descriptores se puede usar en el área de información del módulo y/o en un área privada conforme sea necesario. Los descriptores contenidos en un área privada en una DII se aplican a los módulos en la DII. Cuando el área de información del módulo y el área privada tienen el mismo conjunto de descriptores, sólo los descriptores en el área de información del módulo son activados.

Tabla 5 — Tipos de descriptores

Valor de *tag*	Descriptor	Función	Área de información del módulo	Área privada
0x01	*type_descriptor*	Tipo de módulo (forma MIME etc.)	o	
0x02	*name_descriptor*	Nombre del módulo (nombre del archivo)	o	
0x03	*info_descriptor*	Información del módulo (tipo de carácter)	o	o
0x04	*module_link_descriptor*	Información del *link* (id del módulo)	o	
0x05	*CRC32_descriptor*	CRC32 del módulo	o	
0x06	*location_descriptor*		o	o
0x07	*est_download_time_descriptor*	Tiempo estimado de *download* (s)	o	o
0x08 - 0x7F	Reservado para el futuro			
0x80 - 0xBF	Disponible para un *broadcaster*			
0xC0 - 0xC1	Reservado para el futuro			
0xC2	*compression_Type_descriptor*	Algoritmo de compresión cuando el módulo se transmite	o	
0xC3 - 0xCC	Reservado para el futuro			
0xCD - 0xEE	Reservado para el futuro			

5.4.2 Descriptor de tipo

El descriptor de tipo (ver Tabla 6) indica el tipo de archivo transmitido como un módulo único, implementando la transmisión del carrusel de datos con base en esta Norma, que especifica que un archivo único se transmite como un módulo único.

Tabla 6 — Descriptor de tipo

Sintaxis	Número de bits	Mnemónico
Type_descriptor(){		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
for(i=0;i<N;i++) {		
text_char	8	*uimsbf*
}		
}		

La semántica de campo en el descriptor de tipo debe ser la siguiente:

— ***text_char:*** campo de 8 bits. La secuencia de estos campos indica tipo de media de acuerdo con la RFC 1521.

5.4.3 Descriptor del nombre

El descriptor del nombre (ver Tabla 7) indica el nombre del archivo transmitido como un módulo único, implementando la transmisión del carrusel de datos con base en esta Norma, que especifica que un archivo único se transmite como un módulo único. Sin embargo, cuando hay el descriptor *Module Link,* el descriptor del nombre no debe estar presente en otra posición sino = módulo 0x00 en la DII.

Tabla 7 — Descriptor del nombre

Sintaxis	Número de bits	Mnemónico
Name_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
for(i=0;i<N;i++) {		
text_char	8	*uimsbf*
}		
}		

La semántica de campo en el descriptor del nombre deber ser la siguiente:

— ***text_char:*** campo de 8 bits. La secuencia de este campo indica el nombre del archivo transmitido como un módulo único usando la especificación de codificación de datos o un código de carácter especificado en un estándar operativo.

5.4.4 Descriptor de información

El descriptor de información (ver Tabla 8) describe las informaciones relacionadas al módulo.

Tabla 8 — Descriptor de información

Sintaxis	Número de bits	Mnemónico
info_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
ISO_639_language_code	24	*bslbf*
for(i=0;i<N;i++) {		
text_char	8	*uimsbf*
}		
}		

La semántica de campos en el descriptor de información debe ser la siguiente:

— ***ISO_639_language_code:*** Campo de 24 bits que identifica el lenguaje que se utiliza en el área *text_char.* El código del lenguaje es representado por tres caracteres alfabéticos especificados en la ISO 639-2. Cada carácter se codifica dentro de una representación de 8 bits de acuerdo con la ISO 8859-1 y es insertado dentro de un campo de 24 bits en ese orden;

— ***text_char:*** campo de 8 bits. La secuencia de estos campos indica la información textual relacionada al archivo transmitido como un módulo único usando la especificación de codificación de datos o un código de señalización especificado en una norma operativa.

5.4.5 Descriptor del *Module_link*

El descriptor *Module_link* (ver Tabla 9) genera una lista de módulos conectados a otros módulos. Por ser la extensión del campo número de bloques de un mensaje DDB restringido a 16 bits, el tamaño máximo de un módulo en la transmisión del carrusel de datos es de 256 Mbytes. Cuando se transmite un archivo mayor que 256 Mbytes, el archivo se divide en dos o más módulos antes de ser enviado y esta información se asocia al descriptor del *Module_link.*

Tabla 9 — Descriptor del *Module_link*

Sintaxis	Número de bits	Mnemónico
module_link_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
position	8	*uimsbf*
moduleId	16	*uimsbf*
}		

La semántica de campos en el descriptor *Module_link* debe ser la siguiente:

— ***position:*** campo de 8 bits que indica la relación de posición con módulo conectado. “0x00” indica que está localizado en la parte superior del *link,* “0x01 indica que está en medio y “0x02” indica que está en el final;

— ***module Id:*** campo de 16 bits que es la identificación del módulo conectado. Cuando la posición es “0x02”, el valor de este campo es ignorado.

5.4.6 Descriptor de la localización

El *location_descriptor* contiene la localización del PID donde los bloques, módulos o grupos pueden ser encontrados conteniendo los datos del carrusel. La Tabla 10 muestra la sintaxis del *location_descriptor.*

Tabla 10 — Sintaxis del *location_descriptor*

Sintaxis	Número de bits	Valor
location_descriptor(){		
descriptor_tag	8	0x06
descriptor_length	8	
location_tag	8	
}		

La semántica del *location_descriptor* debe ser la siguiente:

— ***descriptor_tag:*** campo de 8 bits que identifica el descriptor. El *location_descriptor* está configurado en 0x06;

— ***descriptor_length:*** campo de 8 bits que especifica el número de bytes del descriptor inmediatamente después de este campo;

— ***location_tag:*** campo de 8 bits que tiene el mismo valor que el campo *component_tag* en el descriptor identificador del *stream.*

5.4.7 Descriptor CRC

El descriptor CRC (ver Tabla 11) describe el valor CRC del módulo completo.

Tabla 11 — Descriptor CRC

Sintaxis	Número de bits	Mnemónico
CRC32_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
CRC_32	32	*rpchof*
}		

La semántica de campo en el descriptor CRC debe ser la siguiente:

— **CRC_32:** campo de 32 bits que almacena el valor CRC calculado para el módulo completo. El valor CRC debe ser calculado como definido en la ABNT NBR 15603-2:2007, Anexo B.

5.4.8 Descriptor de tiempo estimado de *download*

El descriptor de tiempo estimado de *download* (ver Tabla 12) describe el período estimado necesario para el *download* del módulo.

Tabla 12 — Descriptor de tiempo estimado de *download*

Sintaxis	Número de bits	Mnemónico
est_download_time_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
est_download_time [a]	32	*uimsbf*
}		

La semántica del campo en el descriptor del tiempo estimado de *download* debe ser la siguiente:

— ***est_download_time:*** campo de 32 bits que indica el período estimado, en segundos, necesario para hacer el *download* del módulo.

5.4.9 Descriptor de tipo de compresión

El descriptor de tipo de compresión (ver Tabla 13) indica que el módulo fue comprimido en el formato zlib basado en la RFC 1950 y muestra su algoritmo de compresión y el tamaño del módulo antes de la compresión en bytes. Un módulo que no haya sido comprimido no tiene ese descriptor.

Tabla 13 — Descriptor de tipo de compresión

Sintaxis	Número de bits	Mnemónico
Compression_Type_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
compression_type	8	*uimsbf*
original_size	32	*uimsbf*
}		

La semántica de campos en el descriptor de tipo de compresión debe ser la siguiente:

— ***compression_type:*** campo de 8 bits que define el tipo de compresión usada para comprimir el módulo;

— ***original_size:*** campo de 32 bits que indica el tamaño del módulo antes de la compresión en bytes.

5.5 Mensaje *DownloadDataBlock* (DDB)

5.5.1 Sintaxis y semántica del mensaje DDB

El contenido de un mensaje DDB se transmite por almacenamiento en el campo *downloadDataMessage()* en la sección DSM-CC.

Un mensaje DDB es la estructura de datos transmitiendo bloques de datos (ver Tabla 14). Un módulo se puede dividir con extensión fijada para formar bloques. En ese caso, cada bloque es representado con un número de bloque en el mensaje DDB para permitir que una unidad receptora reorganice los bloques en el orden pretendido.

De acuerdo con lo especificado en la ISO/IEC 13818-6, cuando los mensajes DDB se transmiten en MPEG-2 TS, apenas los mensajes DDB que tienen el mismo *downloadId* deben ser incluidos en el mismo paquete PID. Eso significa que los mensajes DDB en dos carruseles diferentes no se deben presentar en un único *stream* elemental.

Tabla 14 — Estructura de datos del bloque de datos de *download*

Sintaxis	Número de bits	Mnemónico
DownloadDataBlock() {		
dsmccDownloadDataHeader()		
moduleId	16	*uimsbf*
moduleVersion	8	*uimsbf*
reserved	8	*bslbf*
blockNumber	16	*uimsbf*
for(i=0;i<N;i++) {		
blockDataByte	8	*uimsbf*
}		
}		

Los campos de DDB deben ser los siguientes:

— ***moduleId:*** campo de 16 bits que indica el número de identificación al cual este bloque pertenece;

— ***moduleVersion:*** campo de 8 bits que indica la versión del módulo al cual este bloque pertenece;

— ***blockNumber:*** campo de 16 bits que indica la posición de este bloque dentro del módulo. El primer bloque de un módulo debe ser representado por el bloque número 0;

— ***blockDataByte:*** campo de 8 bits. El tamaño de una serie del área de datos del bloque es igual al tamaño del bloque de la DII, es decir, el tamaño de los bloques divididos desde un módulo. Sin embargo, el número del último bloque en el módulo puede ser menor que el tamaño de bloque descrito en la DII.

5.5.2 Sintaxis y semántica del *dsmccDownloadDataHeader()*

La estructura de datos del *dsmccDownloadDataHeader()* está definida en la Tabla 15.

Tabla 15 — Estructura de datos del *dsmccDownloadDataHeader*

Sintaxis	Número de bits	Mnemónico
dsmccDownloadDataHeader() {		
protocolDiscriminator	8	*uimsbf*
dsmccType	8	*uimsbf*
messageId	16	*uimsbf*
downloadId	32	*uimsbf*
Reserved	8	*bslbf*
adaptationLength	8	*uimsbf*
messageLength	16	*uimsbf*
if(adaptationLength>0) {		
dsmccAdaptationHeader()	8	*uimsbf*
}		
}		

Los campos del *dsmccDownloadDataHeader()* deben ser los siguientes:

— ***protocol discriminator:*** campo de 8 bits que es configurado en 0x11 e indica que este mensaje es un mensaje DSM-CC MPEG-2 *dsmccType.* Este campo de 8 bits indica el tipo del mensaje DSM-CC MPEG-2 y es configurado en 0x03 (mensaje *download* U-N) para el mensaje DDB en la transmisión del carrusel de datos;

— ***messageId:*** campo de 16 bits que identifica el tipo del mensaje DSM-CC y es configurado en 0x1003 para un mensaje DDB;

— ***downloadId:*** campo de 32 bits que es configurado en el mismo valor que el identificador de *download* en el mensaje DII correspondiente;

— ***adaptationLength:*** campo de 8 bits que indica el número de bytes del campo *dsmccAdaptationHeader();*

— ***dsmccAdaptationHeader():*** la estructura de datos de este campo es definida en 5.5.3;

— ***messageLength:*** campo de 16 bits que indica la extensión del mensaje, no incluyendo este campo y su área precedente en bytes. El valor es idéntico a la suma de la extensión de *payload* y la extensión *dsmccAdaptationHeader.*

5.5.3 Sintaxis del *dsmccAdaptationHeader()*

En el *dsmccMessageHeader()* que es el encabezamiento de un mensaje DII y en el *dsmccDownloadDataHeader()* que es el encabezamiento de un mensaje DDB, puede ser establecida la estructura de datos común *dsmccAdaptationHeader().*

La estructura de datos del *dsmccAdaptationHeader* es indicada en la Tabla 16.

Tabla 16 — Estructura del *dsmccAdaptationHeader*

Sintaxis	Número de bits	Mnemónico
dsmccAdaptationHeader() {		
adaptationType	8	*uimsbf*
for (i=0; i<(adaptationLength-1);i++) {		
adaptationDataByte	8	*uimsbf*
}		
}		
}		

La semántica del *dsmccAdaptationHeader()* debe ser la siguiente:

— ***adaptation Type:*** campo de 8 bits que indica el tipo de encabezamiento de adaptación. El valor de este campo indica un formato de adaptación conforme Tabla 17.

Tabla 17 — Tipo de adaptación

Tipo de adaptación	Formato de la adaptación	Definición en la ISO/IEC 13818-6
0x00	Reservado	Lo mismo que en la columna a la izquierda
0x01	Reservado	DSM-CC Acceso condicional
0x02	Reservado	DSM-CC identificador de usuario
0x03	*DIIMsgNumber*	Lo mismo que en la columna a la izquierda
0x04-0x7F	Reservado	Lo mismo que en la columna a la izquierda
0x80-0xFF	Definición del usuario	Lo mismo que en la columna a la izquierda
NOTA Para los tipos de adaptación utilizados en esta Norma, la operación del formato de adaptación de definición del usuario del tipo de adaptación 0x80 – 0xFF es opcionalmente hecha por un operador de servicio.		

5.5.4 Sintaxis de la sección DSM-CC

Los mensajes DII y DDB se transmiten usando las secciones DSM-CC, como se muestra en la Tabla 18.

Tabla 18 — Sección DSM-CC (transmisión de mensajes DII/DBB)

Sintaxis	Número de bits	Mnemónico
DSMCC_section () {		
table_id	8	*uimsbf*
section_syntax_indicator	1	*bslbf*
private_indicator	1	*bslbf*
reserved	2	*bslbf*
dsmcc_section_length	12	*uimsbf*
table_id_extension	16	*uimsbf*
reserved	2	*bslbf*
version_number	5	*uimsbf*
current_next_indicator	1	*bslbf*
section_number	8	*uimsbf*
last_section_number	8	*uimsbf*
if (table_id==0x3B) {		
userNetworkMessage ()		
}		
else if (table_id==0x3C) {		
downloadDataMessage()		
}		
else if (table_id==0x3E) {		
for (i=0;i<dsmcc_section_length-9;i++) {		
private_data_byte	8	*uimsbf*
}		
}		
if (section_syntax_indicator=='0') {		
Checksum	32	*uimsbf*
}		
else {		
CRC_32	32	*rpchof*
}		
}		

La semántica de la sección DSM-CC debe ser la siguiente:

— ***table_id:*** campo de 8 bits que contiene el número de identificación del tipo de datos en la sección de *payload* DSM-CC. Basado en el valor de este campo, se aplica una regla de codificación específica para el campo siguiente en la sección DSM-CC. La tabla de los valores de identificación es mostrada en la Tabla 19, como especificado en la ISO/IEC 13818-6;

Tabla 19 — *Table_id*

table_id	Tipo de sección DSM-CC	Definición en la ISO/IEC 13818-6
0x3A	Reservado	Cápsula multiprotocolo [a]
0x3B	Mensaje DII	Mensaje U-N incluyendo DII
0x3C	Mensaje DDB	Lo mismo que en la columna a la izquierda
0x3D	Descriptor de *Stream*	Lo mismo que en la columna a la izquierda
0x3E	Datos privados	Lo mismo que en la columna a la izquierda
0x3F	Reservado	Lo mismo que en la columna a la izquierda
[a] Ver ISO/IEC 13818-6.		

— ***section_syntax_indicator:*** campo de 1 bit. Cuando es configurado en 1, indica que existe un CRC32 al final de la sección. Cuando es configurado en ***0, indica que existe una suma de verificación. Se debe configurar en 1 para la transmisión de los mensajes DII y DDB;

— ***private_indicator:*** campo de 1 bit que almacena el valor complementar del *flag* del *section_syntax_indicator;*

— ***dsmcc_section_length:*** campo de 12 bits que indica el número de bytes del área desde el inicio del campo, inmediatamente después de ese campo hasta el fin de la sección. El valor en este campo no debe exceder 4 093 bytes;

— ***table_id_extension:*** campo de 16 bits que es configurado como mostrado abajo, de acuerdo con el campo *table_id:*

- cuando el valor del campo *table_id* es igual a 0x3B, este campo debe transportar una copia de los 2 bytes menos significativos del campo *transaction_id;*
- cuando el valor del campo *table_id* es igual a 0x3C, este campo debe transportar una copia del campo *module_id;*

— ***version_number:*** campo de 5 bits que es configurado de acuerdo con el identificador de tabla (table_id). Cuando el valor del campo table_id es igual a 0x3B, este campo se debe configurar en “0”. Cuando el valor del campo table_id es igual a 0x3C, se debe configurar en los 5 bits menos significativos del campo versión del módulo;

— ***current_next_indicator:*** designación de 1 bit que indica que la subtabla está activa cuando está en “1”. Cuando está en “0”, a subtabla enviada aún no fue aplicada y usada como la próxima subtabla. Cuando el valor del campo *table_id* es igual a un valor en la banda de 0x3A a 0x3C, este campo se debe configurar en “1”;

— ***section_number:*** campo de 8 bits que indica el número de la sección de la primera sección en la subtabla. Cuando la sección contiene un mensaje DII, este campo se debe configurar en 0. Cuando esta sección contiene un mensaje DDB, este campo debe transportar una copia de los 8 bits menos significantes del número del bloque de la DDB;

— ***last_section_number:*** campo de 8 bits que indica el número de la última sección (sección que tiene el número máximo de la sección) de la subtabla a la cual pertenece la sección;

— ***userNetworkMessage():*** mensaje DII es almacenada;

— ***downloadDataMessage():*** mensaje DDB es almacenada.

6 Especificación del carrusel de objetos

6.1 Objetivo del carrusel de objetos

La especificación del carrusel de objetos fue añadida para brindar soporte a los servicios de transmisión de datos que requieren transmisión periódica de objetos DSM-CC U-U a través de las redes de transmisión compatibles con el sistema brasileño de televisión digital terrestre (SBDTV).

La transmisión de datos de acuerdo con la especificación del sistema brasileño de televisión digital terrestre para carrusel de objetos se transmite de acuerdo con la DSM-CC de carrusel de objetos y especificación de carrusel de datos DSM-CC que se definen en MPEG-2 DSM-CC (ver a ISO/IEC 13818-6:1998, Sección 5).

6.2 Especificación del transporte de datos

6.2.1 Dirección de carrusel NSAP

La especificación SBTVD para carrusel de objetos se basa en la especificación DSM-CC de carrusel de objetos (ver a ISO/IEC 13818-6). Un carrusel de objetos SBTVD representa un dominio de servicio particular que consiste en una colección de objetos DSM-CC U-U dentro de una red SBTVD. El dominio de servicio tiene un puerto de servicio que presenta un gráfico de servicios y nombres de objetos para los receptores.

La única identificación del puerto de servicio en las redes de transmisión se realiza por medio de la dirección *Network Service Access Point* (NSAP) del carrusel, conforme definido en DSM-CC (ver a ISO/IEC 13818-6). Esta dirección contiene una parte específica de la red que debe tornar la dirección única dentro del ambiente de red usado. La dirección NSAP del carrusel se utiliza para referirse al carrusel de objetos desde otro dominio de servicio. Para los ambientes del SBTVD, la sintaxis y la semántica de la dirección NSAP del carrusel se definen abajo.

6.2.2 Estructura de la dirección NSAP del carrusel

La dirección NSAP del carrusel tiene una estructura según definida en la Figura 2 (ver a ISO/IEC 13818-6).

AFI	***Type***	***carouselId***	**specifier**	***privateData***
1 byte	**1 byte**	**4 bytes**	**4 bytes**	**10 bytes**

Figura 2 — Formato de la dirección NSAP del carrusel

La semántica del AFI (identificador de autorización y formato), tipo, *carouselId* y especificador se definen en la ISO/IEC 13818-6. En particular:

— **AFI:** campo de 8 bits que se debe configurar en el valor de 0x00 para indicar el uso privado del formato NSAP (ver a EN 301 192);

— ***Type* (Tipo):** campo de 8 bits que se debe configurar en 0x00 para indicar el uso de la dirección NSAP para carruseles de objetos;

— ***carouselId:*** campo de 32 bits que se debe configurar en el identificador del carrusel de objetos, es decir, el campo *carouselId;*

— ***specifier* (especificador)**: campo de 32 bits que debe transportar el campo *specifierType* (configurado en el valor de 0x01) y el código OUI (Identificador Único Organizacional) según lo definido en la DSM-CC (ver ISO/IEC 13818-6:1998, Sección 5);

— ***privateData:*** campo que debe transportar la estructura *ginga_service_location* que es definida en la Tabla 20.

