NORMA
BRASILEIRA

ABNT NBR 15606-3

Terceira edição
28.03.2012

Válida a partir de
28.04.2012

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital
Parte 3: Especificação de transmissão de dados

Digital terrestrial television — Data coding and transmission specification for digital broadcasting
Part 3: Data transmission specification

ICS 33.160.01

ISBN 978-85-07-03295-3

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referência
ABNT NBR 15606-3:2012
97 páginas

ABNT
Av.Treze de Maio, 13 - 28º andar
20031-901 - Rio de Janeiro - RJ
Tel.: + 55 21 3974-2300
Fax: + 55 21 3974-2346
abnt@abnt.org.br
www.abnt.org.br

Sumário

Página

Anexos

Figuras

Tabelas

Prefácio

A Associação Brasileira de Normas Técnicas (ABNT) é o Foro Nacional de Normalização. As Normas Brasileiras, cujo conteúdo é de responsabilidade dos Comitês Brasileiros (ABNT/CB), dos Organismos de Normalização Setorial (ABNT/ONS) e das Comissões de Estudo Especiais (ABNT/CEE), são elaboradas por Comissões de Estudo (CE), formadas por representantes dos setores envolvidos, delas fazendo parte: produtores, consumidores e neutros (universidades, laboratórios e outros).

Os Documentos Técnicos ABNT são elaborados conforme as regras da Diretiva ABNT, Parte 2.

A Associação Brasileira de Normas Técnicas (ABNT) chama atenção para a possibilidade de que alguns dos elementos deste documento podem ser objeto de direito de patente. A ABNT não deve ser considerada responsável pela identificação de quaisquer direitos de patentes.

A ABNT NBR 15606-3 foi elaborada pela Comissão de Estudo Especial de Televisão Digital (ABNT/CEE-85). O Projeto circulou em Consulta Nacional conforme Edital nº 10, de 13.10.2011 a 12.12.2011, com o número de Projeto ABNT NBR 15606-3.

Esta Norma é baseada nos trabalhos do Fórum do Sistema Brasileiro de Televisão Digital Terrestre, conforme estabelecido no Decreto Presidencial nº 5 820, de 29.06.2006.

Esta terceira edição cancela e substitui a edição anterior (ABNT NBR 15606-3:2010), a qual foi tecnicamente revisada.

A ABNT NBR 15606, sob o título geral "*Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital*", tem previsão de conter as seguintes partes:

— Parte 1: Codificação de dados;

— Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações;

— Parte 3: Especificação de transmissão de dados;

— Parte 4: Ginga-J – Ambiente para a execução de aplicações procedurais;

— Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações;

— Parte 6: Java DTV 1.3;

— Parte 7: Ginga-NCL – Diretrizes operacionais para as ABNT NBR 15606-2 e ABNT NBR 15606-5;

— Parte 8: Ginga-J – Diretrizes operacionais para a ABNT NBR 15606-4;

— Parte 9: Diretrizes operacionais para a ABNT NBR 15606-1.

O Escopo desta Norma Brasileira em inglês é o seguinte:

Scope

This part of ABNT NBR 15606 provides a data transmission specification for the digital data broadcasting scheme.

This part of ABNT NBR 15606 is apllied to data transmission for data broadcasting carried out as part of digital data broadcasting.

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital
Parte 3: Especificação de transmissão de dados

1 Escopo

Esta parte da ABNT NBR 15606 fornece uma especificação de codificação e transmissão de dados para o esquema de transmissão digital.

Esta parte da ABNT NBR 15606 se aplica à transmissão de dados realizada como parte da transmissão digital de dados.

2 Referências normativas

Os documentos relacionados a seguir são indispensáveis à aplicação deste documento. Para referências datadas, aplicam-se somente as edições citadas. Para referências não datadas, aplicam-se as edições mais recentes do referido documento (incluindo emendas).

ABNT NBR 15602-3, *Televisão digital terrestre – Codificação de vídeo, áudio e multiplexação – Parte 3: Sistemas de multiplexação de sinais*

ABNT NBR 15603-1, *Televisão digital terrestre – Multiplexação e serviços de informação (SI) – Parte 1: SI do sistema de radiodifusão*

ABNT NBR 15603-2:2007, *Televisão digital terrestre – Multiplexação e serviços de informação (SI) – Parte 2: Estrutura de dados e definições da informação básica de SI*

ABNT NBR 15603-3, *Televisão digital terrestre – Multiplexação e serviços de informação (SI) – Parte 3: Sintaxes e definições de informação estendida do SI*

ABNT NBR 15606-1, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiofusão digital – Parte 1: Codificação de dados*

ABNT NBR 15606-2:2011, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiofusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações*

ABNT NBR 15606-4:2010, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 4: Ginga-J – Ambiente para a execução de aplicações procedurais*

ABNT NBR 15607-1, *Televisão digital terrestre – Canal de interatividade – Parte 1: Protocolos, interfaces físicas e interfaces de software*

ABNT NBR 15608-3, *Televisão digital terrestre – Guia de operação – Parte 3: Multiplexação e serviço de informação (SI) – Guia para implementação da ABNT NBR 15603:2007*

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

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

ISO/IEC 8859-15, *Information technology – 8-bit single-byte coded graphic character sets – Part 15: Latin alphabet N° 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*

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*

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

ARIB STD-B10:2008, *Service information for digital broadcasting system*

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

ARIB STD-B24:2008, *Data coding and transmission specification for digital 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)*

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 EN 300 468:2010, *Digital video broadcasting (DVB) – Specification for service information (SI) in DVB systems*

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

ETSI EN 301 790, *Digital video broadcasting (DVB) – Interaction channel for satellite distribution systems*

ETSI TR 101 202, *Digital video broadcasting (DVB) – Implementation guidelines for Data Broadcasting*

ETSI TS 101 162, *Digital video broadcasting (DVB) – Allocation of service information (SI) and data broadcasting codes for digital video broadcasting (DVB) systems*

ETSI TS 101 812:2006, *Digital video broadcasting (DVB) – Multimedia home platform (MHP) specification 1.1.1*

GEM 1.3:2011, *Globally executable MHP (GEM) Especification 1.3*

RFC 768, *User datagram protocol*

RFC 791, *Internet protocol – Darpa internet program – Protocol specification*

RFC 793, *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, *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 2396, *Uniform Resource Identifiers (URI): Generic Syntax*

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

RFC 2818, *HTTP Over TLS*

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*

3 Termos e definições

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se os seguintes termos e definições.

3.1
***buffer* de transporte**
buffer que decodifica um pacote de *stream* de transporte em um decodificador-alvo de um *stream* de transporte MPEG2

3.2
cabeçalho do pacote *Packetizer Elementary Stream*
campo que compreende a primeira parte de um pacote *Packetizer Elementary Stream* (PES)

3.3
carrossel de dados
método que envia qualquer conjunto de dados repetidamente para que os dados possam ser baixados via transmissão sempre que necessário

NOTA Este método é especificado na ISO/IEC 13818-6.

3.4
comando e controle de armazenamento de mídia digital
DSMCC
método que suporta o acesso a arquivos e *streams* em serviço digital interativo

3.5
comitê para sistema de televisão avançado
ATSC
comitê com o propósito de padronizar o sistema de transmissão digital nos EUA

3.6
conselho audiovisual digital
DAVIC
consórcio industrial para padronização do serviço de multimídia interativa

3.7
conteúdo
grupo de dados transmitidos por meio de um programa de transmissão de dados ou serviço de comunicações bidirecionais que é gerado pelo programa de transmissão de dados para servir como parte do programa de transmissão de dados

NOTA Como um termo no contexto de transmissão, "conteúdo" indica um conjunto de *streams* no programa que envia o grupo de dados. Um único programa de transmissão pode ser composto de múltiplos conteúdos.

3.8
conteúdo local
parte do conteúdo de transmissão de dados contido em um único evento de dados

NOTA Geralmente, um conteúdo local é um grupo de dados agrupados com base no contexto ou para uma maior conveniência de produção de programa.