Tabla 20 — Sintaxis para la estructura *ginga_service_location*

Sintaxis	Número de bits	Mnemónico
ginga_service_location() {		
transport_stream_id	16	*uimsbf*
org_network_id	16	*uimsbf*
service_id	16	*uimsbf*
reserved	32	*bslbf*
}		

La semántica de la estructura *ginga_service_location* debe ser la siguiente:

— ***transport_stream_id:*** campo de 16 bits que identifica el *stream* de transporte en el cual el carrusel se transmite;

— ***org_network_id:*** campo de 16 bits que identifica el *network_id* del sistema de entrega del que se origina el carrusel;

— ***service_id:*** campo de 16 bits que suministra el identificador del servicio que contiene el carrusel de objetos. El *service_id* es lo mismo que el *program_number* en la *program_map_section* asociada.

6.3 Descriptores

NOTA Todos los descriptores del carrusel de datos son los mismos que se utilizan en la Sección 5.

6.3.1 Especificación PSI y SI

El servicio de transmisión de datos indica el uso de un carrusel de objeto SBTVD por la inclusión de uno o más descriptores de componentes de los datos – descriptor de ID del carrusel – descriptor de *tag* de asociación, de acuerdo con la ARIB STD-B23.

Cada descriptor debe indicar un carrusel de objetos y ser asociado a un *stream* particular vía un identificador *component_tag.* En particular, el valor del campo *component_tag* es idéntico al valor del campo *component_tag* de un *stream_identifier_descriptor* (ver a EN 300 468) que puede estar presente en la sección de mapa del programa PSI para el *stream* que se utiliza como *stream* de datos.

Cada descriptor de transmisión de datos permite el uso de protocolos de capas más altas basados en el criterio de lenguaje usando una lista de objetos.

Un carrusel de objeto se puede implementar usando servicios de transmisión de datos múltiples. Los servicios de transmisión de datos pueden publicar que ellos son parte de un carrusel de objeto particular por la inclusión del *carousel_identifier_descriptor* como definido por la DSM-CC (ver la ISO/IEC 13818-6) en el primer enlace de descriptor de la tabla de mapa de programa.

Además de ello, los carruseles-objetos usan el concepto de *taps* (ver la ISO/IEC 13818-6) para identificar los *streams* en los cuales los objetos se transmiten. La asociación entre los *taps* y los *stream*s del servicio de datos se puede realizar por uno u otro, usando el descriptor *association_tag* definido en (ver la ISO/IEC 13818-6) o el *stream_identifier_descriptor* en EN 300 468.

En último caso, se presume que el campo *component_tag* del descriptor *stream*_identifier sea el byte de menor significación del valor *association_tag* indicado que tiene el byte más significativo configurado en 0x00.

Finalmente, los objetos de *stream* dentro de los carruseles de objetos U-U pueden ser conectados a los *streams* elementales del servicio de transmisión de datos por ellos mismos a los *streams* elementales de otros servicios o para completar servicios SBTVD. Si el objeto de *stream* es conectado a los *stream* elementales de otros servicios o para completar los servicios SBTVD, la tabla de mapa de programa del servicio de transmisión de datos debe incluir el *deferred_association_tags_descriptor* en el primer enlace del descriptor.

6.3.2 *Deferred_association_tags_descriptor*

La sintaxis y la semántica del *deferred_association_tags_descriptor()* en las redes compatibles con el SBTVD se describen en la Tabla 21.

Tabla 21 — *Deferred_association_tags_descriptor*

Sintaxis	Número de bits	Mnemónico
deferred_association_tags_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
association_tags_loop_length	8	*uimsbf*
for (i=0;i<N1;i++) {		
association_tag	16	*uimsbf*
}		
transport_stream_id	16	*uimsbf*
program_number	16	*uimsbf*
for (i=0;i<N2;i++){		
private_data_byte	8	*uimsbf*
}		
}		

La semántica del *deferred_association_tags_descriptor* debe ser la siguiente:

— ***descriptor_tag:*** campo de 8 bits que debe tener el valor de 0x15;

— ***descriptor_length:*** campo de 8 bits que especifica la extensión del descriptor en bytes;

— ***association_tags_loop_length:*** campo de 8 bits que define la extensión en bytes del enlace de las *tags* de asociación que siguen este campo;

— ***association_tag:*** campo de 16 bits que contiene la *association_tag* que está asociada a un *stream* que no forma parte del servicio de transmisión de datos o con otro servicio SBTVD;

— ***transport_stream_id:*** campo de 16 bits que indica la s*tream* de transporte en el cual reside el servicio que está asociado a las *tags* de asociación listadas;

— ***program_number:*** campo de 16 bits que se debe configurar en el *service_id* del servicio que está asociado a las *tags* de asociación listadas;

— ***private_data_byte:*** campo que debe contener la estructura *deferred_service_location* definida en la Tabla 22.

Tabla 22 — Sintaxis para la estructura *deferred_service_location*

Sintaxis	Número de bits	Mnemónico
deferred_service_location() {		
org_network_id	16	*uimsbf*
for (i=0;i<N;i++) {		
private_data_byte	8	*uimsbf*
}		
}		

La semántica de la estructura *deferred_service_location* debe ser la siguiente:

— ***org_network_id:*** campo de 16 bits que identifica el *network_id* del sistema de entrega a partir del cual se origina el servicio;

— ***private_data_byte:*** campo de 8 bits que no se especifica en esta Norma.

6.3.3 Tipo de flujo

La presencia de un carrusel de objetos en un servicio debe ser indicada en la tabla de mapa de programa de ese servicio colocando el tipo de *stream* que contiene el carrusel de datos en el valor de 0x0B (ver ISO/IEC 13818-1)

o un valor definido por el usuario.

7 Encapsulado multiprotocolo (MPE)

7.1 Especificación de transporte de datos

Los datagramas son encapsulados en las *datagram_sections* que son compatibles con el formato *DSMCC_section* para datos privados (ver la ISO/IEC 13818-6). El mapeo de la sección dentro de los paquetes MPEG-2 de s*tream* de transporte se define en sistemas MPEG-2 (ver la ISO/IEC 13818-1).

La sintaxis y la semántica del *datagram_section* se definen en la Tabla 23.

Tabla 23 — Sintaxis del *datagram_section*

Sintaxis	Número de de bits	Mnemónico
datagram_section () {		
table_id	8	*uimsbf*
section_syntax_indicator	1	*bslbf*
private_indicator	1	*bslbf*
reserved	2	*bslbf*
section_length	12	*uimsbf*
MAC_address_6	8	*uimsbf*
MAC_address_5	8	*uimsbf*
reserved	2	*bslbf*
payload_scrambling_control	2	*bslbf*
address_scrambling_control	2	*bslbf*
LLC_SNAP_flag	1	*bslbf*
current_next_indicator	1	*bslbf*
section_number	8	*uimsbf*
last_section_number	8	*uimsbf*
MAC_address_4	8	*uimsbf*
MAC_address_3	8	*uimsbf*
MAC_address_2	8	*uimsbf*
MAC_address_1	8	*uimsbf*
if (LLC_SNAP_flag == '1') {		
LLC_SNAP()		
}		
Else{		
for (j=0;j<N1;j++) {		
IP_datagram_data_byte	8	*bslbf*
}		
}		
if (section_number == last_section_number) {		
for(j=0;j<N2;j++){		
stuffing_byte	8	*bslbf*
}		
}		
If(section_syntax_indicator == '0'){		
checksum	32	*uimsbf*
}		
else{		
CRC32	32	*rpchof*
}		
}		

La semántica del *datagram_section* debe ser la siguiente:

— ***table_id:*** campo de 8 bits que se debe configurar en 0x3E, secciones DSM-CC con datos privados (ver la ISO/IEC 13818-6:1998, Sección 5);

— ***section_syntax_indicator:*** campo que se debe configurar conforme definido en la ISO/IEC 13818-6:1998, sección 5;

— ***private_indicator:*** campo que se debe configurar conforme definido en la ISO/IEC 13818-6:1998, Sección 5;

— ***reserved:*** campo de 2 bits que se debe configurar en "11";

— ***section_length:*** campo que se debe configurar conforme definido en la ISO/IEC 13818-6:1998, sección 5;

— ***MAC_address_[1.. 6]:*** campo de 48 bits que contiene la dirección MAC del destino. La dirección MAC es fragmentada en 6 campos de 8 bits, rotulados como *MAC_address_1* a *MAC_address_6.* El campo *MAC_address_1* contiene el byte más significativo de la dirección MAC, mientras el *MAC_address_6* contiene el byte menos significativo. La Figura 3 ilustra el mapeo de los bytes de la dirección MAC en los campos de la sección.

NOTA El orden de los bits en los bytes no está reservado y el MSB (Bit Más Significativo) de cada byte también se transmite primero.

Los campos *MAC_address* contienen una dirección MAC clara o codificada, como indicado por el campo *address_scrambling_control;*

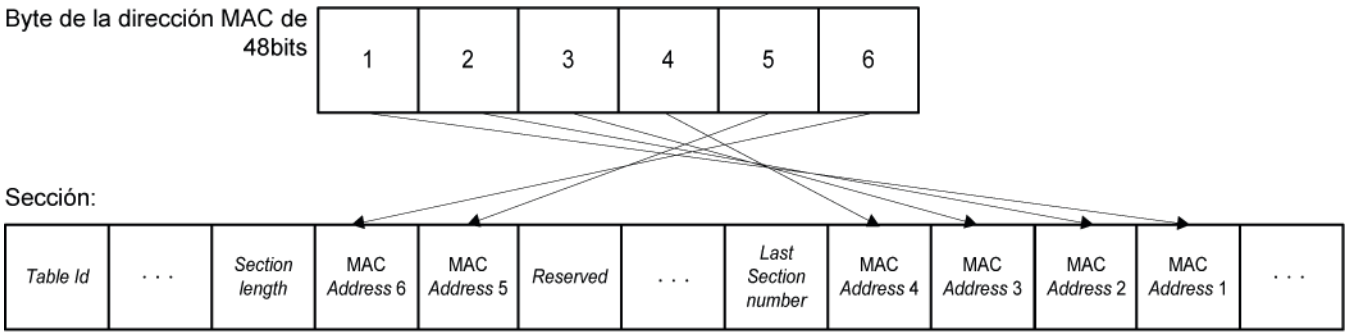


Figura 3 — Mapeo de los bytes de la dirección MAC para los campos de la sección

— ***payload_scrambling_control:*** campo de 2 bits que define el modo de codificación del *payload* de la sección. Eso incluye el comienzo del *payload* después del *MAC_address_1,* pero excluye a *checksum* o campo CRC32 (ver Tabla 24). El método de codificación aplicado es exclusivo del usuario;

Tabla 24 — Codificación del campo *payload_scrambling_control*

Valor	Control de codificación del *payload*
00	No codificado
01	Definido por el servicio
10	Definido por el servicio
11	Definido por el servicio

— ***address_scrambling_control:*** campo de 2 bits que define el modo de dispersión de la dirección MAC en esta subsección (ver Tabla 25). Este campo permite un cambio dinámico de las direcciones MAC. El método de codificación aplicado es exclusivo del usuario;

Tabla 25 — Codificación del campo *address_scrambling_control*

Valor	Control de dirección de codificación
00	No codificado
01	Definido por el servicio
10	Definido por el servicio
11	Definido por el servicio

— ***LLC_SNAP_flag:*** *flag* de 1 bit. Si el *flag* está configurado en “1”, la *payload* carga un datagrama siguiendo campo *MAC_address_1.* La estructura LLC/SNAP debe indicar el tipo de datagrama transportadas. Si el flag está configurado en “0”, la sección debe contener un datagrama IP sin encapsulado LLC/SNAPP;

— ***current_next_indicator:*** campo de 1 bit que se debe configurar en el valor de “1”;

— ***section_number:*** campo de 8 bits. Si el datagrama es transportado en secciones múltiples, entonces este campo indica la posición de la sección dentro del proceso de fragmentación. En caso contrario será configurado en cero;

— ***last_section_number:*** campo de 8 bits que debe indicar el número de la última sección usada para cargar el datagrama, es decir, el número de la última sección del proceso de fragmentación;

— **LLC_SNAP:** estructura que debe contener el datagrama de acuerdo con las especificaciones de la ISO/IEC 8802-2 LLC (Control de Conexión Lógica) y de la ISO/IEC TR 8802-1 SNAP (Punto de Anexión de la Subrede). Si la *payload* de la sección está codificada (ver *payload_scrambling_mode),* estos bytes deben estar diseminados;

— ***IP_datagram_data_byte:*** Bytes contienen los datos del datagrama. Si la *payload* de la sección está codificada (ver *payload_scrambling_mode),* estos bytes deben estar codificados;

— ***stuffing_byte:*** campo opcional de 8 bits cuyo valor no se especifica. Si la *payload* de la sección está codificada (ver *payload_scrambling_mode),* estos bytes se codifican. Deben auxiliar la codificación del bloque y procesamiento de datos en los ambientes de *wide bus.* El número de *stuffing_bytes* que se utilizan debe adecuarse a las exigencias de alineamiento de los datos definidos en el *data_broadcast_descriptor;*

— ***checksum:*** campo que se debe configurar conforme definido en la ISO/IEC 13818-6:1998, Sección 5. Es calculado sobre el *datagram_section* completo;

— **CRC_32:** campo que se debe configurar conforme definido en la ISO/IEC 13818-6:1998, Sección 5. Es calculado sobre el *datagram_section* completo.

7.2 Especificaciones PSI y SI

El servicio de transmisión de datos debe indicar la transmisión de datagramas por la inclusión de uno o más descriptores de transmisión de datos en SI (ver la ARIB STD-B23). Cada descriptor debe ser asociado a un *stream* vía un identificador *component_tag.* En particular, el valor del campo *component_tag* debe ser idéntico al valor del campo *component_tag* de un *stream_identifier_descriptor* (ver la EN 300 468:2005, Sección 2) que puede estar presente en la tabla de mapa de programa PSI (PMT) para el *stream* usado para transmitir los datagramas.

7.3 Descriptor de protocolo de transporte

El descriptor de protocolo de transporte se utiliza de la siguiente manera:

— ***protocol_id:*** campo que se debe configurar en 0x0002 para indicar el uso de la especificación de encapsulado multiprotocolo;

— ***component_tag:*** campo que debe tener el mismo valor de un campo *component_tag* de un *stream_identifier_descriptor,* que pueda estar presente en la sección de mapa de programa PSI para el *stream* en el que se transmiten los datos;

— ***selector_byte:*** bytes selectores que deben transportar la estructura *multiprotocol_encapsulation_info* que es definida en la Tabla 26.

Tabla 26 — Sintaxis para la estructura *multiprotocol_encapsulation_info*

Sintaxis	Número de bits	Mnemónico
multiprotocol_encapsulation_info() {		
MAC_address_range	3	*uimsbf*
MAC_IP_mapping_flag	1	*bslbf*
alignment_indicator	1	*bslbf*
reserved	3	*bslbf*
max_sections_per_datagram	8	*uimsbf*
}		

La semántica de la estructura *multiprotocol_encapsulation_info* debe ser la siguiente:

— ***MAC_address_range:*** campo de 3 bits que debe indicar el número de bytes de la dirección MAC que el servicio usa para diferenciar los receptores de acuerdo con la Tabla 27;

Tabla 27 — Codificación del campo *MAC_address_range*

MAC_address_range	***Bytes* de *MAC_address* válidos**
0x00	Reservado
0x01	6
0x02	6,5
0x03	6,5,4
0x04	6,5,4,3
0x05	6,5,4,3,2
0x06	6,5,4,3,2,1
0x07	Reservado

— ***MAC_IP_mapping_flag:*** *flag* de 1 bit. El servicio debe configurar ese *flag* en “1”, si usa el IP para mapeo MAC (ver la RFC 1112). Si ese *flag* está configurado en “0”, el mapeo de las direcciones IP para direcciones MAC se realiza fuera del objetivo de esta Norma;

— ***alignment_indicator:*** campo de 1 bit que debe indicar la alineamiento que existe entre los bytes del *datagram_section* y los bytes del *stream* de transporte, de acuerdo con la Tabla 28;

Tabla 28 — Codificación del campo *alignment_indicator*

Valor	**Alineamiento en bits**
0	8 (estándar)
1	32

— ***reserved:*** Campo de 3 bits que se debe configurar en "111";

— ***max_sections_per_datagram:*** Campo de 8 bits que debe indicar el número máximo de secciones que se puede usar para cargar una única unidad de datagrama.

7.4 Tipo de s*tream*

La presencia de un *stream* de datos de multiprotocolo en un servicio debe ser indicada en la sección de mapa de programa de ese servicio por la configuración del tipo de *stream* para el valor de 0x0A (ver la ISO/IEC 13818-6:1998, Sección 5) o un valor definido por el usuario.

8 Especificación de la transmisión del *data piping*

8.1 Especificación del transporte de datos

El servicio de transmisión de datos debe insertar los datos a ser transmitidos directamente en la *payload* de los paquetes MPEG-2 TS.

El servicio de transmisión de datos puede usar el campo *payload_unit_start_indicator* y el campo *transport_priority* de los paquetes de la s*tream* de Transporte MPEG-2 en forma de servicio privado. El uso del *adaptation_field* debe ser compatible con MPEG-2.

La entrega de los bits en tiempo a través de un *data pipe* es un servicio privado y no se especifica en esta Norma.

8.2 Especificaciones PSI y SI

El servicio de transmisión de datos debe indicar el uso de un *data pipe* (canal de datos), incluyendo uno o más descriptores de transmisión de datos en SI (ver la EN 300 468). Cada descriptor debe ser asociado a un canal de datos particular vía un identificador *component_tag.*

En particular, el valor del campo *component_tag* debe ser idéntico al valor del campo *component_tag* de un *stream_identifier_descriptor* (ver la EN 300 468) que se puede presentar en la sección de mapa de programa PSI para el *stream* que se utiliza como un *data pipe.*

8.3 Descriptor de protocolo de transporte

El descriptor de transmisión de datos se debe usar de la siguiente forma:

— ***protocol_id***: campo que debe ser configurado en 0x0005 para indicar un canal de datos Ginga. Los otros campos están presentes.

8.4 Tipo de s*tream*

La especificación del *stream_type* en la sección de mapa de programa debe ser 0x7E (ver la ABNT NBR 15603-2:2007, Tabla J.1),

9 Especificación de transmisión PES independiente

9.1 Transmisión independiente de PES

La especificación de transmisión PES independiente es un método utilizado para implementar el *streaming* para servicios de transmisión de datos. Hay dos tipos de especificación de transmisión PES: Sincronizada y asíncrona.

El sistema de transmisión PES sincronizada se utiliza cuando es necesario sincronizar datos en un *stream* con otros *stream*s, incluyendo video y audio. La especificación de transmisión PES asíncrona se utiliza cuando la sincronización no es necesaria. Como un ejemplo de aplicación importante, se espera que el tipo sincronizado sea utilizado para transmitir *closed caption,* y el tipo asíncrono para transmisión de caracteres superpuestos *(superimposed).* Para informaciones relacionadas a la PES independiente, ver Anexo A.

9.2 PES sincronizada

De acuerdo con la especificación de transmisión PES sincronizada, los datos se transmiten utilizando un paquete PES especificado en la ISO/IEC 13818-1. Cualquier mapeo de paquete PES para un *stream* de transporte MPEG-2 debe cumplir la ISO/IEC 13818-1.

De acuerdo con la especificación de transmisión del tipo sincronizada, un paquete PES con las siguientes restricciones se utiliza además de la sintaxis y semántica especificadas en la ISO/IEC 13818-1.

Para el encabezamiento del paquete PES correspondiente al private_*stream*_1, se deberá utilizar lo siguiente:

— ***stream_id:*** en el caso de un *stream* del tipo sincronizado, éste se debe configurar en ‘0xBD‘(private_*stream*_1);

— ***PES_packet_length:*** campo de 16 bits que debe tener un valor que no sea cero.

La estructura de datos de la PES sincronizada mostrada en la Tabla 29 se debe insertar en el campo *PES_packet_data_bytes.*

Tabla 29 — Estructura de datos de la PES sincronizada

Sintaxis	Número de bits	Mnemónico
syncronized_PES_data() {		
data_identifier	8	*uimsbf*
private_stream_id	8	*uimsbf*
reserved_future_use	4	*bslbf*
PES_data_packet_header_length	4	*uimsbf*
for (i=0; i<N1; i++) {		
PES_data_private_data_byte	8	*bslbf*
}		
for(i=0;i<N2;i++){		
syncronized_PES_data_byte	8	*bslbf*
}		
}		

La semántica de campos en un paquete PES sincronizada es:

— ***data_identifier:*** campo de 8 bits que se debe configurar en ‘0x80’;

— ***private_stream_id:*** no utilizado (0xFF);

— ***PES_data_packet_header_length:*** campo de 4 bits indica la extensión en bytes del PES_data_private_date_bytes;

— ***PES_data_private_data_byte:*** campo de 8 bits que es una utilización más detallada de este campo y depende de un servicio. Una unidad receptora puede omitir este campo;

— ***synchronized_PES_data_byte:*** campo de 8 bits conteniendo los datos transmitidos.

9.3 PES asíncrona

De acuerdo con la especificación de PS asíncrona, los datos se transmiten utilizando un paquete PS especificada en la ISO/IEC 13818-1. Cualquier mapeo de paquete PS para un *stream* de transporte MPEG-2 debe cumplir la ISO/IEC 13818-1.