3.9
Digital Storage Media – Command and Control User-to-User Interface
DSM-CC U-U
protocolo ISO/IEC que permite o acesso a um servidor através de um cliente

3.10
endereço
protocolo utilizado para definir um nome de servidor utilizando o PPP

NOTA Esta definição está de acordo com a RFC 1877.

3.11
ethernet
padrão de *local area network* (LAN) que define uma rede baseada em barramento empregando o CSMA/CD (acesso múltiplo de sentido de portador/detecção de colisão) para controle de acesso

NOTA Esta definição está de acordo com a IEEE 802.

3.12
evento de dados
conjunto de *streams* de transmissão de dados que representa um grupo de conteúdo de transmissão de dados a ser distribuído com os tempos de início e término pré-configurados

NOTA O conceito de evento de dados (*data event*) é introduzido para permitir que um grupo de conteúdo de transmissão de dados seja alternado para outro, se estiverem ou não no mesmo programa, conforme necessário. Em outras palavras, um evento de dados é independente de um evento.

3.13
formato de adaptação
formato utilizado em um cabeçalho DSM-CC e que é uma forma de informação inserida em uma área de adaptação que codifica a informação para atender a uma solicitação, dependendo da rede de distribuição

3.14
host
máquina
dispositivo de ponto de acesso ou dispositivo de servidor, necessário para serviços de transmissão bidirecional

3.15
hypertext transfer protocol
HTTP
camada de aplicação para transmitir dados através da World Wide Web

NOTA Esta definição está de acordo com a RFC 1954.

3.16
identificador de pacote
PID
identificador de pacote de um *stream* de transporte MPEG-2

3.17
informação de serviço
SI
dados digitais que descrevem um arranjo de programas, um sistema de distribuição para transmissão de *streams* de dados, descrição de programas, informações de grade de programação/tempo de duração

NOTA Estes dados também transportam MPEG-2 PSI (Informações Específicas do Programa), bem como partes da extensão definidas de forma independente.

3.18
informação específica do programa
PSI
informação de controle de transmissão, que fornece a informação necessária para permitir a um receptor automaticamente demultiplexar e decodificar vários *streams* de programa que foram multiplexados

3.19
MIME
protocolo de camada de aplicação que fornece uma arquitetura de conteúdo que permite que dados multimídia, como arquivos de texto, áudio e imagens, em formato que não seja US-ASCII (*American Stardard Code for Information Interchange*), sejam transmitidos via *e-mail*

3.20
pacote PES
formato de dados usado para transmitir *streams* básicos que consiste em um cabeçalho de pacote PES e uma carga útil PES imediatamente seguindo o cabeçalho

3.21
private_stream_1
tipo de *stream* transmitido usando PES e que é usado para transmitir um *stream* privado sincronizado com outros *streams*

3.22
private_stream_2
tipo de *stream* transmitido usando PES e que é usado para transmitir um *stream* privado que não precisa ser sincronizado com outros *streams*

3.23
procedimento de controle de conexão de dados de alto nível
procedimento HDLC
procedimento de controle de transmissão com alta confiabilidade, usado para comunicação entre computadores, principalmente em LAN e *internet*

3.24
protocolo de datagrama de usuário
UDP
protocolo de camada de transporte que promove entrega de dados sem conexão entre duas máquinas

NOTA 1 Embora o UDP não suporte mensagens de reconhecimento, ele minimiza o protocolo elevado para maior eficiência em serviços de transmissão.

NOTA 2 Esta definição está de acordo com a RFC 768.

3.25
protocolo de controle de transmissão
TCP
protocolo de camada de transporte que promove distribuição de dados altamente confiável, de ponta a ponta, orientada por conexão, utilizando um mecanismo de detecção e correção de erro

NOTA Esta definição está de acordo com a RFC 793.

3.26
protocolo de *internet*
IP
protocolo de camada de rede que define o mecanismo de endereçamento na *internet* para permitir que os dados sejam transmitidos

NOTA Esta definição está de acordo com a RFC 791.