De acuerdo con la especificación de transmisión asíncrona, un paquete PES con las siguientes restricciones se utiliza además de la sintaxis y semántica especificadas en la ISO/IEC 13818-1.

Para el encabezamiento del paquete PES correspondiente al *private_stream_2,* se deberá utilizar:

— ***stream_id:*** en caso de un *stream* tipo asíncrono, se debe configurar a ‘0xBF’ *(private_stream_2);*

— ***PES_packet_length:*** campo de 16 bits que debe tener un valor que no sea cero.

La estructura de datos de la PES asíncrona mostrada en la Tabla 30 se inserta en el campo del *PES_ packet_data_bytes.*

Tabla 30 — Estructura de datos de PES asíncrona

Sintaxis	Número de bits	Mnemónico
Asynchronous_PES_data() {		
data_identifier	8	*uimsbf*
private_stream_id	8	*uimsbf*
reserved_future_use	4	*bslbf*
PES_data_packet_header_length	4	*uimsbf*
for (i=0; i<N1; i++) {		
PES_data_private_data_byte	8	*bslbf*
}		
for(i=0;i<N2;i++){		
Asynchronous_PES_data_byte	8	*bslbf*
}		
}		

La semántica de campos en un paquete PES asíncrona es:

— ***data_identifier:*** campo de 8 bits que se debe configurar en ‘0x81;

— ***private_stream_id:*** no utilizado (0xFF);

— ***PES_data_packet_header_length*:** campo de 4 bits que indica la extensión en bytes del PES_data_private_date_bytes;

— ***PES_data_private_data_byte:*** campo de 8 bits que es una utilización más detallada de este área y depende de un servicio. Una unidad receptora puede omitir este campo;

— ***asynchronous_PES_data_byte:*** campo de 8 bits conteniendo los datos transmitidos.

10 Protocolos de transporte

10.1 Protocolo del canal de transmisión

10.1.1 *Stream* de transporte MPEG-2

El *stream* de transporte MPEG-2 debe estar de acuerdo con la ISO/IEC 13818-1 y ABNT NBR 15602-3.

10.1.2 Sección MPEG-2

La sección MPEG-2 debe estar de acuerdo con la ISO/IEC 13818-1 e ABNT NBR 15602- 3.

10.1.3 Datos privados DSM-CC

Los datos privados DSM-CC deben estar de acuerdo con la ISO/IEC 13818-6.

10.1.4 Carrusel de datos DSM-CC

El carrusel de datos DSM-CC debe estar de acuerdo con la ISO/IEC 13818-6.

10.1.5 Carrusel de objetos DSM-CC

Para el protocolo de transporte de transmisión que transmite contenidos de aplicación GINGA, se especifican dos sistemas: El sistema de transmisión de carrusel de objetos y el de carrusel de datos.

Cada sistema está de acuerdo con la ISO/IEC 13818-6 (para equivalencias funcionales ver Tabla 31).

Tabla 31 — Equivalentes funcionales

<table>
<tr><th>Nombre</th><th>GEM</th><th>Implementación en la ARIB STD-B23</th><th>Observaciones</th></tr>
<tr><td rowspan="3">Carrusel</td><td rowspan="3">Ver la GEM 1.0:2005, subsecciones 6.2.5 y 11.7.2</td><td>Ver la ARIB-STD-B23:2004, Anexo B</td><td>En caso de utilización de carrusel de datos, se aplica la ARIB STD-B23:2004, Anexo B Ver ARIB STD-B24</td></tr>
<tr><td colspan="2">transport_stream_id, original_network_id, service_id de dvb_service_location() en la ARIB STD-B23:2004, Tabla B.26: DVB dirección carrusel NSAP debe seguir la semántica de la ARI B-SI. Cualquier sistema estándar de carrusel es seleccionable.</td></tr>
<tr><td>ETSI TS 101 812:2003, Anexo B, e ISO/IEC 13818-6 (DSM-CC carrusel de objetos)</td><td>En caso de utilización de un carrusel de objetos, se aplica la ETSI TS 101 812:2003, Anexo B,</td></tr>
</table>

10.1.6 Protocolo IP de transporte de *multicast* en un canal de transmisión

El protocolo IP de transporte de *multicast* en un canal de transmisión debe estar de acuerdo con la EN 301 192.

10.1.7 Protocolo IP

El protocolo IP debe estar de acuerdo con la RFC 791.

10.1.8 Protocolo UDP

El protocolo UDP debe estar de acuerdo con la RFC 768.

10.1.9 Informaciones de servicio

Las informaciones de servicio deben estar de acuerdo con ABNT NBR 15603-1, ABNT NBR 15603-2 y ABNT NBR 15603-3.

10.1.10 Señalización de IP

La señalización de IP debe estar de acuerdo con la EN 301 192.

10.2 Protocolos de canal de interacción

10.2.1 Pila de protocolo del canal interactivo

Los protocolos de canal de interacción deben estar de acuerdo con la ABNT NBR 15607.

La Figura 4 ilustra el conjunto de protocolos de canal de interacción SBTVD que son accesibles por aplicativos Ginga en algunos o todos los perfiles (ver ABNT NBR 15606-1). Los detalles completos de las API que ofrecen acceso a estos protocolos de interacción están en la ETSI TS 101 812:2003, Sección 11.

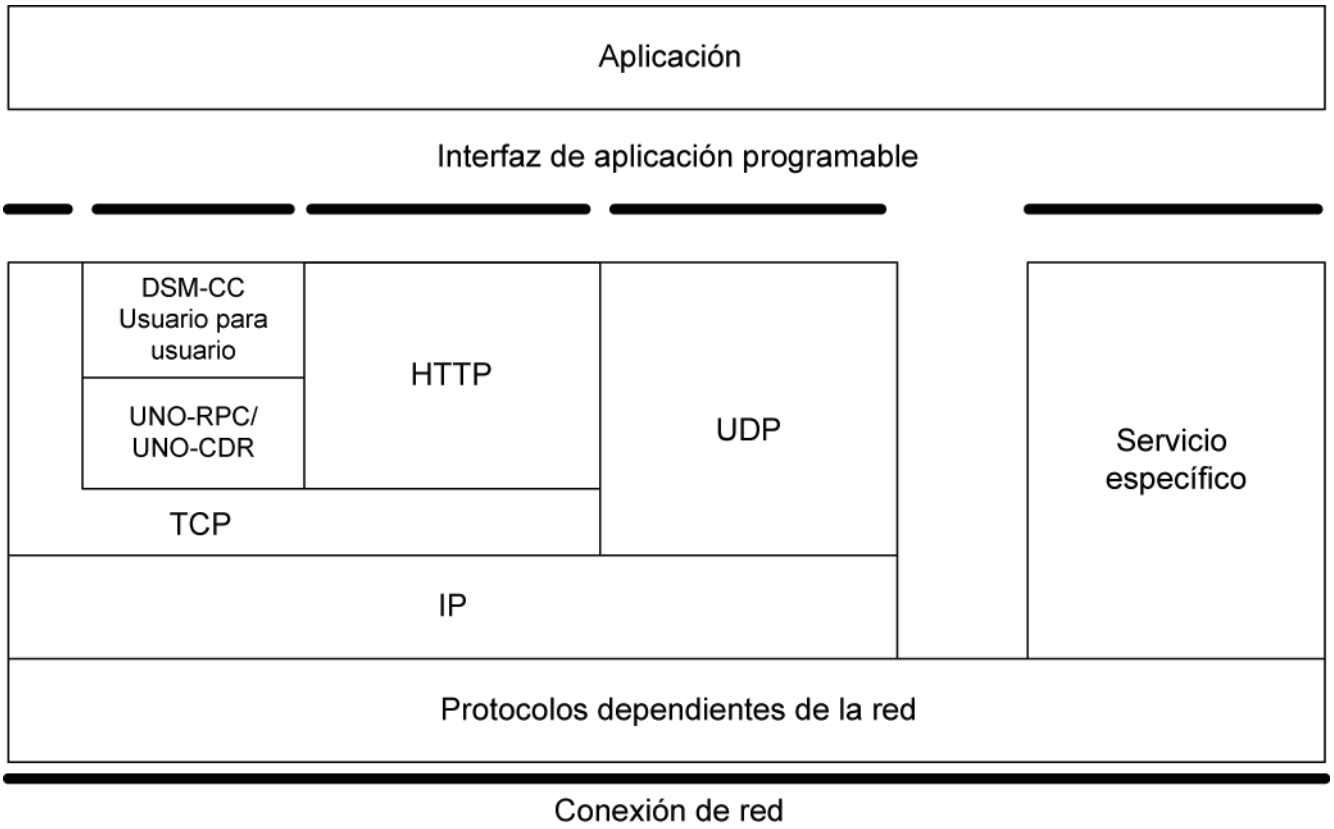


Figura 4 — Pila de protocolo del canal de interacción

10.2.2 Protocolo dependiente de la red

Los protocolos dependientes de la red debem estar de acuerdo con la ETS 300 800, ETS 300 801, EN 301 193, EN 301 195, EN 301 199, ETSI TR 101 201, EN 301 790 respectivamente para CATV, PSTN/ISDN, DECT, GSM, LMDS SMATV y redes de satélite.

Para conexiones basadas en los canales de interactividad, el protocolo PPP se utiliza según lo definido en la RFC 1332, RFC 1661, RFC 1717. Redes que utilizan direccionamiento a servidores DNS deben estar de acuerdo con la RFC 1877.

10.2.3 Protocolo de *internet* (IP)

El protocolo de internet está definido en la RFC 791.

10.2.4 Protocolo de control de transmisión (TCP)

El protocolo de control de transmisión está definido en la RFC 793.

10.2.5 UNO-RPC

El UNO-RPC consiste en *Internet Inter-ORB Protocol* (IIOP) y debe estar de acuerdo con la CORBA/IIOP.

10.2.6 UNO-CDR

EL UNO-CDR debe estar de acuerdo con la CORBA/IIOP.

10.2.7 DSM-CC usuario para usuario

El DSM-CC usuario para usuario debe estar de acuerdo con la ISO/IEC 13818-6 con restricciones y extensiones definidas en la EN 301 192 y ETSI TR 101 202.

10.2.8 Protocolo HTTP

El protocolo HTTP debe estar de acuerdo con la RFC 2616 para HTTP 1.1, RFC 1945 para HTTP 1.0 y RFC 2818 para HTTPS.

10.2.9 Protocolo específico para el servicio

El protocolo específico o propietario debe ser soportado a través de proveedores registrados para este servicio.

10.2.10 Protocolo de datagrama del usuario (UDP)

El protocolo de datagrama de usuario (UDP) debe estar de acuerdo con la RFC 768.

10.2.11 DNS

Los terminales que implementan el DNS deben estar de acuerdo con la RFC 1034 y RFC 1035, RFC 1982 y RFC 2181.

10.3 Protocolos de transporte para aplicaciones siendo cargados en el canal de interacción

Los protocolos de transporte para aplicaciones siendo cargados en el canal de interacción deben estar de acuerdo con la GEM 1.1:2006, Subsección 6.4.

El sistema de archivo implementado apenas por el canal de interacción debe estar de acuerdo con la GEM 1.0:2005, Subsección 6.4.

El híbrido entre el stream de transmisión y el canal de interacción debe estar de acuerdo con la GEM 1.0:2005, Subsección 6.4.

11 Modelo de aplicación

11.1 Aplicación Ginga

La aplicación Ginga-J debe estar de acuerdo con la ABNT NBR 15606-4. En esta Norma, este modelo se utiliza como modelo Ginga-J; para el modelo Ginga-NCL, la aplicación Ginga debe estar de acuerdo con la ABNT NBR 15606-2.

11.2 Modelo Ginga-J

El modelo se utiliza como el modelo Ginga-J y debe estar de acuerdo con la ABNT NBR 15606-4:2010, Sección 7.

11.3 Como manejar el modelo NCL

El Modelo NCL debe estar de acuerdo con los detalles de la ABNT NBR 15606-2.

11.4 Gestión de recursos entre aplicaciones

La gestión de recursos entre aplicaciones debe estar de acuerdo con la ABNT NBR 15606-2.

12 Transmisión de informaciones de aplicación

12.1 Descriptores AIT y valores constantes

Los descriptores AIT y los valores constantes deben estar de acuerdo con la ARIB STD-B23 y con la Tabla 32.

Tabla 32 — Descriptores AIT y valores constantes

Donde se utiliza	Tipo	Valor	Donde es definido	Foco	Descriptores
Data contents descriptor	*Descriptor tag*	0xC7	EIT	Ver Sección12, Anexo B, Anexo C e ARIB STD-B23	Ver ARIB STD-B23
Data coding descriptor		0xFD	PMT		
Carousel ID descriptor		0x13			
Association tag descriptor		0x14			
Extension tag descriptor		0x15			
Label descriptor	*Descriptor tag*	0x70	*DII moduleinfo*	Ver ARIB STD-B23 (carroussel de dados e objetos)	Ver GEM - *Middleware*
Caching priority descriptor		0x71			
Content type descriptor		0x72	*BIOP objectinfo*		
Reservado para el futuro descritor OC		0x73-0x7F	OC		
Application information table (AIT)	*Table* ID no PID AIT	0x74		Ver Sección12	
Application descriptor	*Descriptor tag*	0x00	AIT	Ver Sección12 e Anexo C	
Application name descriptor		0x01			
Transport protocol descriptor		0x02			
Ginga-J *application descriptor*		0x03			
Ginga-J *application location descriptor*		0x04			
External application authorisation descriptor		0x05			
Ginga-NCL *application descriptor*		0x06			
Ginga-NCL *application location descriptor*		0x07			
NCL (Reservado para el futuro)		0x08-0x0A			
Application icons descriptor		0x0B			

Tabla 32 (continuación)

Pre-fetch descriptor		0x0C			
DII *location descriptor*		0x0D			
Reservado para el futuro		0x0E - 0x010			
IP *signalling descriptor*		0x11			
Reservado para el futuro		0x12-0x5E			
Private data specifier descriptor		0x5F			
Reservado para el futuro		0x60-07F			
Definido por el usuario		0x80-0xFE			
Sistema de codificação Ginga	Sistema de codificación de datos *(data_component_id)*	A0 (Full seg) A1 (One seg)	PMT	Ver Sección12, e ARIB STD-B23	Ver ARIB STD-B23
Sistema de transmissão AIT		A3 (Full seg) A4 (One seg)			
Sistema de transmissão de carrossel de dados	Formato de transmisión *(transmission_format)*	1	Área de sistema de codificação de dados	Ver Sección12, e ARIB STD-B23	
Sistema de transmissão de carrossel de objetos		10			
Carrossel de objetos GINGA	Identificación de protocolo *(protocol_id)*	0x0001	AIT	Ver Sección12, e ARIB STD-B23	
Carrossel de dados GINGA		0x0004			
Ginga-J application type	Tipo de aplicación *(application_type)*	0x0001	AIT	Ver Sección12, e ARIB STD-B23	

12.2 Ejecución de la aplicación Ginga

Para realizar la ejecución de las aplicaciones Ginga, es necesario especificar la aplicación y transmitir a informaciones adicionales de la aplicación para controlarla.

El sistema de transmisión de la información de la aplicación para ser utilizada en esta parte de la ABNT NBR 15606 debe estar de acuerdo con la Sección 12.

Las informaciones adicionales de acuerdo con la ARIB STD-B23 son las siguientes:

— valores de identificación referentes al Ginga y la AIT, para identificar el almacenamiento del componente de datos del *additional_ ginga_info()* en a partir del *additional identifying information* no *data component descriptor* para el ES que transmite la aplicación Ginga en la PMT;

— almacenamiento del *ginga_info()* a partir del *additional information* dentro del *data contents descriptor* para ser almacenado con el área de los descriptores de un evento de programa que utilice aplicaciones Ginga en la EIT;

— almacenamiento de la *ait_identifier_info()* a partir del *additional information* dentro del *data component descriptor* para el ES que transmite la AIT en la PMT;

— en las estructuras *additional_ginga_info() y ginga_info()*, el campo *transmission_format* debe ser identificado con el valor '10', que indica el formato de transmisión como carrusel de objetos.

Además de las informaciones arriba, la *Application Information Table* (Tabla de Informaciones de la Aplicación), especificada en 12.16.1, debe ser transmitida en un ES que engloba el programa en forma de secciones privadas. Utilizada la AIT, la información de la aplicación es almacenada (ver 12.17.1) sobre la estructura de los grupos de descriptores almacenados en la AIT. Para informaciones adicionales relacionadas a las tablas PMT y EIT, ver el Anexo C.

12.3 Señalizaciones comunes de las aplicaciones

Las señalizaciones comunes de las aplicaciones son los requisitos mínimos de señalización capaz de identificar algunos aspectos como la fuente del código de la aplicación, el nombre de la aplicación, el ID de la aplicación e ID de la Organización referentes a aplicación.

Las informaciones de aplicaciones se transmiten en secciones privadas, desde la Tabla 46 (ver 12.16.1), como un ES que engloba el programa. De esta forma, se transmite la información adicional necesaria para cada aplicación.

Como complemento, los valores de identificación del componente de datos son determinados para indicar la existencia de la transmisión de la AIT, así como la aplicación GINGA, y la estructura de la selección de área del *data component descriptor* (ver 12.7 para detalles).

Los siguientes descriptores deben ser almacenados en la AIT como información común, sin llevar en consideración el formato de la aplicación:

— ***transport protocol descriptor:*** Todas las aplicaciones deben estar en el objetivo de por lo menos un *transport protocol descriptor.* Este descriptor puede ser almacenado tanto en el *common information descriptor loop* como en el *application information descriptor loop*;

— ***application descriptor:*** Un único descriptor de aplicación debe ser almacenado en un enlace de descriptor de información de aplicación para cada una de éstas;

— ***application name descriptor:*** Una única aplicación debe ser almacenada en un enlace de descriptor de información de aplicación para cada una de éstas.

12.4 Señalizaciones adicionales de las aplicaciones Ginga

Los parámetros de señalización de la aplicación y la indicación del *initial entity* son requisitos mínimos que se deben transmitir vía AIT.

En el enlace del descriptor de informaciones de aplicación AIT Ginga, los siguientes deben ser almacenados por lo menos para una aplicación.

— Ginga-J *application descriptor*;

— Ginga-J *application location descriptor*;

— Ginga-NCL *application descriptor;*

— Ginga-NCL *application location descriptor*.

12.5 Informaciones adicionales en PSI/SI

Como para las informaciones de aplicación, la Tabla 32 de informaciones de aplicación (AIT) se transmite en modo sección privada como un ES que comprende el programa. Las informaciones adicionales referentes a la transmisión de informaciones de aplicación son las siguientes:

— definición de valores de identificación correspondientes a Ginga y AIT para identificar el almacenamiento del componente de datos de *additional_ginga_info()* de informaciones adicionales, identificando las informaciones del descriptor de componente de datos para ES que transmite Ginga en la PMT;

— almacenamiento de *ginga_ info()* en información adicional dentro del descriptor de contenidos de datos que se almacenarán en el área del descriptor de un evento de programa que utiliza la aplicación Ginga en la EIT;

— almacenamiento *ait_identifier_info()* en información adicional dentro del descriptor de componente de datos para ES que transmite AIT de PMT;

— diferenciación de otros sistemas de transmisión atribuyendo 0x0004 como el *protocol_id* que corresponde a los datos de transmisión de carrusel. Para detalles del *selector_byte,* ver 12.17.6;

— atribución de un sistema de transmisión de carrusel de objetos en '10' en el campo de *additional_ginga_info()* y *ginga_info()* para identificar el sistema de transmisión de contenidos en nivel de PMT;

— en el caso de *transmission_format*=’10’ (= sistema de transmisión de carrusel de objetos), el descriptor *association_tag* (valor de *tag:* 0x14), el descriptor *deffered_Association_tag* (valor de *tag:* 0x15) o el descriptor *Carousel_id* (valor: 0x13), especificados en la ISO/IEC 13818-6, deben ser almacenados en PMT, conforme sea necesario.

12.6 Identificación del componente de datos

Un *data_component_id* es atribuido a la aplicación Ginga. Mientras tanto, un valor de identificación de componente de datos es atribuido a la transmisión AIT y un *stream* elemental a ser transferido en el modo de sección privada se agrega al PMT.

12.7 Descriptor de componente de datos y descriptor de contenidos de datos

12.7.1 Referencia indirecta

Desde el carrusel que transmite la aplicación Ginga, se debe realizar referencia indirecta con el descriptor de componente de datos relevante al sistema de codificación Ginga por el componente de *tag (component_tag.)*

12.7.2 Descriptor de componente de datos en aplicación Ginga - Sistema de codificación de datos

Cuando la identificación de la codificación de datos se hace por el sistema de codificación Ginga, la estructura *additional_ginga_info()*, según lo demostrado en la Tabla 33, se describe dentro del área de informaciones adicionales de identificación en el descriptor de componente de datos.

Tabla 33 — *Additional_ginga_info()*

Sintaxis	Número de bits	Mnemônico
additional_ginga_info() {		
transmission_format	2	*bslbf*
application_identifier_flag	1	*bslbf*
recommended_resolution	4	*bslbf*
independent_flag	1	*bslbf*
if (application_identifier_flag == 1) {		
application_identifier()	8	*bslbf*
}		
if (transmission_format == '00'){		
download_id	32	*uimsbf*
ondemand_retrieval_flag	1	*bslbf*
file_storable_flag	1	*bslbf*
event_section_flag	1	*bslbf*
reserved_future_use	5	*bslbf*
}		
else if (transmission_format == '01'){	8	*bslbf*
reserved_future_use		
}		
else if (transmission_format == '10'){		
carousel_id	32	*uimsbf*
ondemand_retrieval_flag	1	*bslbf*
file_storable_flag	1	*bslbf*
event_section_flag	1	*bslbf*
reserved_future_use	5	*bslbf*
}		
}		

La descripción del additional_ginga_j_info( ) debe ser la siguiente:

— ***transmission_format*** (formato de transmisión): el área de 2 bits especifica el sistema de transmisión de la aplicación Ginga (ver Tabla 34);

Tabla 34 — Formato de transmisión

Valor	Descripción
00	Carrusel de datos y mensaje de eventos (excepto servicio de datos sólo para almacenamiento)
00	Carrusel de datos (servicio de datos solo para almacenamiento)
10	Carrusel de objetos
11	Reservado para el futuro

— ***application_identifier_flag*** *(flag* identificador de la aplicación): flag de 1 bit que indica si el identificador de aplicación está incluido en el área de selección (ver Tabla 35);

Tabla 35 — *Flag* identificador de la aplicación

default_version_flag	Descripción
0	No utilizar el valor estándar para el número de la versión
1	Usar valor estándar para el número de la versión

— ***recommended_resolution*** (resolución recomendada): resolución de la aplicación Ginga (correspondiente a las características de resolución) y tasa de aspecto (correspondiente a las características de *display-aspect-ratio*) son indicadas en este campo. En el SBTVD el campo *recommended resolution* es especificado para dispositivos *full seg* y *one* seg y utilizado en el *loop additional_data_component_info* do *data_component_descriptor* en la Tabla AIT siendo obligatorio en los dispositivos *one seg* conforme la Tabla 36.

Tabla 36 — Sintaxis del campo r*ecommended resolution*

Sintaxis	Número de bits	*Mnemônico*
recommended_resolution	4	*bslbf*

Escoger uno de los valores de la Tabla 37.

Tabla 37 — Opciones de resolución

Valor	Descripción
0000	Aplicaciones Ginga con múltiples tamaños y resoluciones
0001	1920 x 1080 (16:9)
0010	1280 x 720 (16:9)
0011	960 x 540 (16:9)
0100	720 x 480 (16:9)
0101	720 x 480 (4:3)
0110	160 x 120 (4:3)
0111	160 x 90 (16:9)
1000	320 x 240 (4:3)
1001	320 x 180 (16:9)
1010	352 x 288 (4:3)
1011	240 x n (min = 320) (*one-seg* orientación "retrato")
1100	n (min = 320) x 240 (*one-seg* orientación "paisaje")
1101-1111	Reservado para el futuro

— ***independent_flag*** *(flag* independiente de disponibilidad de audio y video): indica si se presume que el programa de transmisión de datos sea oído y visto independientemente; 0 = Imposible y 1 = Posible.

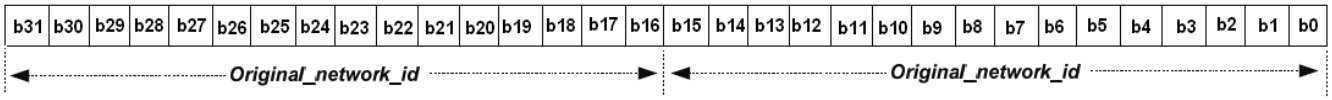
— ***application_identifier()*** (identificador de aplicación): un valor para identificar exclusivamente la aplicación. Ver Tabla 38 para los detalles;

Tabla 38 — Codificación de identificador de aplicación

Estructura de datos	Número de bits	*String* de bits
application_identifier() {	32	*bslbf*
organization_id	16	*bslbf*
application_id		
}		

organization_id (ID de la organización): campo de 32 bits que indica la organización que preparó la aplicación. Este ID almacena un número único de 32 bits atribuidos en el SBTVD por la emisora matriz y su valor es representado por los 16 bits del *network_id* a partir del bit más significativo completando los 16 bits restantes con el mismo valor;

EJEMPLO



Una emisora matriz que posee el *Original_network_id* igual a (0000000010000001)B utiliza el o*rganization_id* igual a (00000000100000010000000010000001)B identificación única para todas las emisoras de su red.

— ***application_id*** (ID de la aplicación): campo de 16 bits que almacena el número exclusivamente atribuido en el sistema para identificar la aplicación. Si la aplicación descrita por el descriptor es un servicio adicional a un programa de televisión o de radio, se utiliza para especificar la aplicación que realmente se asocia al programa de televisión o radio;

— ***download_id*** (ID de download): campo de 32 bits que sirve como rótulo que identifica el carrusel de forma única. Muestra el carrusel que debía estar montado como configuración estándar;

— ***ondemand_retrieval_flag*** (*flag* de disponibilidad de recepción de audio y video por demanda): área de 1 bit que indica, para la recepción de la aplicación transmitida por dicho ES, si la adquisición de la aplicación desde el carrusel en cada caso de la operación de audiencia está prevista. La capacidad de recepción es regulada por la operación de cada entidad de medios; 0 = No disponible y 1 = Disponible;

— ***file_storable_flag*** *(flag* de archivo almacenable): indica si el almacenamiento del archivo del programa de transmisión de datos correspondiente es posible. Por ejemplo, el almacenamiento de archivo es difícil si la información se actualiza durante el programa. La capacidad de almacenamiento es regulada por la operación de cada entidad de medios. 0 = Archivo no almacenable y 1 = Archivo almacenable;

— ***event_section_flag*** *(flag* de evento de la sección de transmisión): campo de 1 bit que indica si el mensaje de evento es distribuida por este componente. 0 = El mensaje de evento no es distribuida y 1 = El mensaje de evento es distribuida;

— ***carousel_id*** (ID del carrusel): campo de 32 bits que es el valor de identificación que especifica de forma exclusiva el carrusel de objetos. Este valor de identificación es especificado por el descriptor *carousel_id* (descriptor del identificador del carrusel), que está almacenado en el PMT.

12.7.3 Descriptor de contenidos de los datos en la aplicación Ginga - Sistema de contenido de datos

Si la identificación de la codificación de datos se hiciere conforme el sistema de codificación Ginga, la estructura *ginga_info()* demostrada en el Tabla 39 se debe describir en el área de selección del descriptor de contenidos de datos en el EIT. Eso permite que la notificación avanzada de la aplicación Ginga sea programada para uso por la unidad de evento de programa.

Las informaciones en cuanto a la aplicación Ginga y señales de controles son almacenadas en la AIT. No se presume que la aplicación sea controlada por la unidad de evento de programa. Consecuentemente, no hay mecanismo en la AIT que comprenda el cronograma por el cual la aplicación Ginga será utilizada para cada unidad de programa (ver ABNT NBR 15603-2:2007, 8.3.28).

La descripción de la *ginga_j_info( )* debe ser la siguiente:

Tabla 39 — *Ginga_info()*

Estructura de datos	Tasa de bits	*Mnemonico*
ginga_info(){		
transmission_format	2	*bslbf*
reserved_future_use	1	*bslbf*
recommended_resolution	4	*bslbf*
default_version_flag	1	*bslbf*
independent_flag	1	*bslbf*
application_identifier_flag	1	*bslbf*
content_id_flag	1	*bslbf*
associated_application_flag	1	*bslbf*
reserved_future_use	3	*bslbf*
update_flag	1	*bslbf*
ISO_639_language_code	24	*bslbf*
if (application_identifier_flag == 1) {		
application_identifier()	bslbf	
}		
if (content_id_flag==1) {		
content_id	32	*uimsbf*
content_version	16	*uimsbf*
}		
if (default_version_flag==0) {		
application_profiles_length	8	*uimsbf*
for (i=0; I<N; i++) {		
application_profile	16	*uimsbf*
profile_major_version	8	*uimsbf*
profile_minor_version	8	*uimsbf*
profile _micro_version	8	*uimsbf*
}		
}		
if (transmission_format == '00') {		
ginga_carousel_info()		
ondemand_retrieval_flag	1	*bslbf*
file_storable_flag	1	*bslbf*
reserved_future_use	6	*bslbf*
} else if (transmission_format == '01') {		
ginga_stored_carousel_info()		
} else if (transmission_format == '10') {		
ginga_object_carousel_info()		
ondemand_retrieval_flag	1	*bslbf*
file_storable_flag	1	*bslbf*
reserved_future_use	6	*bslbf*

La descripción de la ginga_j_info( ) debe ser la siguiente:

— ***transmission_format*** (formato de transmisión): área de 2 bits que especifica el sistema de transmisión de la aplicación Ginga (ver Tabla 40);

Tabla 40 — Formato de transmisión

Valor	Descripción
00	Carrusel de datos y mensaje de eventos (excepto servicio de datos sólo para almacenamiento)
00	Carrusel de datos (servicio de datos sólo para almacenamiento)
10	Carrusel de objetos
11	Reservado para el futuro

— ***recommended_resolution*** (resolución recomendada): resolución de la aplicación Ginga y display-aspect-ratio que son indicadas en la Tabla 41;

Tabla 41 — Opciones de resolución

Valor	Descripción
0000	Aplicaciones Ginga con múltiples tamaños y resoluciones
0001	1920 x 1080 (16:9)
0010	1280 x 720 (16:9)
0011	960 x 540 (16:9)
0100	720 x 480 (16:9)
0101	720 x 480 (4:3)
0110	160 x 120 (4:3)
0111	160 x 90 (16:9)
1000	320 x 240 (4:3)
1001	320 x 180 (16:9)
1010	352 x 288 (4:3)
1011	240 x n (min = 320) (*one-seg* orientación "retrato")
1100	n (min = 320) x 240 (*one-seg* orientación "paisaje")
1101-1111	Reservado para el futuro

— ***default_version_flag*** (*flag* de utilización de la versión-estándar): *flag* de 1 bit que indica que el valor-estándar especificado por la operación se utiliza como un perfil para la ejecución de la aplicación Ginga que debe ser transmitida por el ES correspondiente. Debe estar de acuerdo con la Tabla 42;

Tabla 42 — *default_version_flag*

default_version_flag	Descripción
0	No utilizar el valor estándar para el número de la versión
1	Usar valor estándar para el número de la versión

— ***independent_flag*** *(flag* independiente de disponibilidad de audio y video): indica si se presupone que el programa de transmisión de datos sea oído y visto independientemente; 0 = Imposible y 1 = Posible;

— ***application_identifier_flag*** *(flag* identificador de la aplicación): *flag* de 1 bit que indica si el identificador de aplicación está incluido en el área de selección; 0 = No Incluido y 1 = Incluido;

— ***content_id_flag*** *(flag* de ID de contenidos): *flag* de 1 bit que indica si la ID de los contenidos y su Versión están incluidos en el descriptor; 0 = No Incluido y 1 = Incluido;

— ***associated_application_flag*** *(flag* de aplicación asociada): *flag* de 1 bit que indica los contenidos asociados al programa de televisión o radio, cuando la aplicación descrita por este descriptor sea un servicio adicional de datos para el programa de televisión o radio. Para una aplicación que no sea un servicio adicional, el valor debe siempre ser 0;

— ***update_flag*** *(flag* de actualización): indica si hay distribución diferencial para esta aplicación en el futuro; 0 = No hay distribución diferencial y 1 = Hay distribución diferencial;

— ***ISO_639_language_code*** (código de lenguaje): código de lenguaje usado para la aplicación Ginga;

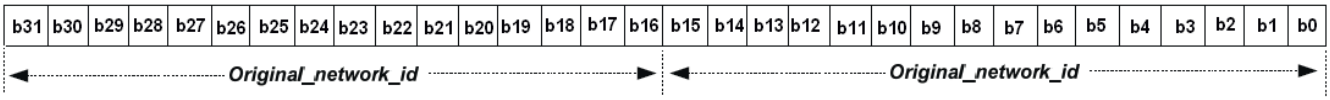
— ***application_identifier()*** (identificador de aplicación): valor para identificar exclusivamente la aplicación (ver Tabla 43);

Tabla 43 — Estructura del identificador de aplicación

Estructura de datos	Número de bits	String de bits
application_identifier() {		
organization_id	32	*bslbf*
application_id	16	*bslbf*
}		

organization_id (ID de la organización): campo de 32 bits que indica la organización que preparó la aplicación. Este ID almacena un número único de 32 bits atribuidos en el SBTVD por la emisora matriz y su valor es representado por los 16 bits del *network_id* a partir del bit más significativo completando los 16 bits restantes con el mismo valor;

EJEMPLO



— Una emisora matriz que posee el *Original_network_id* igual a (0000000010000001)B utiliza el o*rganization_id* igual a (00000000100000010000000010000001)B identificación única para todas las emisoras de su red.

— ***application_id*** (ID de la aplicación): campo de 16 bits que almacena el número que identifica la aplicación. El número es atribuido de forma exclusiva en el sistema. Cuando la aplicación descrita por el descriptor sea un servicio adicional al programa de televisión o radio, se utiliza para especificar la aplicación que de hecho se asocia al programa de televisión o radio;

— ***content_id*** (ID de contenidos): campo de 32 bits que es un rótulo que identifica el programa de transmisión de datos y es atribuido de forma exclusiva en compañía de la transmisión. En caso de almacenamiento de datos, si el *content_id* tiene el mismo valor de un programa de transmisión de datos anterior, los datos pueden ser sobrescritos;

— ***content_version*** (versión de los contenidos): campo de 16 bits que indica el número de la versión entre el programa de transmisión de datos que tiene una ID de contenidos idéntica;

— ***application_profiles_length*** (especificación de extensión del perfil de la aplicación): indica la extensión del campo que especifica el perfil receptor con el cual la aplicación es ejecutable;

— ***profile_major_version*** (número principal del perfil): campo de 8 bits que indica que el número principal entre los números de versión de los perfiles de recepción debe corresponder por lo menos a la ejecución de aplicación Ginga relevante;

— ***profile_minor_version*** (número secundario de perfil): campo de 8 bits que indica el número secundario entre los números de versión de los perfiles de recepción que debe corresponder a por lo menos la ejecución de aplicación Ginga relevante;

— ***profile_micro_version*** (micronúmero de perfil): campo de 8 bits que indica el micronúmero entre los números de versión de los perfiles de recepción que debe corresponder a por lo menos la ejecución de aplicación Ginga relevante;

— ***ginga_carousel_info( ):*** estructura de datos especificada en la ARIB STD-B24:2007, Anexo C2;

— ***ondemand_retrieval_flag*** (*flag* de disponibilidad de recepción de audio y video por demanda): área de 1 bit que indica, para la recepción de la aplicación transmitida por dicho ES, si la adquisición de la aplicación desde el carrusel en cada caso de la operación de audiencia está prevista. La capacidad de recepción es regulada por la operación de cada entidad de medios; 0 = No disponible y 1 = Disponible;

— ***file_storable_flag*** *(flag* de archivo almacenable): indica si el almacenamiento del archivo del programa de transmisión de datos correspondiente es posible. Por ejemplo, el almacenamiento de archivo es difícil si la información se actualiza durante el programa. La capacidad de almacenamiento es regulada por la operación de cada entidad de medios;

— ***ginga_ stored_carousel_info():*** estructura de datos especificada en la ARIB STD-B24:2007, Anexo C2;

— ***ginga_object_carousel_info():*** estructura de datos especificada en 12.7.4.2.

12.7.4 Descriptor de componente de datos para transmisión AIT

12.7.4.1 *Ait identifier inf*

Cuando la ID de la codificación de datos es transmisión AIT, la estructura *ait_identifier_info()* mostrada en la abla 43 debe ser descrita dentro del área de selección del descriptor de componente de datos en el PMT (ver la ABNT NBR 15603-2:2007, Subsección 7.2.3).

Tabla 44 — *Ait_identifier_info()*

Estructura de datos	Número de bits	*String* de bits
ait_identifier_info(){		
for (i=0; i<N; i++) {		
application_type	16	*uimsbf*
reserved_future_use	3	*bslbf*
AIT_version_number	5	*uimsbf*
}		
}		

La descripción de la *ait_identifier_info( )* debe ser la siguiente:

— ***application_type*** (tipo de aplicación): indica el valor del tipo de aplicación a ser transmitido en la AIT. El valor que se atibuye deve estar conforme a la Tabla 46;

— ***AIT_version_number*** (número de la versión AIT): versión actual *version_number* almacenada.

12.7.4.2 Área de selección del descriptor de contenidos de datos en la transmisión del carrusel de objetos

Cuando las informaciones para el control de recepción del carrusel de objetos están dentro del área de selección del descriptor de contenido de datos, las informaciones mostradas en la Tabla 45 deben ser agregadas en la posición del área de selección para cada esquema de codificación de datos.

Tabla 45 — Estructura Ginga_object_carousel_info()

Estructura de datos	Número de bits	String de bits
ginga_object_carousel_info(){		
num_of_carousels	8	uimsbf
for(i=0; i< num_of_carousels; i++) {		
association_tag	16	uimsbf
event_section_flag	1	bslbf
reserved_future_use	3	bslbf
Component_size_flag	1	bslbf
default_transaction_id_flag	1	bslbf
default_timeout_DSI_flag	1	bslbf
default_leak_rate_flag	1	bslbf
if (component_size_flag == ‘1’) {		
component_size	32	uimsbf
}		
if (default_transaction_id_flag == ‘1’) {		
transaction_id	32	uimsbf
}		
if (default_timeout_DSI_flag == ‘1’) {		
timeout_value_DSI	32	uimsbf
}		
if (default_leak_rate_flag == ‘1’) {		
leak_rate	22	uimsbf
reserved	2	bslbf
}		
}		
}		

La descripción de la ginga_object_carousel_info() debe ser la siguiente:

— ***num_of_carousels*** (número de carruseles): campo de 8 bits que indica el número de carruseles de objetos incluidos en una vuelta del enlace;

— ***association_tag*** *(tag* de asociación): campo de 16 bits que especifica el componente de *stream* en el cual *el* mensaje DSI con el *ServiceGatewayInfo* del carrusel de objeto es almacenada, por el *tag* de asociación que es atribuido por el descriptor *association_tag* de PMT;

— ***event_section_flag:*** con este componente, es indicada la distribución de mensaje de evento;

— ***component_size_flag*** *(flag* de tamaño de componente): campo de 1 bit que indica si el tamaño del componente se codifica en la estructura de datos. Cuando el valor del campo *component_size* aún no está definido, no está codificado (0: no codificado; 1: codificado);

— ***default_transaction_id_flag:*** campo de 1 bit que indica si la ID de transacción está codificada en la estructura de datos. Cuando la adquisición de DLL para ID opcional de transacción está especificada, la ID de transacción no está codificada (0: no codificada; 1: codificada);

— ***default_timeoutDSI_flag:*** campo de 1 bit que indica si el valor de intervalo DSI está codificado en la estructura de datos. Cuando el valor estándar definido por la operación se utiliza como el valor de intervalo DSI, no está codificado (0: No codificado; 1: Codificado);

— ***default_leak_rate_flag:*** campo de 1 bit que indica si la tasa de flujo está codificado en la estructura de datos. Cuando el valor estándar definido por la operación se utiliza como valor de tasa de flujo, no está codificado (0: No codificado; 1: Codificado);

— ***component_size*** (tamaño de componente): campo de 32 bits que indica el tamaño total de los datos (unidad: Byte) a ser transferidos por dicho carrusel de objetos;

— ***transaction_id*** (ID de la transacción): valor de ID de la transacción a ser transmitido por el componente. La ID no codificada de transacción muestra la necesidad de adquisición DSI que tiene ID de transacción opcional;

— ***time_out_value_DSI*** (valor de intervalo DSI): campo de 32 bits que indica el valor de intervalo recomendado (unidad en milisegundos) para la recepción de toda la sección DSI del carrusel relevante. Cuando el valor sea 0xFFFFFFFF, será una indicación de que no es valor de intervalo recomendable;

— ***leak_rate*** (tasa de flujo): campo de 22 bits que indica la tasa de flujo del *buffer* de transporte del receptor. La unidad es de 50 byte/s.

12.8 Localizador en descripción de aplicación

Algunos campos de la descripción de la aplicación pueden contener localizadores (*locators)* para indicación en ciertos tipos de archivos en directorios, o en transportes HTTP, url etc. El localizador utilizado en la descripción de la aplicación debe estar de acuerdo con la ARIB STD-B23:2004, Sección 14.

12.9 Descripción de aplicación

La descripción de la aplicación debe estar de acuerdo con 12.10 a 12.15. La estructura de sesión AIT que se transmitirá en el modo de transmisión de sesión privada debe estar de acuerdo con 12.16. El campo de *string* en la AIT puede ser codificado por un código de caracteres ISO/IEC 8859-15. Por otro lado, los *strings* adquiridos por el método *getName()* pueden ser automáticamente convertidos en *strings* utilizables en Java.