3.27
receptor *full-seg*
dispositivo capaz de decodificar informações de áudio, vídeo, dados etc., contidas na camada do fluxo de transporte de 13 segmentos destinada ao serviço fixo (*indoor*) e móvel

3.28
receptor *one-seg*
dispositivo que decodifica exclusivamente informações de áudio, vídeo, dados etc., contidas na camada "A" locada no segmento central dos 13 segmentos

3.29
rede digital de serviços integrados
ISDN
serviços integrados de áudio, vídeo e dados em um sistema digital, transmitidos em uma rede de telefonia pública

3.30
reservado
termo que, quando utilizado em sentenças que definem o *stream* de bit codificado, indica que o valor pode ser usado no futuro para extensões definidas pela ISO

NOTA Os bits reservados são configurados em 1.

3.31
reserved_future_use
termo que, quando utilizado em sentenças definindo o *stream* de bit codificado, indica que o valor pode ser utilizado em extensões definidas pela ISO no futuro

NOTA Os bits reservados são configurados em 1.

3.32
seção
estrutura sintática utilizada para mapear as informações de serviço e outros dados dentro de um pacote de *stream* de transporte

3.33
serviço de nome de domínio
DNS
protocolo utilizado pelo serviço que mapeia um nome de máquina em uma rede dentro de seu endereço de IP

NOTA Esta definição está de acordo com as RFC 1034 e RFC 1035.

3.34
tabela de informação de evento
EIT
tabela de informação de evento que contém dados relacionados a um evento e a um programa como um nome de evento (programa), horário de início e um período

3.35
tabela de mapeamento de programa
PMT
tabela que é parte do PSI

NOTA Esta tabela indica uma localização (PID de pacote de *stream* de transporte) de uma tabela de mapeamento de programa correspondente a cada serviço no *stream* multiplexado.

3.36
transmissão de vídeo digital
DVB
projeto para padronização do sistema de transmissão digital na Europa

4 Tipos de especificação de transmissão de dados

Os tipos de especificações para transmissão de dados e identificação de tipo de *stream* contidas em um PMT são apresentados na Tabela 1.

Tabela 1 – Tipos de especificação de transmissão

Especificação da transmissão	Função majoritária e usabilidade	Identificação do tipo de *stream*
PES independente	Utilizado para *streams* de dados síncronos e assíncronos para serviços de radiodifusão	0x06
Carrossel de dados/ objetos	Utilizado para transferências de dados em geral: sincronizados e assíncronos para serviços de radiodifusão. Aplicado à transmissão de dados para serviços de *download* e serviços multimídia	0x0B, 0X0D [a]
Mensagem de eventos	Utilizada para notificações sincronizadas e assíncronas referentes à aplicações no TA a partir da estaca de *broadcast*. Utilizada para serviços de multimídia	0x0C, 0X0D [b]
Protocolos de canal de interatividade	Protocolos de transmissão utilizados em redes fixas como redes PSTN/ISDN e redes de celular, incluindo rede celular/PHS com comunicações bidirecionais [d]	—
Encapsulamento multiprotocolo	Datagramas são encapsulados em *datagram_sections* que são compatíveis com o formato *DSMCC_section* para dados privados	0x0A [c]
Data piping	Protocolo que permite inserir dados de uma rede de radiodifusão diretamente no *payload* do pacote MPEG-2	0x7E

[a] Quando um *stream* não contém dados DSM-CC, a não ser um carrossel de dados, 0x0B ou 0x0D é usado e, quando também ele tem outros dados DSM-CC, 0x0D é usado.

[b] Quando um *stream* não contém dados DSM-CC, a não ser como mensagem de evento, 0x0C ou 0x0D é usado e, quando também ele tem outros dados DSM-CC, 0x0D é usado.

[c] Quando um *stream* não contém dados DSM-CC, a não ser dados de encapsulamento de multiprotocolos (MPE), 0x0A é usado e, quando também ele tem outros dados DSM-CC, 0x0D é usado.

[d] PSTN: rede telefônica comutada pública.