Los tipos de aplicaciones deben ser modificados, incluso el tipo de aplicación 0x0008, como aplicación reservada GINGA, y 0x0009, como aplicación Ginga-NCL, como mostrado en la Tabla 46.

Tabla 46 — Tipo de aplicación

Tipo de aplicación	Descripción
0x0000	Reservado
0x0001	Ginga-J
0x0002 – 0x0006	Reservado
0x0007	ARIB
0x0008	Reservado
0x0009	Ginga-NCL
0x000A – 0x7FFF	Reservado

12.10 Transmisión y monitoreo de descripción de aplicación

La descripción debe ser constantemente transmitida y monitorizada. El intervalo entre una actualización y la detección de la nueva versión por receptor no puede exceder de 30 s. En el proceso de transmisión, el tabla de la sección que está de acuerdo con la estructura AIT es transmitido como ES que representa el modo de transmisión vía sección privada del programa.

Para la operación, el valor 0x0001, 0x0008 y 0x0009 son atribuidos como el *application_type* de Ginga-J, Ginga y Ginga-NCL, y 0x0001 es atribuido como el valor de *protocol_id* para significar el sistema de transmisión de carrusel de objetos.

El *selector_byte* en el descriptor de protocolo de transporte para el sistema de transmisión del carrusel de datos debe ser de la misma sintaxis, como para el caso del “protocolo de transporte de carrusel de datos” *(protocol_id* =0x0004). Para los detalles de la estructura, ver Tabla 57.

12.11 Visibilidad de la descripción de aplicación

Mientras no se seleccione un nuevo servicio y la aplicación esté recibiendo su señalización se permite su exhibición aun si la descripción no estuviere disponible.

12.12 Detalles de la descripción de aplicación

Las descripciones de aplicación se transmiten con base en el sistema de transmisión descrito en 12.3. Los descriptores de aplicación especificados en 12.17 se deben utilizar para el almacenamiento de las descripciones de aplicación.

12.13 Tratamiento de la aplicación a partir de servicio previamente seleccionado

A partir de un servicio previamente seleccionado, si un aplicativo con un *service_bound_flag* 0 se ejecuta cuando una selección de servicios se realiza, ella debe continuar la ejecución en un nuevo servicio, si el mismo pedido fuere señalado.

12.14 Descripción de aplicación específica para el Ginga-J

La descripción de aplicación específica debe estar de acuerdo con la Tabla 46.

12.15 Detalles de la descripción de aplicación Ginga

Las descripciones de aplicación Ginga son transmitidas con base en el sistema de transmisión descrito en 12.3.

La estructura AIT y la estructura del descriptor en AIT deben estar de acuerdo con 12.6. Los descriptores de aplicación especificados en 12.18 se deben usar para el almacenamiento de los descriptores de la aplicación Ginga.

12.16 Sistema de codificación de información de aplicación

12.16.1 Información de aplicación

En la AIT, todas las informaciones relevantes a la aplicación y exigencias para *estatus* de partida están almacenadas. También es posible instruir el receptor para alterar el *estatus* de partida desde la estación de TV con los datos en la AIT.

Todas las secciones AIT que tengan el mismo *Application_Type* en el PID idéntico comprenden una subtabla. El valor de *tag* de descripción en la AIT debe ser único en la AIT.

La estructura de datos de la información de aplicación AIT es mostrada en la Tabla 47.

Tabla 47 — Tabla de información de la aplicación (AIT)

Estructura de datos	Tasa de bits	String de bits
application_information_section (){		
Table_id	8	*uimsbf*
section_syntax_indicator	1	*bslbf*
reserved_future_use	1	*bslbf*
reserved	2	*bslbf*
section_length	12	*uimsbf*
application_type	16	*uimsbf*
reserved	2	*bslbf*
version_number	5	*uimsbf*
current_next_indicator	1	*bslbf*
section_number	8	*uimsbf*
last_section_number	8	*uimsbf*
reserved_future_use	4	
common_descriptors_length	12	*uimsbf*
for (i=0, i<N; i++) {		
descriptor ()		
}		
reserved_future_use	4	*bslbf*
application_loop_length	12	*uimsbf*
for (i=0; i<N;i++) {		
application_identifier ()		
application_control_code	8	*uimsbf*
recommended_resolution	4	*bslbf*
application_descriptors_loop_length	12	*uimsbf*
for (j=0; j<M; ;j++) {		
descriptor ()		
}		
}		
CRC_32	32	*rpchof*
}		

La descripción de la *application_information_section ()* debe ser la siguiente:

— ***table_id*** (ID tabla): en este campo de 8 bits, 0x74 es almacenado para indicar que ésta es la tabla AIT;

— ***section_syntax_indicator*** (indicador de sintaxis de la sección): la indicación de sintaxis de la sección es siempre “1” en un campo de 1 bit;

— ***section_length*** (extensión de la sección): campo de 12 bits. Los primeros 2 bits deben siempre ser “00.” Esto especifica el número de bits del campo de extensión de sección para la última sección, incluyendo CRC32. El valor debe ser inferior a 1021 (0x3FD en hexadecimal);

— ***application_type*** (tipo de aplicación): campo de 16 bits que indica el valor del tipo de aplicación que está siendo transmitido por la AIT. El valor atribuido a este campo se muestra em la Tabla 46;

— ***version_number*** (número de la versión): campo de 5 bits que es el número de la versión de la subtabla. Se debe agregar uno al número de la versión, cuando se haga una alteración en la información dentro de la subtabla. Cuando el valor alcance “31,” el próximo valor será nuevamente “0.”

— ***current_next_indicator*** (próximo indicador actual): esta indicación de 1 bit es siempre “1.”;

— ***section_number*** (número de sección): campo de 8 bits que muestra el número de la sección. El número de la sección de la primera sección en la subtabla es 0x00. Cada adición de una sección que tiene la ID de tabla y tipo de aplicación idénticos añade “1” al número de la sección;

— ***last_section_number*** (número de la última sección): campo de 8 bits que especifica el número de la última sección en la subtabla a la cual las secciones pertenecen;

— ***common_descriptors_length*** (extensión del enlace de descriptores comunes): campo de 12 bits que especifica la extensión del byte del área común subsiguiente de descriptores. Los descriptores dentro del área de descriptores son aplicables a todas las aplicaciones en la subtabla AIT;

— ***application_control_code*** (código de control de la aplicación): campo de 8 bits que especifica el código de control del estatus de la aplicación. El campo es dependiente del valor de tipo de aplicación. Ver 12.16.4 para mayores detalles;

— ***recommended_resolution*** (resolución recomendada): resolución de la aplicación Ginga (correspondiente a las características de resolución) y tasa de aspecto (correspondiente a las características de *display-aspect-ratio*) son indicadas en este campo. *E*l campo *recommended resolution* es obligatorio en los dispositivos *one seg* e debe estar conforme la Tabla 36;

— ***application_loop_length*** (extensión del enlace de información de aplicación): campo de 12 bits que especifica la extensión total del enlace en byte donde se almacena la información de la aplicación subsecuente;

— ***application_identifier ()*** (identificador de aplicación): ver Tabla 48;

— ***application_descriptors_loop_length*** (extensión del enlace de descriptores de información de aplicación): campo de 12 bits que especifica la extensión del byte del área subsiguiente de descriptores. Estos descriptores en el área de descriptores se aplican solamente a la aplicación designada.

12.16.2 *Application ID* – Identificación de codificación de la aplicación

Una aplicación es identificada exclusivamente por el identificador de aplicación mostrado en la Tabla 48. Este identificador está compuesto por una estructura de 6 bytes (48 bits).

Tabla 48 — Identificador de la aplicación

Estructura de datos	Número de bits	*String* de bits
application_identifier () {		
organization_id	32	*bslbf*
application_id	16	*bslbf*
}		*bslbf*

La descripción de la *application_identifier ()* debe ser la siguiente:

— ***organization_id*** (ID de la organización): campo de 32 bits que indica la organización que creó la aplicación. Esta ID almacena el número atribuido exclusivamente en el mundo;

— ***application_id*** (ID de la aplicación): campo de 16 bits que almacena el número que identifica la aplicación y que se atribuye exclusivamente en la ID de la organización. La ID de aplicación se divide en dos fajas. Una es la faja de aplicación no suscripta y la otra es la faja de aplicación suscripta. Esta división se hace para fines de seguridad (ver Anexo B). La faja del valor se muestra respectivamente en el Tabla 49.

Tabla 49 — Valor de ID de la aplicación

Valor de ID de la aplicación	Tipo de aplicación
0x0000...0x3fff	Faja de aplicación no suscripta
0x4000...0x7fff	Faja de aplicación suscripta
0x8000...0xfffd	Reservado
0xfffe	*Wild card value* (indica todas las aplicaciones suscriptas de la misma ID de organización)
0xffff	*Wild card value* (indica todas las aplicaciones de la misma ID de organización)

En la ID de la aplicación, los valores 0xffff y 0xfffe son para *wild card value.* Éstos no se utilizan para especificar la aplicación, pero, por ejemplo, se usan como descriptor para autorización de aplicación externa. 0xffff corresponde a todas las aplicaciones que tienen la misma ID de organización *(organization_id).* 0xfffe corresponde a todas las aplicaciones firmadas que tienen la misma ID de organización.

Algunas veces el mismo identificador de aplicación se utiliza entre aplicaciones de diferentes tipos, como, por ejemplo, en el caso de la ejecución de la misma función por diferentes tipos de aplicación.

NOTA Un sólo tipo aparece en la colección de subtablas AIT del mismo tipo de aplicación en un servicio.

12.16.3 Efecto sobre el ciclo de vida

La directriz básica del efecto sobre el ciclo de vida se presenta para la ocasión en la que el servicio es alterado o en que las aplicaciones que tienen el mismo identificador de aplicación son iniciadas, de la siguiente manera:

— en el *changeover* de servicio, si el *service_bound_flag* en la aplicación activa en el servicio anterior es “0” y el identificador de la aplicación existe en la AIT del servicio recientemente seleccionado, entonces la aplicación funciona continuamente, a menos que haya alguna restricción de recurso;

— en el *changeover* de servicio, si el *service_bound_flag* en la aplicación activa en el servicio anterior es “0” y si sólo la aplicación está en la lista de descriptores de autorización de aplicación externa, la aplicación trabaja continuamente, incluso si la aplicación no forma parte del servicio recientemente seleccionado, a menos que haya alguna restricción de recurso;

— como para una aplicación que tiene un identificador de aplicación, sólo una instancia es activada. Incluso si otra aplicación tiene el mismo identificador, no puede ser activada si una aplicación con el mismo identificador de aplicación ya ha existido;

— eso afecta el comportamiento de la API de inicio de aplicación o inicio automático de la aplicación. Si el *service_bound_flag* “1” es configurado para la aplicación, terminará *(KILL)* a cada selección de servicio.

12.16.4 Control de aplicaciones de ciclo de vida

Para *Application Life Cycle Control,* mecanismos de señalización se deben suministrar desde la estación de transmisión, para controlar el ciclo de vida de las aplicaciones del tipo estándar.

12.16.5 Acceso y salida del dominio de la aplicación

Entering and Leaving the Application Domain Application es el dominio definido como una colección de servicios que tiene aplicaciones listadas en la AIT. Eso significa que las aplicaciones son las listadas en los enlaces de información de aplicación de AIT o las listadas en los descriptores de autorización de aplicación externa. Los servicios cuyas aplicaciones no están listadas como arriba mencionado son considerados fuera del dominio de la aplicación.

12.16.6 Control dinámico del ciclo de vida de las aplicaciones Ginga

El control dinámico del ciclo de vida de las aplicaciones Ginga debe estar de acuerdo con 12.16.1 a 12.16.4.

Ginga define el *application_control_code*" valor 0x07 como aplicaciones UNBOUND (ver Tabla 50).

Tabla 50 — Valores de código de control de aplicación Ginga

Código	Identificador	Semántica
0x00		Reservado para uso futuro
0x01	AUTOSTART	Aplicativos con el código de control AUTOSTART son iniciados automáticamente cuando el receptor cambia para este servicio
0x02	PRESENT	Los aplicativos con el código de control PRESENT no son iniciados automáticamente, pero son adicionados a la lista de aplicativos disponibles del receptor. El usuario puede entonces escoger iniciar este aplicativo seleccionándolo en la lista
0x03	DESTROY	Aplicativos con el código de control DESTROY deben ser finalizados por el gestionador de aplicaciones así que posible. Caso sea lanzada una excepción del tipo *javax.microedition.xlet.XletStateChangeException* durante un intento de finalización el aplicativo debe continuar en ejecución. Aplicativos señalizados previamente con código de control STORE pueden opcionalmente ser mantenidos en el *cache*
0x04	KILL	Aplicativos con el código de control KILL deben ser finalizados por el gestionador de aplicaciones así que posible. Aplicativos señalizados previamente con código de control STORE deben ser removidos del *cache*
0x05	PREFETCH	La aplicación NCL se carga y la máquina Ginga NCL es preparada. La aplicación espera por un comando de edición NCL [a] antes de pasar para el estado activo
0x06	REMOTE	Identifica un aplicativo remoto que solamente es lanzado tras la selección del servicio
0x07	UNBOUND	Aplicativos con código de control UNBOUND son similares a aplicativos señalados con PRESENT, excepto que el usuario decide si la aplicación se puede almacenar para ejecución posterior. Caso el receptor no posea capacidad de almacenamiento disponible para almacenar la aplicación o el usuario escoja por no instalar la aplicación, esta se debe tratar como no disponible (no puede ser listada entre las aplicaciones disponibles). Receptores sin soporte a almacenamiento de aplicaciones deben ignorar las aplicaciones señaladas con este código de control
0x08	STORE	Aplicativos con código de control STORE no pueden ser iniciados automáticamente, pero indican qué técnicas de *caching* se pueden utilizar de manera a acelerar la carga de sus recursos durante el inicio caso la plataforma no tenga capacidad para realizar técnicas de *caching* operante o almacenamiento de datos, aplicaciones señaladas como STORE se deben tratar de manera idéntica a aplicaciones señaladas con PRESENT
0X09 - 0Xff		Reservado para uso futuro
[a] Ver ABNT NBR 15606-2, sección 9,1, Tabla 56.		

12.17 Descriptores para AIT - Descriptores para transmisión de informaciones de las aplicaciones

12.17.1 Descriptores comunes

Los descriptores que usualmente se utilizarán en la AIT, independientemente del tipo de aplicación, son descritos en 12.17.2 a 12.17.12.

12.17.2 Descriptor de aplicación

Un descriptor de aplicación es almacenado en el enlace del descriptor de información de la aplicación AIT para cada aplicación (ver Tabla 51).

Tabla 51 — Estructura del descriptor de aplicación

Estructura de datos	Número de bits	*String* de bits
application_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
application_profiles_length	8	*uimsbf*
for (i=0; i<N ; i++) {		
application_profile	16	*uimsbf*
version.major	8	*uimsbf*
version.minor	8	*uimsbf*
version.micro	8	*uimsbf*
}		
service_bound_flag	1	*bslbf*
visibility	2	*bslbf*
reserved_future_use	5	*bslbf*
application_priority	8	*uimsbf*
for (i=0; i<N ; i++) {		
transport_protocol_label	8	
}		
}		

La descripción de la *application_descriptor* () debe ser la siguiente:

— ***descriptor_tag*** *(tag descriptor):* campo de 8 bits; 0x00 es almacenado para indicar que éste es el descriptor mencionado;

— ***application_profiles_length*** (extensión de información del perfil de la aplicación): Campo de 8 bits que indica la extensión total de información del perfil de la aplicación que está incluida en el enlace subsiguiente;

— ***application_profile*** (perfil de la aplicación): campo de 16 bits. El perfil de la aplicación que puede ejecutar la aplicación es almacenado. Si el perfil es montado en el receptor, significa que la aplicación es ejecutable. Los detalles de perfil se definen para cada tipo de aplicación;

— ***version.major*** (versión principal): campo de 8 bits que indica la versión principal del perfil arriba mencionado;

— ***version.minor*** (versión secundaria): campo de 8 bits que indica la versión secundaria del perfil arriba mencionado;

— ***version.micro*** (microversión): campo de 8 bits que indica la microversión del perfil arriba mencionado;

Los cuatro primeros campos descritos arriba comprenden un perfil mínimo para ejecución de esta aplicación. El terminal Ginga inicia esa aplicación si cualquiera de las siguientes fórmulas teóricas es aplicable y cualquier perfil evidencial “verdadero” en la información de perfil de la aplicación:

(perfil de la aplicación Є recolecta de los perfiles montados en el terminal)

Y {(versión principal de la aplicación < versión principal del terminal para el perfil)

O [(versión principal de la aplicación – versión principal del terminal para el perfil)

Y ({versión secundaria de la aplicación < versión secundaria del terminal para el perfil }

O {[versión secundaria de la aplicación = versión secundaria del terminal para el perfil]

Y [microversión de la aplicación ~ microversión del terminal para el perfil]})] }

Las definiciones del perfil detallado de la plataforma en aplicación Ginga-J y Ginga-NCL deben estar de acuerdo con la ABNT NBR 15606-1.

La codificación del perfil en aplicación Ginga-J debe estar de acuerdo con la ABNT NBR 15606-4.

La codificación del perfil de la aplicación Ginga-NCL debe estar de acuerdo con la ABNT NBR 15606-2.

— ***service_bound_flag*** *(service bound flag):* campo de 1 bit que indica si la aplicación está efectiva apenas en el presente servicio. Si el campo es “1,” la aplicación es relevante apenas al servicio actual. Cuando el servicio es alterado para otro servicio, la finalización del procesamiento de la aplicación mencionada será iniciada;

— ***visibility*** (visibilidad): campo de 2 bits que indica si la aplicación es visible a los usuarios finales cuando está activada. Las definiciones de estatus para el valor de visibilidad son mostrados en la Tabla 52;

— ***application_priority*** (prioridad de la aplicación): campo de 8 bits que indica la prioridad relativa entre las aplicaciones notificadas en el servicio;

— ***transport_protocol_label*** (rótulo de protocolo de transporte): campo de 8 bits que almacena el valor para identificación única del protocolo de transporte que transporta la aplicación. El valor corresponde al valor del campo *transport_protocol_label* del descriptor del protocolo de transporte.

Tabla 52 — Visibilidad

Valor visible	Descripción
0	Esta aplicación no es visible a las otras aplicaciones vía enumeración de aplicación API, ni a los usuarios vía navegador, excepto en caso de error de información de *logout* y análogos
1	Esta aplicación no es visible para los usuarios, pero es visible desde otras aplicaciones vía enumeración de aplicación API
10	Reservado para uso futuro
11	Esta aplicación es visible tanto para los usuarios como para las otras aplicaciones

12.17.3 Descriptor del nombre de aplicación

Un descriptor de nombre de la aplicación es almacenado para cada aplicación en el enlace del descriptor de información de la aplicación AIT. El nombre de la aplicación se utiliza para diferenciación y suministra la información a los usuarios (ver Tabla 53).

Tabla 53 — Estructura del descriptor de nombre de la aplicación

Estructura de datos	Número de bits	*String* de bits
application_name_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
for (i=0; i<N ; i++) {		*uimsbf*
ISO_639_language_code	24	*bslbf*
application_name_length	8	*uimsbf*
for (j=0; j<application_name_length ; j++) {		
application_name_char	8	*uimsbf*
}		
}		
}		

La descripción de la *application_name_descriptor ()* debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag):* campo de 8 bits; 0x01 es almacenado para indicar que éste es el descriptor mencionado;

— ***ISO_639_language_code*** (código del lenguaje): campo de 24 bits que identifica el lenguaje del descriptor del nombre de la aplicación. El código del lenguaje es indicado por el código *three-alphabetical-letter* como dispuesto en la ISO 639-2. Cada letra se codifica en 8 bits, de acuerdo con la ISO 8859-1, e inserida de acuerdo con el orden en el campo de 24 bits;

— ***application_name_length*** (descripción de la longitud del nombre de la aplicación): campo de 8 bits que indica la longitud del byte de la subsiguiente descripción del nombre de la aplicación;

— ***application_name_char*** (descripción del nombre de la aplicación): Información y documentación - Trabajos académicos - Este texto de descripción es usado como información de la aplicación por los usuarios y no puede tener valor nulo.

12.17.4 Descriptor de la información de los iconos de la aplicación

El descriptor de la información de los iconos de la aplicación es almacenado apenas una vez para cada aplicación (ver Tabla 54). El descriptor indica la información sobre el icono relevante para la aplicación. El formato de los contenidos del icono se codifica por el PNG y usa un sistema proporcionado de acuerdo con la ABNT NBR 15606-1.