5 Especificação de transmissão do carrossel de dados

5.1 Transmissão com carrossel de dados DSM-CC

A especificação de transmissão do carrossel de dados é destinada a implementar a transmissão geral sincronizada ou assíncrona sem a necessidade de dados *streaming*, tais como *download* de dados

para uma unidade receptora ou transmissão de conteúdos para serviços de multimídia. A especificação de transmissão do carrossel de dados definida nesta parte da ABNT NBR 15606 está baseada na especificação do carrossel de dados DSM-CC estabelecida na ISO/IEC 13818-6.

A transmissão repetida de dados, como é definida na especificação do carrossel de dados DSC-CC, permite à unidade receptora obter dados em demanda em qualquer momento durante um período de transmissão.

Os dados são transmitidos em uma unidade modular formada por blocos onde todos os blocos, exceto aqueles ao final do módulo, têm o mesmo tamanho e cada bloco é transmitido em seções.

Na transmissão destes dados são utilizadas a mensagem *download* de bloco de dados (referida como mensagem DDB) e a mensagem de indicação de informação *download* (referida como mensagem DII). As duas mensagens são componentes do protocolo de *download* do usuário da rede especificado na ISO/IEC 13818-6. O corpo de dados é transmitido pela mensagem DDB com cada módulo dividido dentro dos blocos. Para informações adicionais relacionadas à transmissão PSI/SI, ver o Anexo B.

5.2 Mensagem de controle DSM-CC

5.2.1 Mensagem de indicação de informação de *download* (DII)

Uma mensagem DII faz parte de uma mensagem de controle DSM-CC. Assim, a mensagem DII transmite o conteúdo da mensagem retendo-o no *userNetworkMessage()* na seção DSM-CC.

A versão da mensagem DII é indicada pelo *transaction_number* (número da transação) no campo *transaction_id* (identificação de transação) do *dsmccMessageHeader*. Este número de versão é comum a todas as mensagens DII do carrossel de dados e é incrementado em um quando o conteúdo de uma ou mais mensagens DII for trocado.

5.2.2 Sintaxe e semântica da mensagem DII

A estrutura dos dados da mensagem DII é apresentada na Tabela 2.

Tabela 2 – Estrutura dos dados da mensagem de indicação de informação de *download*

Sintaxe	Número de bits	Mnemônico
DownloadInfoIndication() {		
dsmccMessageHeader()		
downloadId	32	*uimsbf*
blockSize	16	*uimsbf*
windowSize	8	*uimsbf*
ackPeriod	8	*uimsbf*
tCDownloadWindow	32	*uimsbf*
tCDownloadScenario	32	*uimsbf*
compatibilityDescriptor()		
numberOfModules	16	*uimsbf*

Tabela 2 (continuação)

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A semântica dos campos DII deve ser a seguinte:

— ***dsmccMessageHeader ()* (cabeçalho da mensagem DSM-CC)**: conforme especificado em 5.3;

— ***downloadId* (identificador de *download*)**: campo de 32 bits que serve como um rótulo para a identificação única do carrossel de dados. No caso de evento de operação de dados, *data_event_id* (identificação do evento de dados) deve ser inserido nos bits 28-31 do *downloadId* (identificador de *download*). Caso contrário, a faixa e os valores para assegurar a unicidade são especificados em um padrão operacional;

— ***windowSize***: campo de 8 bits que não é usado pela transmissão do carrossel de dados e o valor deve ser ajustado em 0;

— ***ackPeriod***: campo de 8 bits que não é usado pela transmissão do carrossel de dados e o valor deve ser ajustado em 0;

— ***tCDownloadWindow***: campo de 32 bits que não é usado pela transmissão do carrossel de dados e o valor deve ser ajustado em 0;

— ***tCDownloadScenario***: campo de 32 bits que indica o período de limite de tempo em que se presume que o *download* está completo, em microssegundos;

— ***compatibilityDescriptor()***: estrutura do descritor de compatibilidade (*compatibilityDescriptor()*) que é especificada na ISO/IEC 13818-6 e que deve ser configurada neste campo. Quando o conteúdo da estrutura do *compatibilityDescriptor()* não é necessária, o *descriptorCount* deve ser configurado em 0x0000 e, assim, a extensão do campo deve ser de 4 bytes;