Tabla 54 — Estructura del descriptor de la información de los iconos de la aplicación

Estructura de datos	Número de bits	*String* de bits
application_icons_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
icon_locator_length	8	*uimsbf*
for (i=0; i<N ; i++) {		
icon_locator_byte	8	*uimsbf*
}		
icon_flags	16	*bslbf*
for (i=0; i<N ; i++) {		
reserved_future_use	8	*uimsbf*
}		
}		

La descripción de la application *application_icons_descriptor* () debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag):* campo de 8 bits que almacena 0x0B, que indica el descriptor mencionado;

— ***icon_locator_length*** (longitud del localizador de icono): campo de 8 bits que indica la longitud de byte del subsiguiente localizador del icono;

— ***icon_locator_byte*** (localizador de icono): campo de 8 bits que almacena el localizador del archivo de imagen estática como un icono. Este localizador debe ser colocado antes del nombre del archivo de icono y debe estar de acuerdo con 12.16.1, 12.16.2, 12.16.3 e 12.16.5. Para la aplicación Ginga-J, existen reglas en el caso en el que *application_type* sea 0x0008 y hay un camino relativo de la base del directorio definido por la aplicación Ginga-J almacenado;

— ***icon_flags*** **(icono de *flag):*** este campo de 16 bits almacena la *flag* que indica el tamaño y el aspecto de la relación del icono usable. Los detalles deben estar de acuerdo con la 12.16.1, 12.16.2, 12.16.3 e 12.16.5, y la codificación de cada caso es según mostrado en la Tabla 55. El valor es almacenado después del OR (dirección lógica) por la unidad.

Tabla 55 — Bits del icono de la *flag*

Bits de icono de *flag*	Tamaño de los iconos y relación de aspecto
0000 0000 0000 0001	32 x 32 para exhibir *pixels* cuadrados
0000.0000.0000.0010	32 x 32 para transmitir *pixels* en una pantalla 4:3
0000.0000.0000.0100	24 x 32 para transmitir *pixels* en una pantalla 16:9
0000.0000.0000.1000	64 x 64 para exhibir *pixels* cuadrados
0000.0000.0001.0000	64 x 64 para transmitir *pixels* en una pantalla 4:3
0000.0000.0010.0000	48 x 64 para transmitir *pixels* en una pantalla 16:9
0000.0000.0100.0000	128 x 128 para exhibir *pixels* cuadrados
0000.0000.1000.0000	128 x 128 para transmitir *pixels* en una pantalla 4:3
0000 0001 0000 0000	96 x 128 para transmitir *pixels* en una pantalla 16:9
xxxx xxx0 0000 0000	*reserved_future_use*

El nombre del archivo del símbolo es codificado de acuerdo con el valor del símbolo de la *flag* descripto arriba. La codificación del nombre del archivo sigue en la descripción a continuación:

filename = icon_locator "/ginga.icon." hex_string

hex_string = 4*4hex

hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

Cada archivo debe contener un símbolo. El símbolo contenido en el archivo debe tener el formato especificado por los 4 dígitos *postscript* de este nombre del archivo. En el caso del campo *icon_flags* del *application_icons_descriptor* indicando la presencia de múltiples símbolos, cada uno de los archivos indicados deben tener su propio símbolo.

EJEMPLO icon_flags con valor 0x0005, el directorio especificado por el icon_locator contiene 2 archivos nombrdos ginga.0004(para 24x32 square pixel) y ginga.0001 (para 32x32 square pixel)

12.17.5 Descriptor de autorización de aplicación externa

El descriptor de autorización de aplicación externa puede ser almacenado en uno o más en el primer enlace descriptor común del AIT, conforme necesidad. En cada descriptor hay información relevante almacenada de las aplicaciones externas. Una aplicación externa es aquélla que puede operar continuamente con las aplicaciones listadas en la subtabla AIT, pero no pueden ser activadas del servicio mencionado.

La autorización externa es aplicable para la aplicación que tiene application_identifier() en la application_type especificada por la AIT que incluye este descriptor (ver Tabla 56).

Tabla 56 — Estructura del descriptor de autorización de aplicación externa

Estructura de datos	Número de bits	*String* de bits
external_application_authorisation_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
for (i=0; i<N ; i++) {		
application_identifier ()		
application_priority	8	*uimsbf*
}		
}		

La descripción del *external_application_authorisation_descriptor ()* debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag):* campo de 8 bits donde 0x05 es almacenado para indicar el descriptor de autorización de aplicación.

— ***application_identifier()*** (identificador de aplicación): campo de 48 bits que indica la aplicación en la que la referencia externa está disponible. Para detalles de la estructura del campo, ver 12.16.1;

— ***application_priority*** (prioridad de la aplicación): campo de 8 bits que indica la prioridad de la presente aplicación en la suposición del contexto del servicio mencionado. Para detalles de la prioridad, ver 12.17.2.

12.17.6 *Transport protocol descriptor* (descriptor de protocolo de transporte)

El *transport protocol descriptor* indica la identificación del protocolo de transporte relevante para el componente del servicio y almacena información sobre el protocolo. Este protocolo es almacenado en el primer enlace del descriptor común o enlace del descriptor de la información de la aplicación. Cuando esto es almacenado en el enlace de descriptor común, es aplicable para todas las subtablas del AIT. El descriptor del protocolo de transporte en el enlace del descriptor de la información de la aplicación describe el protocolo de transporte adicional para ser usado específicamente en la aplicación (ver Tabla 57).

Tabla 57 — Estructura del descriptor de protocolo de transporte

Estructura de datos	Número de bits	*String* de bits
transport_protocol_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
protocol_id	16	*uimsbf*
transport_protocol_label	8	*uimsbf*
for (i=0; i<N ; i++) {		
selector_byte	8	*uimsbf*
}		
}		

La descripción del *transport_protocol_descriptor* () debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag):* campo de 8 bits que almacena 0x02, que indica que éste es el actual descriptor de transporte;

— ***protocol_id*** (identificación de protocolo): campo de 16 bits que indica el protocolo que transporta la aplicación (ver Tabla 58);

Tabla 58 — Valores de la identificación de protocolo

Valores	Descripción
0x0000	Reservado para el futuro
0x0001	Protocolo de transporte del carrusel de objetos
0x0002	IP a través de multiprotocol y encapsulado definido en la EN 301192 y ETSI TR 101202
0x0003	Transporte vía HTTP sobre el canal de interactividad según lo descrito en la sección 12.17.9
0x0004	Protocolo de transporte del carrusel de datos
0x0005...0xFFFF	Reservado para el futuro

— ***transport_protocol_label*** (rótulo de protocolo de transporte): Campo de 8 bits que indica el valor que identifica únicamente el protocolo de transporte en la sección AIT. Para el descriptor de la aplicación, ver el dispositivo de conexión (por ejemplo, el *stream* elemental del carrusel) que transporta la aplicación con estos valores;

— ***selector_byte*** (selector de área): Campo de 8 bits que almacena información adicional proveída por cada identificación de protocolo. La estructura de datos descrita en el área se especifica por separado por cada *protocol_id* (ver Tabla 59).

Tabla 59 — Estructura del *selector_byte*

Valores del *Protocol_id*	Estructura del seletor de área
0x0000	Reservado para el futuro
0x0001	Ver 12.17.7
0x0002	Ver 12.17.8
0x0003	Ver 12.17.9
0x0004	Ver 12.17.7
0x0005...0xFFFF	Reservado para el futuro

12.17.7 Transporte a través del OC (carrusel de objeto)

La estructura de datos se muestra en la Tabla 60 para el protocolo de transporte del carrusel de objetos (protocol_id=0x0001) y el protocolo de transporte del carrusel de datos (protocol_id=0x0004).

Tabla 60 — Estructura del protocolo de transporte del descriptor del selector de área (en el caso de protocolo de transporte del carrusel de objetos/protocolo de transporte de carrusel de datos)

Estructura de datos	Número de bits	*String* de bits
remote_connection	1	*bslbf*
reserved_future_use	7	*bslbf*
if (remote_connection == “1”) {		
original_network_id	16	*uimsbf*
transport_stream_id	16	*uimsbf*
service_id	16	*uimsbf*
}		
component_tag	8	*uimsbf*

La descripción debe ser la siguiente:

— ***remote_connection*** (conexión remota): si este campo de 1 bit está en “1,” esto muestra que el componente de servicio vigente se transmite de otra fuente distinta a la que transmite para AIT. Un servicio como éste no se ejecuta automáticamente, pero es visible y viable de iniciarse a través de la API. Además de ello, *REMOTE* es almacenado de esa manera en la aplicación en *application_control_code*;

— ***original_network_id*** (identificador de la red original): cuando *remote_connection* está en “1,” el identificador de la red original para el servicio de transmisión vigente es almacenado;

— ***transport_stream_id*** (identificador de transport *stream):* cuando *remote_connection* es “1,” el identificador de *transport stream* de la transmisión de servicio actual es almacenado;

— **service_id** (identificador de servicio): cuando *remote_connection* está en “1,” el servicio (identificado por el identificador de servicio) de la transmisión vigente es almacenado;

— ***component_tag*** (componente de *tag):* indica el componente del servicio principal que transmite la aplicación. En el caso de carrusel de datos, el *stream* elemental que transmite el carrusel automáticamente montado en el inicio de la aplicación es indicado. En el caso de carrusel de objetos, el *stream* elemental que transmite DSI es indicado.

12.17.8 Transporte a través de IP

Cuando el identificador de protocolo es 0x0002 el selector de bytes en el descriptor del protocolo de transporte debe estar de acuerdo con la Tabla 61, para proveer toda la información necesaria para la obtención de las aplicaciones Ginga y componentes de datos de las aplicaciones entregadas por el protocolo de IP.

Tabla 61 — Sintaxis del selector de bytes para el transporte de IP

Estructura de datos	Número de bits	*String* de bits
remote_connection	1	*bslbf*
reserved_future_use	7	*bslbf*
if (remote_connection == "1") {		
original_network_id	16	*uimsbf*
transport_stream_id	16	*uimsbf*
service_id	16	*uimsbf*
}		
alignment indicator	1	*bslbf*
reserved for future use	7	*bslbf*
for (i=0; i<N; i++) {		
URL_length	8	*uimsbf*
for (j=0; j<URL length; j++) {		
URL_byte }	8	*uimsbf*
}		

La descripción debe ser la siguiente:

— ***remote_connection:*** este y los tres campos asociados *(original_network_id, transport_stream_id* y *service_id)* tienen semántica y sintaxis idénticas a los campos con los mismos nombres de acuerdo con 12.17.7;

— ***alignment_indicator:*** campo de 1 bit que indica la alineamiento que existe entre los bytes del *datagram_section* y los bytes de transporte de *stream;*

— ***URL_length:*** campo de 8 bits que indica el número de bytes en la URL;

— ***URL_byte:*** estos bytes forman una URL conforme RFC 2396.

Para la URL usando el campo del "servidor" incluyendo la notación *host:port,* conforme definido en la RFC 2396, solamente direcciones numéricas IP serán usadas para identificar las transmisiones de IP transportadas en el

canal de difusión conforme no hay *Domain Name Service*-Servicio del Nombre de Dominio en el escenario solamente de difusión a ser usado para nombres en resolución. IP para mapeo MAC se hará tal como se describe en la RFC 1112.

NOTA Esta Norma no define el formato de URL a ser mantenido por este descriptor. Por eso, el formato de URL no se puede usar de manera interoperable.

12.17.9 1Transporte vía canal de interactividad

Cuando el valor del protocolo ID es 0x0003, el *selector byte* en el descriptor de protocolo de transporte (*Transport protocol descriptor*) debe estar de acuerdo con el Tabla 62. De esta forma se permite codificar un número de URL. Para eficiencia en la codificación de URL similares, la codificación divide la URL dentro de una parte común y un conjunto de extensiones de URL.El conjunto de URL puede identificar archivos ZIP o URL bases terminadas en el carácter "/", que encapsula parte del sistema de archivos.

Múltiples descriptores (*Transport protocol descriptor*) con el valor de ID 0x0003 con el mismo rótulo de protocolo de transporte se pueden proveer para definir un conjunto mayor de URL para descripción del sistema de archivos.

Tabla 62 — Sintaxis del *selector bytes* para el transporte en el canal de interactividad

Sintaxis	Bits	Mneumonic
for(i=0; i<N; i++){		
URL_base_length	8	*uimsbf*
for (j=0; j<N; j++){		
URL_base_byte	8	*uimsbf*
}		
URL_extension_count	8	*uimsbf*
for(j=0; j<URL_extension_count; j++){		
URL_extension_length	8	*uimsbf*
for(k=0; k<URL_length; k++){		
URL_extension_byte	8	*uimsbf*
}		
}		
}		

La sintaxis detallada del selector bytes para el transporte en el canal de interactividad es:

— ***URL_base_length***: este campo de 8 bits provee el número de bytes en la parte de la URL base;

— ***URL_base_byte:*** estos bytes de la primera parte de una URL HTTP de acuerdo con el HTTP 1.0 (ver RFC 1945), o la primera parte de otra URL HTTPS de acuerdo con la HTTPS (ver RFC 2818). Cuando una HTTP URL es encontrada, HTTP se debe usar de acuerdo con la RFC 2616.

Cuando la URL HTTPS es encontrada, HTTPS se debe usar de acuerdo con la RFC 2818.

Para cualquier otra URL, la implementación proveedora de servicio de transporte de protocolos en el canal de interactividad registrada se usa para el esquema de URL asociada, caso exista. En caso de no existir proveedor para el esquema, el descriptor *transport protocol descriptor* es ignorado.

— ***URL_extension_count***: este campo de 8 bits indica el numero de URL relacionados por el descriptor;

— ***URL_extension_length***: este campo de 8 bits indica el número de bytes en la parte de extensión de la URL;

— ***URL_extension_byte***: estos bytes forman parte posterior de una URL HTTP de acuerdo con el HTTP 1.0 (ver RFC 1945), o la parte posterior de una URL HTTPS de acuerdo con la RFC 2818 u otra URL en el cual el esquema es soportado por una implementación proveedora de servicio de transporte de protocolos en el canal de interactividad registrada.

URL son formadas por la concatenación de extensión de URL con el URL base anterior. La URL formada o identifica el directorio del sistema de archivo o el archivo ZIP especifico.

12.17.10 Descriptor de señalización de IP

El descriptor de señalización IP se define para el uso tanto en el “común” como en la “aplicación” del enlace de la AIT. Este descriptor indica la identificación de la organización que provee los *streams multicast* usados por todas las aplicaciones (cuando está presente en el enlace “común”) o por la aplicación de señalización particular (cuando está presente en el enlace de la “aplicación”). Para la definición del INT, ver EN 301 192.

Este descriptor y el INT con *action_type* 0x01 se deben utilizar por las aplicaciones confiando en la presencia de los *streams multicast* IP en el *link* de la difusión. El conocimiento de la identificación presente en el descriptor habilita la recuperación de la tabla apropiada de notificación de IP (INT) con *action_type* 0x01 que contiene la correspondencia entre la dirección del IP *multicast, port* y localización del *stream* (ver Tabla 63).

Tabla 63 — Sintaxis del descriptor de señalización de IP

Sintaxis	Bits	Mnemónicos
ip_signalling_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
platform_id	24	*uimsbf*

La descripción del *ip_signalling_descriptor ()* debe ser la siguiente:

— ***descriptor_tag:*** campo de 8 bits que con valor 0x11 identifica el descriptor;

— ***descriptor_length:*** campo de 8 bits que identifica el número de bytes, siguiendo la longitud del campo;

— ***platform_id:*** campo de 24 bits conteniendo un *platform_id* de la organización proveyendo *streams* IP/MAC en el transporte de *streams*/servicios. Asignaciones del valor de *platform_id* son encontradas en el ETSI TR 101 162.

12.17.11 *Pre-fetch descriptor* (descriptor de pre-busca)

Solamente un descriptor *pre-fetch* es almacenado en el enlace del descriptor de la información de la aplicación AIT, según la necesidad. Este descriptor se define para ser usado por el carrusel de objetos (protocol_id=0x0001). Cada descriptor es asociado a un descriptor de protocolo de transporte a través del rótulo del protocolo de transporte (ver Tabla 64).

El terminal Ginga puede adquirir adelantadamente módulos denotados por el rótulo terminal para acelerar el tiempo de inicio de la aplicación. Así como para los rótulos, los descriptores de rótulos especificados por la transmisión del carrusel de objetos son usados de acuerdo con la ISO/IEC 13818-6:1998, Anexo B. La difusión de esta señalización es opcional.

Tabla 64 — Estructura del descriptor de *pre-fetch*

Estructura de datos	Número de bits	*String* de bits
prefetch_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
transport_protocol_label	8	*uimsbf*
for (i=0; i<N ; i++) {		
label_length	8	*uimsbf*
for (j=0; j<label_length ; j++) {		
label_char	8	*uimsbf*
}		
prefetch_priority	8	*uimsbf*
}		
}		

La descripción del *prefetch_descriptor ()* debe ser la siguiente:

— ***descriptor_tag*** *(descriptor tag):* campo de 8 bits donde 0x0C es almacenado para mostrar dicho descriptor;

— ***transport_protocol_label*** (rótulo del protocolo de transporte): campo de 8 bits que almacena el rótulo del protocolo de transporte para especificar el carrusel de objetos que transmite los módulos referidos en el descriptor *pre-fetch* arriba. Para el rótulo del protocolo de transporte, ver 12.17.6;

— ***label_length*** (longitud de rótulo): campo de 8 bits que indica la longitud de bytes del rótulo de descripción a ser incluido en el consecuente enlace;

— ***label_char*** (descripción de rótulo): campo de 8 bits. Un rótulo del módulo está almacenado. Esto corresponde a la descripción de rótulo transmitida por el descriptor de rótulos que está almacenado en el *userInfo* del *moduleInfo* del DII en el carrusel de objetos;

— ***prefetch_priority*** (prioridad *pre-fetch):* campo de 8 bits que almacena los valores de 1 a 100. Estos valores muestran la prioridad del *pre-fetch.* Mayor valor muestra mayor prioridad.

12.17.12 Descriptor de localización DII

Para cada aplicación solamente un descriptor de localización DII puede ser dado, en caso de ser necesario. Este descriptor puede ser almacenado tanto en el enlace de descriptor común como en el enlace del descriptor de la información de la aplicación. Esto se define para ser usado por el sistema de transmisión del carrusel de objetos *(protocol_id*=0x0001). Cada descriptor es asociado a un descriptor de protocolo de transporte a través de un rótulo de protocolo de transporte.

El grupo de módulos como una parte del carrusel de objetos DSM-CC es notificado por el *DownloadInfoIndication* (DII). Todos los mensajes DII que muestran la existencia del carrusel de objetos no pueden ser listados en localización uno por uno.

Es necesario hacer todos los mensajes DII disponibles en orden para encontrar los módulos correspondientes a los rótulos para hacer el pre-fetch (ver 12.17.11). El descriptor de localización DII lista la localización de estos DII.

En el caso de que el descriptor de localización DII no estuviera incluido, solamente los módulos de indicación DII que incluyen el *ServiceGateway* serían encontrados.

Los enlaces de la identificación DII en el descriptor están localizados en orden de importancia. Es decir, DII que tiene alta prioridad de pre-fetch será localizada al comienzo del enlace. El receptor montado con mecanismos de módulos-base de pre-fetch verifica el DII en el orden listado en el descriptor de localización de DII (ver Tabla 65).

Tabla 65 — Estructura del descriptor de localización de DII

Estructura de datos	Número de bits	*String* de bits
DII_location_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
transport_protocol_label	8	*uimsbf*
for (i=0; i<N ; i++) {		
reserved_future_use	1	*bslbf*
DII_identification	15	*uimsbf*
association_tag	16	*uimsbf*
}		
}		

La descripción del *DII_location_descriptor ()* debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag):* campo de 8 bits que almacena 0x0D que indica que éste es el *DII_location_descriptor*

— ***transport_protocol_label*** (rótulo de protocolo de transporte): campo de 8 bits que almacena el rótulo del protocolo de transporte que especifica el carrusel de objetos que es transmisor del descriptor de *pre-fetch modulesreferredin.* Para detalles del rótulo del protocolo de transporte, ver 12.17.6;

— ***DII_identification*** (identificación DII): campo de 15 bits que almacena el valor que especifica el *DIImessage.* Este valor corresponde al bit 1-15 de la transacción ID en el *dsmMessageHeader()* del *DII message;*

— ***association_tag*** (asociación de la *tag):* campo de 16 bits que indica la relación con *DII message* que se transmite (por ejemplo, el *stream* elemental).

12.18 Descriptor de aplicación Ginga

12.18.1 Estructura del descriptor de aplicaciones Ginga

Un descriptor es almacenado en el enlace del descriptor de la información de la aplicación AIT para cada aplicación Ginga. Esto indica el parámetro de información para el inicio de la aplicación (ver Tabla 66).

Tabla 66 — Estructura del descriptor de aplicación Ginga

Estructura de dados	Bits	Mnemonico
ginga_application_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
for (i=0; i<N ; i++) {		
parameter_length	8	*uimsbf*
for (j=0; j<parameter_length ; j++) {		
parameter_byte	8	*uimsbf*
}		
}		
}		

La descripción del*ginga_application_descriptor ()* debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag*): campo de 8 bits donde 0x03 es almacenado para indicar el descriptor de aplicación Ginga-J y 0x06 es almacenado para indicar en el respectivo descriptor de aplicación Ginga NCL;

— ***parameter_length*** (extensión del parámetro): campo de 8 bits que muestra la extensión de bytes de la descripción de parámetro subsecuente;

— ***parameter_byte*** (descripción del parámetro): campo de 8 bits. La *string* que se dará a la aplicación como parámetro es almacenada.

12.18.2 Descriptor de localización de la aplicación Ginga

El descriptor es almacenado en el enlace del descriptor de la información de la aplicación AIT una vez para cada aplicación Ginga. Éste almacena la información de camino necesaria en la aplicación (ver Tabla 67).

Tabla 67 — Estructura del descriptor delocalización de la aplicación Ginga

Estructura de dados	Bits	Mnemonico
ginga_application_location_descriptor()		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
base_directory_length	8	*uimsbf*
for (i=0; i<N ; i++) I		
base_directory_byte	8	*uimsbf*
}		
entitypath_extension_length	8	*uimsbf*
for (i=0; i<N ; i++) I		
entitypath_extension_byte	8	*uimsbf*
}		
for (i=0; i<N ; i++) I		
initial_entity	8	*uimsbf*
}		
}		

La descripción de *ginga_application_location_descriptor ()* debe ser la siguiente:

— ***descriptor_tag*** (descriptor de *tag*): campo de 8 bits donde 0x04 es almacenado para indicar descriptor de localización de aplicación Ginga-J y 0x07 es almacenado para indicar en el respectivo descriptor de aplicación Ginga NCL;

— ***base_directory_length*** (extensión del directorio base): campo de 8 bits que indica la extensión de los bytes que se incluirán en los enlaces subsecuentes. El valor almacenado será 1 o mayor;

— ***base_directory_byte*** (byte del directorio base): campo de 8 bits. El nombre del directorio del camino del archivo del sistema se debe almacenar por una letra delimitadora, usando "/" (0x2F). Este nombre de directorio se usa por el directorio-base en el camino relativo. Si el directorio de camino del archivo del sistema es designado de acuerdo a un directorio-base, "/" se debe almacenar;

— ***entitypath_extension_length*** (extensión del camino de la entidad): campo de 8 bits que indica o extensión del byte del camino de clase adicional subsecuente;

— ***entitypath_extension_byte*** (camino de la entidad): campo de 8 bits. Cuando un camino de clase es designado por otro directorio que no sea directorio-base, el nombre del camino de clase es almacenamiento. El nombre del directorio del camino del archivo del sistema es almacenado por la letra delimitadora usando "/" (0x2F). Si hubiere más de un camino, ellos deben ser almacenados por enumeración con ";" (0x3B);

— ***initial_entity*** (entidad de activación inicial): campo con 8 bits. El nombre de la entidad inicial de la aplicación que consta en el sistema de archivos almacenados.

13 Especificación de la transmisión del mensaje del evento

13.1 Mensaje de evento

La especificación de la transmisión del mensaje del evento provee un medio para enviar mensajes de información inmediatamente o en horas específicas para una aplicación operada en una unidad de receptor de una estación de difusión.

La especificación de la transmisión del mensaje del evento definido en esta Norma es extendida para negociar los distintos tiempos apuntando métodos por la aplicación con base en la especificación del descriptor de *stream* y su especificación de la transmisión de sección DSM-CC especificada en la ISO/IEC 13818-6.

13.2 Descriptores de *stream*

13.2.1 Descriptor de *stream* DSM-CC

Esta sección está de acuerdo con los requisitos de la ISO/IEC 13818-6.

Los descriptores de s*tream* se pueden usar para proveer información DSM-CC que está correlacionada con un *transport stream* o *program stream* MPEG-2. Estos descriptores están en formato de programas y elementos de programas descriptores conforme definido en ISO/IEC 13818-1. Los descriptores de *stream* DSM-CC deben ser solamente cargados en una *DSMCC_section* (ver a ISO/IEC 13818-6:1998, Sección 9). Esto crea un espacio identificador único (de aquel definido por ISO/IEC 13818-1) para valores de descriptor de *tag*. El formato general de todos los descriptores de *stream* definidos en esta parte de la ABNT NBR 15606 está mostrado en la Tabla 68.

Tabla 68 — Descriptor de *stream* DSM-CC

Sintaxis	Número de bits	Mnemónico
dsmccStreamDescriptor () {		
descriptorTag	8	*uimsbf*
descriptorLength	8	*uimsbf*
descriptor()		
}		

La descripción del *dsmccStreamDescriptor ()* debe ser la siguiente:

— ***descriptorTag:*** campo de 8 bits que identifica cada descriptor. La extensión de posibles valores para el descriptorTag se muestra en la Tabla 69;

— **descriptorLength:** éste es un campo de 8 bits que especifica el número de bytes del descriptor inmediatamente después del campo *descriptorLength.*

Tabla 69 — Valor del campo *DescriptorTag*

Descriptor de *tag*	Descripción
0x00	ISO/IEC 13818-6 reservado
0x01	Descriptor de referencia NPT
0x02	Descriptor de *endpoint* NPT
0x03	Descriptor de modo *stream*
0x04	Descriptor de evento de *stream*
0x05 – 0xFF	ISO/IEC 13818-6 reservado

13.2.2 Descriptor de referencia NPT

Para activar la determinación de tiempo del NPT de un evento, se define el descriptor de referencia NPT. El formato del descriptor de referencia se muestra en la Tabla 70.

Tabla 70 — Descriptor de referencia NPT

Sintaxis	Número de bits	Mnemónico
NPTReferenceDescriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
postDescontinuityIndicator	1	*bslbf*
dsm_contentId	7	*uimsbf*
Reserved	7	*bslbf*
STC_Reference	33	*uimsbf*
reserved	31	*bslbf*
NPT_Reference	33	*tcimbsf*
scaleNumerator	16	*tcimbsf*
scaleDenominator	16	*tcimbsf*
}		

La descripción del *NPTReferenceDescriptor ()* debe ser la siguiente:

— ***descriptorTag***: campo de 8 bits que identifica el tipo del descriptor de stream. El valor del campo descriptorTag para el descriptor de referencia NPT se muestra en la Tabla 69;

— ***descriptorLength:*** campo de 8 bits que especifica el número de bytes del descriptor inmediatamente después del campo *descriptorLength;*

— ***postDiscontinuityIndicator:*** campo de 1 bit. Un valor de 0 indica que el descriptor de referencia NPT es válido en la recepción. Un valor de 1 indica que el descriptor de referencia NPT torna válido en el próximo tiempo la base de discontinuidad del sistema, conforme definido en la ISO/IEC 13818-1;

— ***dsm_contentId:*** campo de 7 bits que identifica que un conjunto anidado de contenido está siendo presentado. El campo satisfecho se puede usar para indicar la transición para una base de tiempo diferente NPT dentro de una base de tiempo existente NPT. Por ejemplo, este campo puede ser cambiado cuando un comercial es presentado dentro de un programa de televisión y después nuevamente cambiado cuando el programa de televisión es reiniciado;

— ***STC_Reference:*** entero sin signo de 33 bits, que indica el valor de STC para el cual NPT iguala el campo *NPT_Reference.* El valor del campo *STC_Reference* se especifica en unidades de períodos del sistema de frecuencia de reloj, conforme definido en la ISO/I EC 13818-1, dividido entre 300, rindiendo unidades de 90 kHz. *STC_Reference* es derivado de un sistema de frecuencia de reloj, según mostrado en la ecuación siguiente:

STC_Referencek = (STCNPT(k) / 300) % 233

donde

STCNPT(k) es el valor del sistema de tiempo del reloj *"System Time Clock"* cuando el NPT iguala el valor de *NPT_Reference;*

— ***NPT_Reference:*** entero con signo de 33 bits indicando el valor de NPT en el valor dado de STC en el campo *STC_Reference field;*

— ***scaleNumerator:*** entero con signo de 16 bits usado con el *scaleNumerator,* un entero sin signo de 16 bits, para definir la extensión del cambio de NPT con relación al STC. Un valor de 1 para *scaleNumerator* con un valor de 1 para *scaleDenominator* indica que el NPT está cambiando en una tasa equivalente al STC, rindiendo la tasa estándar de la presentación. Un valor de 0 para *scaleNumerator* con un valor no cero para *scaleDenominator* indica que NPT no está cambiando con relación al STC, rindiendo un valor constante del NPT. Un valor de 0 para *scaleNumerator* con un valor de 0 para *scaleDenominator* indica que los campos *scaleNumerator* y *scaleDenominator* no están provistos en el descriptor de referencia NPT. Un valor no-cero para *scaleNumerator* con un valor de 0 para *scaleDenominator* no será usado (ver Tabla 71).

Tabla 71 — *ScaleNumerator*

scaleNumerator	***scaleDenominator***	**Semántica**
0	0	Indica que el *scaleNumerator* y el *scaleDenominator* no se usan
0	Otro además de 0	NPT continúa el valor constante irrelevante para STC
1	1	NPT y STC avanzan en la misma tasa
Otro además de 0	0	Tal combinación no se debe usar

13.3 Descriptor de modo de *stream*

El descriptor de modo de *stream* contiene información sobre el modo de la máquina de estado del *stream,* permitiendo que los clientes sincronicen mejor sus acciones con los cambios de estado del *stream.* El formato del descriptor de modo del *stream* se muestra en la Tabla 72.

Tabla 72 — Descriptor de modo de *stream*

Sintaxis	Número de bits	Mnemónico
StreamModeDescriptor () {		
descriptorTag	8	*uimsbf*
descriptorLength	8	*uimsbf*
streamMode	8	*uimsbf*
reserved	8	*bslbf*
}		

La descripción del *StreamModeDescriptor ()* debe ser la siguiente:

— ***descriptorTag***: campo de 8 bits que identifica el tipo del descriptor del *stream*. El valor del campo *descriptorTag* para el descriptor de modo del *stream* se muestra en el Tabla 69;

— ***descriptorLength***: campo de 8 bits que especifica el número de bytes del descriptor inmediatamente después del campo *descriptorLength*;

— ***streamMode***: campo de 8 bits cuyo valor indica el estado actual de la máquina de estado del *stream*. Los valores para *streamMode* se muestran en la Tabla 73. Los estados de la máquina de estado de *stream* son definidos en la ISO/IEC 13818-6:1998, Sección 5.

Tabla 73 — Valores del campo *StreamMode*

Descriptor de modo	Descripción
0	Abierto
1	Pausa
2	Transporte
3	Pausa en el transporte
4	Transporte de busca
5	Pausa de transporte de busca
6	Transporte de pausa de la busca
7	Fin de *stream*
8	Transporte de pre-busca
9	Pausa de transporte de pre-busca
10 - 255	ISO/IEC 13818-6 reservado

13.4 Descriptores de evento de *stream*

El descriptor de evento de *stream* contiene información que permite la transmisión de eventos específicos, conforme definido en la ISO/IEC 13818-6:1998, Sección 5, de modo que puedan ser sincronizados con el *stream.* La definición de evento en este contexto no es la misma si el evento está relacionado con NPT. El formato del descriptor de evento de stream se muestra en la Tabla 74.

Tabla 74 — Descriptor de evento de *stream*

Sintaxis	Número de bits	Mnemónico
StreamEventDescriptor () {		
descriptorTag	8	*uimsbf*
descriptorLength	8	*uimsbf*
eventId	16	*uimsbf*
reserved	31	*bslbf*
eventNPT	33	*uimsbf*
for(i=0;i<N;i++){		
privateDataByte	8	*uimsbf*
}		
}		

La descripción del *StreamEventDescriptor ()* debe ser la siguiente:

— ***descriptorTag***: campo de 8 bits que identifica el tipo del descriptor de stream. El valor del campo descriptorTag para el descriptor de evento de stream se muestra en la Tabla 69;

— ***descriptorLength:*** campo con 8 bits que especifica el número de bytes del descriptor inmediatamente después del campo *descriptorLength;*

— ***eventid:*** campo de 8 bits cuyo valor es un tipo de evento específico de la aplicación;

— ***eventNPT:*** entero sin signo, cuyo valor es el valor de NPT cuando el evento ocurrió, o el valor de NPT cuando el evento ocurra;

— ***privateDataByte:*** campos que permiten la inclusión de datos específicos de aplicación en el descriptor de eventos de *stream.*

13.5 Descriptor de evento general

El descriptor de evento general *(general_event_descriptor)* es un descriptor para comunicar información aplicables a mensajes de eventos.

La estructura de datos del descriptor de evento general se muestra en la Tabla 75.

Tabla 75 — Descriptor de evento general

Sintaxis	Número de bits	Mnemónico
General_event_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
event_msg_group_id	12	*uimsbf*
reserved_future_use	4	*bslbf*
time_mode	8	*uimsbf*
if(time_mode == 0){		
reserved_future_use	40	*bslbf*
}		
else if(time_mode == 0x01 || time_mode == 0x05){		
event_msg_MJD_JST_time	40	*bslbf*
}		
else if(time_mode == 0x02){		
reserved_future_use	7	*bslbf*
event_msg_NPT	33	*uimsbf*
}		
else if(time_mode == 0x03){		
reserved_future_use	7	*bslbf*
event_msg_relative_time	36	*bslbf*
}		
event_msg_type	8	*uimsbf*
event_msg_id	16	*uimsbf*
for(i=0;i<N;i++){		
private_data_byte	8	*uimsbf*
}		
}		

La descripción del *General_event_descriptor ()* debe ser la siguiente:

— ***event_msg_group_id*** (identificador de grupo de mensaje de evento): campo de 12 bits que identifica el grupo de mensajes a ser recibidas por el aplicador. Detalles de las operaciones se especifican en cada identificador de codificación de datos. Cuando está operando un evento de mensaje con la identificación de más de un grupo de mensajes al mismo tiempo, solamente los descriptores de evento general con el identificador del mismo grupo de mensajes deben ser incluidos en una sección DSM-CC;

— ***time_mode*** (modo de tiempo): campo de 8 bits que indica el método para designar el tiempo cuando un evento de mensaje es generado (ver Tabla 76);

Tabla 76 — Modo de tiempo

Time_mode	Método de designación de tiempo	Semántica
0x00	Ninguno	El mensaje de evento es generado inmediatamente después de la recepción
0x01	mjd_utc_time	El mensaje de evento es generado en el tiempo absoluto indicado por el tiempo MJD UTC. El mensaje de evento también es generado cuando el contenido grabado del *stream* es reproducido refiriéndose al tiempo de reproducción
0x02	NPT	El mensaje de evento es generado en el tiempo específico con los datos de tiempo NPT
0x03	*eventRelative Time*	El mensaje de evento es generado cuando el período se especifica en este campo (en milésimos de segundo) después del tiempo de inicio del programa
0x04	---	Reservado para el futuro
0x05	*MJD_UTC_time*	El mensaje de evento es generado en el tiempo absoluto indicado por el tiempo MJD UTC. Cuando el contenido grabado del *stream* es reproducido, el mensaje de evento también es generado refiriéndose al tiempo en el aire
0x06-0xFF	---	Reservado para el futuro

— ***event_msg_MJD_UTC_time:*** campo de 40 bits se codifica en el caso del modo de tiempo = 0 x 01 ó 0x05 e indica el tiempo cuando el evento de mensaje es generado en el UTC y MJD (se refiere a ARIB STD-B10:2003, Parte 2, Anexo C). Este campo contiene una copia de los 16 bits más bajos de MJD y es seguido por seis representaciones de 4 bits decimal codificado en binario (BCD);

— ***event_msg_NPT:*** campo de 33 bits que se codifica en el caso del modo de tiempo = 0 x 02 e indica el tiempo usando el *Normal Play Time* de DSM-CC (ver 7.1), cuando el evento de mensaje es generado;

— ***event_msg_relativeTime:*** campo de 36 bits que se codifica en el caso del modo de tiempo = 0 x 03 e indica que el evento de mensaje es generado cuando el período especificado en este campo pasa después del horario de inicio del programa. El valor de este campo se describe en el orden de hora (2 dígitos), minuto (2 dígitos), segundo (2 dígitos) milésimos de segundo (3 dígitos), para formar nueve representaciones de 4 bits decimales codificados en binario (BCD);

— ***event_msg_type*** **(tipo de mensaje de evento):** un identificador que indica el tipo de mensaje de evento. El uso y la semántica son especificados en cada especificación de codificación de datos.

— ***event_msg_id*** **(identificador de mensaje de evento):** campo de 16 bits que contiene el identificador para identificar cada mensaje de evento. El uso y la semántica son especificados en cada especificación de codificación de datos.

— ***private_data_byte*** **(datos privados):** campo de 8 bits que almacena información relacionada al evento de mensaje requerido por la especificación de codificación de datos especificado en el *event_msg_type.*

13.6 Sintaxis de sección de DSM-CC transmitiendo el descriptor de stream

El descriptor de *stream* se transmite en la sección DSM-CC mostrada en la Tabla 77.

Tabla 77 — Sección DSM-CC (transmisión de descriptor de *stream)*

Sintaxis	Número de bits	Mnemónico
DSMCC_section () {		
table_id	8	*uimsbf*
section_syntax_indicator	1	*bslbf*
private_indicator	1	*bslbf*
Reserved	2	*bslbf*
dsm_cc_section_length	12	*uimsbf*
data_event_id	4	*uimsbf*
event_msg_group_id	12	*uimsbf*
Reserved	2	*bslbf*
version_number	5	*uimsbf*
current_next_indicator	1	*bslbf*
section_number	8	*uimsbf*
last_section_number	8	*uimsbf*
if(table_id == 0x3D){		
for(i=0;i<N;i++){		
stream_descriptor()		
}		
}		
if(section_syntax_indicator == '0'){		
Checksum	32	*uimsbf*
}		
Else{		
CRC_32	32	*rpchof*
}		
}		

La descripción de la *DSMCC_section ()* debe ser la siguiente:

— ***table_id*** (tabla de identificación): campo de 8 bits que es regulado para 0x3D, para indicar que el descriptor de *stream* está almacenado en el *payload* de la sección DSM-CC;

— ***section_syntax_indicator:*** campo de 1 bit que indica que CRC32 existe al final de la sección cuando es 1. Cuando es 0, esto indica que existe verificación de suma. Para transmitir un evento de mensaje, este campo debe ser regulado en 1;

— ***private_indicator:*** campo de 1 bit que almacena el valor complementario de la sección de valor indicador de sintaxis;

— ***dsmcc_section_length:*** campo de 12 bits que indica la longitud de byte del área de posición inmediatamente siguiente a este campo al fin de la sección. Este valor de campo no debe exceder 4 093;

— ***data_event_id:*** campo de 4 bits que es el identificador para identificar el evento de datos entre los datos precedentes y siguientes que usan el evento de mensaje para permitir el contenido local pretendido, a pesar

de ser precedidos y/o seguidos otros contenidos locales, para recibir el evento de mensaje. Contenidos locales adyacentes, cuando están transmitiendo, son asignados con diferentes identificadores;

— ***event_msg_group_id*** (identificador de grupos de mensaje): campo de 12 bits que contiene el campo para identificar el evento de mensajes a ser recibidos por la aplicación. La semántica detallada se detalla en cada especificación de codificación de datos;

— ***version_number:*** campo de 5 bits que es el número de la versión de la subtabla. El número de versión se incrementa en 1 cuando cualquier pieza de información en la subtabla sea cambiada. Los valores disponibles son de 0 a 31. El valor 0 se utiliza para actualizar el 31;

— ***current_next_indicator:*** indicador de 1 bit que indica que la subtabla es la actual subtabla cuando es '1'. Cuando es '0', la subtabla enviada no es aún aplicada y usada como la próxima subtabla;

— ***section_number:*** campo de 8 bits que indica el número de la sección;

— ***last_section_number:*** campo de 8 bits que indica el número de la última sección (que es la sección con el número máximo de secciones de la subtabla a la que las secciones participantes pertenecen).

14 Sistema de archivo de difusión y transporte de disparador

El sistema de archivo de difusión y transporte de disparador debe estar de acuerdo con la GEM 1.0:2005, Anexo B.

Anexo A
(normativo)

Video y audio PES

A.1 Formato de transmisión de datos a través de la PES de video MPEG-2 codificado

En el caso del uso de PES de video codificado con el video MPEG-2 (ver ISO/IEC 13818-2) para transmitir datos, se deberá utilizar el campo de datos del usuario *(user_data_area),* de conformidad con el encabezamiento de la pantalla del *stream* de video. La sintaxis del campo de datos del usuario se muestra en la Tabla A.1. El uso más detallado de este área depende del modo de operación de las emisoras.

Tabla A.1 — Sintaxis del campo de datos del usuario del *stream* de video

Sintaxis	Número de bits	Mnemónico
User Data () {		
user_data_start_code	32	*bslbf*
while (nextbits() != 0x000001){		
user_data	8	*uimsbf*
}		
next_start_code()		
}		
NOTA Código de inicio de datos de usuarios: 0x000001b2.		

A.2 Formato de transmisión de datos del audio PES codificado con MPEG-2 BC audio

En la utilización de paquetes PES de audio MPEG-2 BC audio (ver ISO/IEC 13818-3) para transmisión de datos, el área de datos jerárquicos *(ancillary data area),* que puede contener otros datos que no sean audio MPEG, se debe utilizar. La sintaxis del área de datos jerárquicos se muestra en la Tabla A.2 Un uso más detallado de ese área depende de los operadores de servicio.

Tabla A.2 — Área de datos subordinada al *stream* de audio

Sintaxis	Número de bits	Mnemónico
MPEG1_ancillary_data() { *if(ext_bit_stream_present == 1){* *for(b=0; b<8*n_ad_bytes;b++)* *ancillary_bit* *}* *}*	1	*bslbf*

A.3 Formato de transmisión de datos del audio PES codificado con MPEG-2 AAC audio

En la utilización de paquetes PES de audio MPEG-2 AAC audio (ver ISO/IEC 1381 8-3) para transmisión de datos, el área data_*stream_element* se debe usar para permitir que otros datos, además de datos de audio MPEG, puedan estar contenidos en cada bloque de datos puro *(raw_data_block).* La sintaxis del área se muestra en la Tabla A.3. Un uso más detallado de ese área depende de los operadores de servicio.

Tabla A.3 — Área base de datos de *stream* del audio *stream* (MPEG-2 AAC audio)

Sintaxis	Número de bits	Mnemónico
data_stream_element () {		
element_instance_tag	4	*uimsbf*
data_byte_align_flag	1	*uimsbf*
cnt = count	8	*uimsbf*
if(cnt == 255){	8	*uimsbf*
cnt += esc_count		
}		
if(data_byte_align_flag)		
byte_alignment()		
for(i=0;i<cnt;i++)		
data_stream_byte[element_instance_tag][i]	8	*bslbf*
}		

Anexo B
(normativo)

Información PSI/SI para transmisión de carruseles de datos y mensajes de eventos

B.1 Especificación de la codificación de datos con base en el carrusel de datos y esquema de evento de mensaje

En adición a la especificación de codificación de datos aplicada, la transmisión de datos con base en el carrusel de datos y el esquema de evento de mensaje serán definidos; una sintaxis adicional depende del formato de transmisión de datos a ser insertada en el *data_component_descriptor* en PMT y *data_content_descriptor* en EIT especificados en la ARIB STD-B23.

Esta Norma se basa en las siguientes suposiciones sobre operación de transmisión para servicios de difusión de datos, de la siguiente manera:

— el DII y DDB pertenecientes a un carrusel se transmiten en un ES;

— un servicio de difusión de datos puede consistir en dos o más carruseles. Eventos de mensajes se pueden transmitir.

B.2 Contenido de enlace de *additional_data_component_info* y *data_component_descriptor*

Para insertar la información para control de recepción del carrusel de datos y el posible evento de mensajes en el enlace que contiene *additional_data_component_info* al final de *data_component_descriptor,* la siguiente estructura de datos debe ser colocada en el enlace, según determinado por la especificación de codificación de datos (ver Tabla B.1).

Tabla B.1 – *Additional ginga carousel info*

Sintaxis	Número de bits	Mnemónico
additional_ginga_carousel_info() {		
data_event_id	4	*uimsbf*
event_section_flag	1	*bslbf*
reserved	3	*bslbf*
}		

La descripción de la *additional_ginga_carousel_info()* debe ser la siguiente:

— ***data_event_id:*** identificador de 4 bits que reconoce los eventos de datos precedentes y siguientes, usando el carrusel de datos y mensajes del posible evento para evitar defecto al recibir el contenido local apropiado transmitido en el carrusel de datos y mensajes de posibles eventos. En el caso de todos los bits de este campo regulados en 1, eso significa que los DIIs tienen un identificador de *data_event_id* entregado en este servicio y el evento de mensajes es válido;

— ***event_section_flag:*** campo de 1 bit indica si un evento de mensajes fue o no enviado con este componente, de la siguiente manera:

0: mensajes del evento no fueron transmitidos;

1: mensajes del evento fueron transmitidos;

— ***reserved:*** reservado.

B.3 *Byte selector de data_contents_descriptor*

B.3.1 *Data structure*

Para insertar información para control de recepción de carrusel de datos en el byte selector de descriptor de contenidos de datos como EIT, una estructura de datos debe ser colocada en el campo *selector_byte,* según determinado por la especificación de codificación de datos participante.

B.3.2 Estructura de datos para control de recepción de carrusel de datos para servicios de datos no almacenados

Para servicios de datos no almacenados, insertar las informaciones contenidas en la Tabla B.2.

Tabla B.2 — Servicios de datos no almacenados

Sintaxis	Número de bits	Mnemónico
ginga_carousel_info () {		
num_of_carousels	8	*uimsbf*
for(i=0;i<num_of_carrousels;i++){		
component_tag	8	*uimsbf*
event_section_flag	1	*bslbf*
reserved_future_use	3	*bslbf*
component_size_flag	1	*bslbf*
default_transaction_id_flag	1	*bslbf*
default_timeout_DII_flag	1	*bslbf*
default_leak_rate_flag	1	*bslbf*
if(component_size_flag == '1'){		
component_size	32	*uimsbf*
}		
if(default_transaction_id_flag == '1'){		
timeout_value_DII	32	*uimsbf*
}		
if(default_leak_rate_flag == '1'){		
leak_rate	22	*uimsbf*
reserved	2	*bslbf*
}		
}		
}		

La descripción de la *ginga_carousel_info()* debe ser la siguiente:

— ***num_of carousels:*** campo de 8 bits que indica el número de carruseles, incluidos en el enlace siguiente;

— ***component_tag:*** campo de 8 bits que designa el componente del *stream* transmitiendo los carruseles con el componente de tag dado por el descriptor identificador de *stream* en el PMT;

— ***event_section_flag:*** campo que indica si el evento de mensajes fue o no enviado usando este componente;

— ***component_size_flag:*** campo de 1 bit que indica si la estructura de datos contiene o no el componente de tamaño. Cuando el valor del campo *component_size* no está disponible, debe ser regulado a “0”:

0: no codificado;

1: codificado;

— ***default_transaction_id_flag:*** campo de 1 bit que indica si el identificador de transacción está codificado o no en esta sintaxis. Para designación de manera que se obtenga todo de la identificación de transacción opcional, la identificación de transacción no debe ser codificada (0: no codificado; 1: codificado);

— ***default_timeout DII_flag:*** campo de 1 bit que indica si el valor de explosión de tiempo de DII está codificado en esta sintaxis. Cuando el valor estándar especificado en esta operación como valor de explosión de DII se utiliza, no está codificado (0: no codificado; 1: codificado);

— ***default_leak_rate_flag:*** campo de 1 bit que indica si la tasa de dispersión está codificada o no en esta sintaxis. Cuando el valor estándar especificado en esta operación como valor de dispersión de DII se utiliza, no está codificado (0: no codificado; 1: codificado);

— ***component_size:*** campo de 32 bits que indica el tamaño total (en bytes) de los datos transmitidos en los carruseles de este componente;

— ***transaction_id:*** identificación del valor de la transacción DII transmitida en este componente. En el caso de que la identificación de transacción no esté presente, debe obtenerse una DII con la identificación de transacción;

— ***time_out_value_DII:*** campo de 32 bits que indica el valor de intervalo recomendado (en milésimas de segundo) para recibir la sección completa de DII de este carrusel. Cuando el valor es 0 x FFFFFFFF, eso significa que no existe valor de intervalo recomendado;

— ***leak_rate:*** campo de 22 bits que indica la tasa de dispersión del transporte de *buffer* de la unidad del receptor en una unidad de 50 bytes/s.

B.3.3 Estructura de datos para el control de la recepción del carrusel de datos para el servicio de datos almacenados

Para el servicio de datos almacenados, insertar las informaciones contenidas en la Tabla B.3.

Tabla B.3 — Servicio de datos almacenados

Sintaxis	Número de bits	Mnemónico
ginga_stored_carousel_info () {		
num_of_carousels	8	*uimsbf*
for(i=0;i<num_of_carrousels;i++){		
component_tag	8	*uimsbf*
num_dataEvent_flag	1	*bslbf*
num_modules_flag	1	*bslbf*
num_resources_flag	1	*bslbf*
compressed_component_size_flag	1	*bslbf*
component_size_flag	1	*bslbf*
default_transaction_id_flag	1	*bslbf*
default_timeout_DII_flag	1	*bslbf*
default_leak_rate_flag	1	*bslbf*
if(num_dataEvent_flag == '1'){		
num_dataEvent	16	*uimsbf*
}		
if(num_modules_flag == '1'){		
num_modules	32	*uimsbf*
}		
if(num_resources_flag == '1'){		
num_resources	32	*uimsbf*
}		
if(compressed_component_size_flag == '1'){		
compressed_component_size	32	*uimsbf*
}		
}		

La descripción de la *ginga_stored_carousel_info ()* debe ser la siguiente:

— ***num_of carousels:*** campo de 8 bits que indica el número de carruseles incluyendo el del enlace siguiente;

— ***component_tag:*** campo de 8 bits que designa el componente de *stream* transmitiendo los carruseles con el componente de la *tag* dado por el descriptor identificador de *stream* en PMT;

— ***num_dataEvent_flag:*** campo de 1 bit indica si la estructura de datos contiene o no un número de eventos de datos. Cuando el valor del campo *num_dataEvent* no está disponible, debe ser regulado a 0 (0: no codificado; 1: codificado);

— ***num_modules_flag:*** campo de 1 bit que indica si la estructura de datos contiene o no un número total de módulos. Cuando el valor del campo *num_modules* no está disponible, debe ser regulado a 0 (0: no codificado; 1: codificado);

— ***num_resources_flag:*** campo de 1 bit que indica si la estructura de datos contiene o no un número total de recursos. Cuando el valor del campo *num_resources* no está disponible, debe ser regulado a 0 (0: no codificado; 1: codificado);

— ***compressed_component_size_flag:*** campo de 1 bit que indica si la estructura de datos contiene o no el tamaño comprimido del componente. Cuando el valor del campo *compressed_component_size* no está disponible, debe ser regulado a 0 (0: no codificado; 1: codificado);

— ***component_size_flag:*** campo de 1 bit que indica si la estructura de datos contiene o no el tamaño del componente. Cuando el valor del campo *component_size* no está disponible, debe ser regulado a 0 (0: no codificado; 1: codificado);

— ***default_transaction_id_flag:*** campo de 1 bit que indica si el identificador de la transacción está codificada o no en la sintaxis. Para designar la ganancia de DII de identificación de transacción opcional, la identificación de la transacción no debe estar codificada (0: no codificado; 1: codificado);

— ***default_timeout DII_flag:*** campo de 1 bit que indica si el valor de intervalo está o no codificado en la sintaxis. Cuando el valor básico especificado en la operación conforme el valor de intervalo DII se utiliza, no se codifica (0: no codificado; 1: codificado);

— ***default_leak_rate_flag:*** campo de 1 bit que indica si la tasa de dispersión está o no codificada en la sintaxis. Cuando el valor básico especificado en la operación como el valor de la tasa de dispersión se utiliza, no se codifica (0: no codificado; 1: codificado);

— ***num_dataEvent:*** campo de 32 bits que indica el número total de eventos de datos en el componente participante;

— ***num_modules:*** campo de 32 bits que indica el número total de módulos en los eventos de datos en el componente participante;

— ***num_resources:*** campo de 32 bits que indica el número total de recursos en los eventos de datos en el componente participante;

— ***compressed_component_size:*** campo de 32 bits que indica el tamaño total (en bytes) de los datos en los eventos de datos en los carruseles de datos de este componente. El tamaño del módulo comprimido es calculado con base en el estado comprimido, no en el estado expandido;

— ***component_size:*** campo de 32 bits que indica el tamaño total (en bytes) de los datos en los eventos de datos en los carruseles de datos de este componente. El tamaño del módulo comprimido es calculado con base en el estado extraído, no en el estado comprimido;

— ***transaction_id:*** transacción de identificación del valor DII transmitido en este componente. En el caso de la identificación de la transacción no estar definida, debe obtenerse la DII con la identificación de la transacción;

— ***time_out_value_DII:*** campo de 32 bits que indica el valor de intervalo recomendado (en milésimas de segundos) para recibir la sección completa de DII en este carrusel. Cuando el valor sea 0xFFFFFFFF, ello significa que no existe valor de intervalo recomendado;

— ***leak_rate:*** campo de 22 bits que indica la tasa de dispersión del transporte de *buffer* de la unidad del receptor en una unidad de 50 byte/s.

Anexo C
(informativo)

Relación entre el descriptor PMT/EIT y AIT

La Figura C.1 muestra la relación de AIT para el descriptor componente de datos PMT y el descriptor de contenidos de datos EIT en Ginga.

Los ítems de extensión para Ginga son los siguientes:

— nueva atribución para la identificación de valor a ser usado en Ginga (por ejemplo, componentes de datos ID, *transport_id, application_id);*

— para el componente ES que transmite la aplicación Ginga o para el componente ES que transmite AIT, las áreas del selector son especificadas para los descriptores de componentes de datos que almacenan información adicional, a no ser que se transmita a través de AIT;

— las áreas de selector son especificadas para los descriptores de contenidos de datos en orden para almacenar los detalles de la aplicación Ginga en la base del evento del programa;

— en un descriptor, es regulada *application_identifier_flag* en vez de *entry_point_flag* en orden para almacenar información especificando una aplicación que es un punto de entrada en PMT/EIT.

Las informaciones adicionales son para las áreas del selector de los descriptores de componentes de datos, así como descriptores de contenidos de datos relevantes para transmisión Ginga y AIT. El resto está supuesto para estar de conformidad con la transmisión AIT bajo MHP1.0.

Para el carrusel de datos que no transmite aplicación Ginga, es posible referir a los contenidos especificando el asignador de la aplicación Ginga.

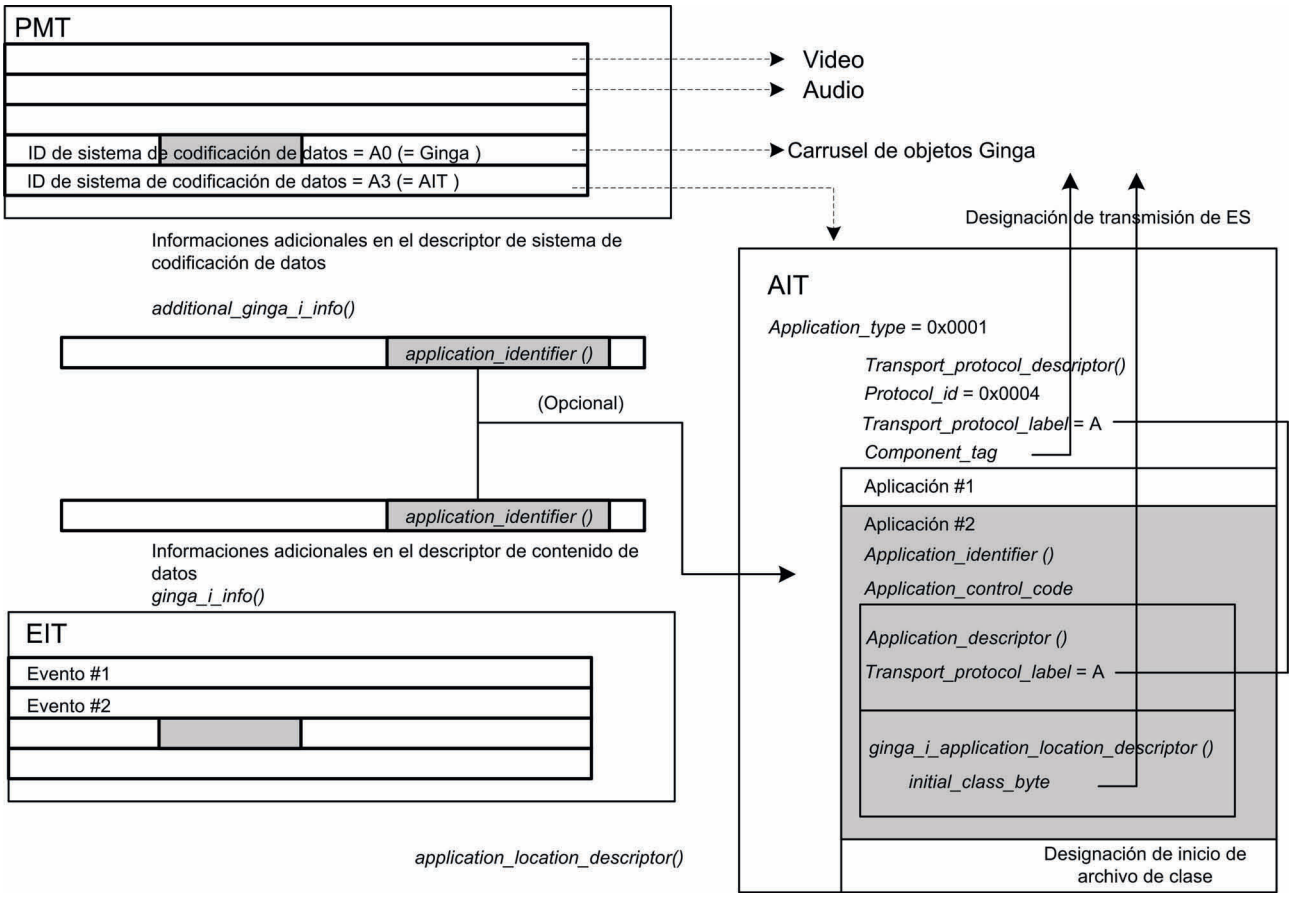


Figura C.1 — Relación de a transmisión AIT y descriptor de componente de datos

Anexo D
(informativo)

Informaciones adicionales sobre trasmisiones utilizando independientes PES

Un *data_identifier* debe estar presente en el inicio de *PES_packet_data_byte* para identificar el tipo de datos. (ver EN 301 192, ATSC DVS 161 y DAVIC 1.4:1998, parte 9).

NOTA En Japón, el Ministerio de Telecomunicaciones (*Notification Hei* 10/260) estipula el uso de *data_component_id* en la PMT.

Para asegurar la conformidad DVB, ATSC y DAVIC, el campo *data_identifier* contiene una copia del valor asignado para el área del usuario definido en esos estándares.

Las razones para emplear las especificaciones independientes PES como un estándar para el método de transmisión PES son las siguientes:

— tamaño (menor de restricciones) permite una libertad mayor;

— se permite que datos de video y audio sean producidos por separado antes de ser multiplexados por menos esfuerzo;

— es permitido compartir datos en múltiples piezas de datos de video y audio para un acceso más fácil.

La transmisión de carrusel de datos se basa en el *download* U-N (carrusel de datos DSM-CC) estipulado en la ISO/IEC 13818-6, en el cual fue agregado lo siguiente:

— con relación al área de datos del módulo, asumiendo su uso para la transmisión de archivos etc., fueron agregados los descriptores de *Activation time,* de *Expire* y de *Compression Type.*

excepto por los servicios de *download* asumidos cuando el original ISO/IEC 13818-6 fue desarrollado, estas estipulaciones permiten el uso de las transmisiones del carrusel de datos que son eficientes y que tienen la recepción mínima procesando cargas en una amplia variedad de aplicaciones, como servicios de multimedia.

Bibliografia

[1] ITU-T X.208: 1988, Specification of abstraction construction describing format (ASN. 1)

[2] ITU-T X.209: 1988, Specification of basic encryption rule of abstraction construction describing format (ASN.1)

[3] ITU-T X.234: 1994, Encryption key management and authenticating system for audio visual service

[4] ITU-T X.509: 1997, Directory – Frame of authentication

[5] JIS X 5055:1996, Security technology – Data completeness function using encryption inspection function by block encryption algorithm

[6] JIS X 5056-3:1996, Security technology – Entity authentication function – Part 3: Authentication function using open key algorithm

[7] JIS X 5057-1:1996, Security technology – Hash function – Part 1: Introduction

[8] JIS X 5057-2:1996, Security technology – Hash function – Part 2: Hush function using n bitblock encryption algorithm

[9] JIS X 5060:1994, Data encryption technology – Registration procedure of encryption algorithm

[10] http://www.nist.gov/aes (1999-3) “Advanced Encryption Standard”

[11] FIPS PUB 46-2:1993, http://www.itl.nist.gov/div897/pubs/fip46-2.htm – Data encryption standard (DES)

[12] RC5, RFC2040:1996, The RC5 Encryption algorithm

[13] FIPS PU B 140-1:1994, http://www-09.nist.gov/div897/pubs/fip140-1 .htm, Security requirements for cryptographic modules

[14] FIPS PU B 180-1:1995, http://www.itl.nist.gov/div897/pubs/fip180-1 .htm, Secure hash standard

[15] MD5, RFC1321:1992, The MD5 Message-Digest Algorithm

[16] MD2, RFC1319:1992, The MD2 Message-Digest Algorithm

[17] RFC 2246:1999, The TLS Protocol Version 1.0

[18] RFC 1590:1994, J.Postel, Media Type Registration Procedure, RFC 1590, ISI

[19] The Notification No. 260 of Ministry of Posts and Telecommunications in 1998 – JAPAN

[20] RFC 1954:1996, Transmission of Flow Labelled IPv4 on ATM Data Links Ipsilon Version 1.0

[21] RFC 1334: 1992, PPP Authentication Protocols