— ***numberOfModules* (número de módulo)**: campo de 16 bits que indica o número de módulos descritos no enlace seguinte nesta mensagem DII;

— ***moduleId* (identificador de módulo)**: campo de 16 bits que indica a identificação do módulo descrito nos seguintes campos: *moduleSize, moduleVersion e moduleInfoByte*;

— ***moduleSize* (extensão do módulo)**: campo de 32 bits que indica a extensão do byte do módulo. Quando a extensão do byte do módulo não é conhecida, deve ser configurada em 0;

— ***moduleVersion***: campo de 8 bits que indica a versão deste módulo;

— ***moduleInfoLength* (extensão da informação do módulo)**: campo de 8 bits que indica a extensão byte da área de informação do módulo;

— ***moduleInfoByte* (informação do módulo)**: campo de unidade de 8 bits que pode ser usado para inserir descritores relacionados ao módulo. Estes descritores são definidos em 5.4. Os valores de tag dos descritores a serem inseridos são definidos na Tabela 5;

— ***privateDataLength***: campo de 16 bits que indica a extensão do byte do campo *PrivateDataByte*;

— ***privateDataByte* (dados privados)**: campo de unidade de 8 bits que pode ser usado para conter uma estrutura de dados em um formato de descritor. A estrutura de dados é definida com base em um formato de codificação de dados ou pelo operador do serviço.

A semântica dos valores de *tag* dos descritores para este campo é definida na Tabela 3. Os descritores possíveis para este campo são aqueles definidos em 5.4 e por um formato de codificação de dados.

Tabela 3 – Semântica dos *tags* dos descritores da área de informação de módulo e área privada no DI

Valor de *tag* do descritor	Semântica
0x01 – 0x7F	Valores de *tag* reservados de descritores compatíveis a serem inseridos na área de informação do módulo e área privada (ver 5.4)
0x80 – 0xBF	Valores de *tag* disponíveis de descritores definidos por um operador de serviço
0xC0 – 0xEF	Valores de *tag* reservados de descritores a serem inseridos na área de informação do módulo e área privada (ver 5.4)
0xF0 – 0xFE	Valores de *tag* reservados de descritores definidos com base em um formato de codificação de dados

5.3 Sintaxe e semântica do *dsmccMessageHeader()*

A estrutura de dados do *dsmccMessageHeader()* é definida na Tabela 4.

Tabela 4 – Estrutura de dados do *dsmccMessageHeader*

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A semântica do *dsmccMessageHeader()* deve ser a seguinte:

— ***protocolDiscriminator***: campo de 8 bits que deve ser configurado em 0x11 e indica que esta mensagem é do tipo MPEG-2 DSM-CC;

— ***dsmccType* (tipo DSM-CC)**: campo de 8 bits que indica o tipo da mensagem MPEG-2 DSM-CC. Em uma mensagem DII para transmissão do carrossel de dados, deve ser configurado em 0x03 (mensagem *download* U-N);

— ***messageId* (identificador do tipo da mensagem)**: campo de 16 bits que identifica o tipo da mensagem DSM-CC. Em uma mensagem DII, deve ser configurado em 0x1002;

— ***transaction_id* (identificador de transação)**: campo de 32 bits que identifica a mensagem e tem a função de controlar a versão. O formato da *transaction_id* é mostrado na Figura 1. O campo *Transaction Number* nos bits 0-29 deve ser usado para identificar a versão da DII, como especificado na ISO/IEC 13818-6. O valor de bits 30-31 deve ser configurado em '10' (*TransactionId* alocado pela rede) conforme definido no *Transaction Id Originator*, como especificado na ISO/IEC 13818-6;

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

— ***messageLength***: campo de 16 bits que indica o número de bytes da mensagem imediatamente após este campo. Isto é, o valor é uma soma da extensão *payload* e da extensão *dsmccAdaptationHeader()*;

— ***dsmccAdaptationHeader()***: a estrutura de dados deste campo é definida em 5.5.3.

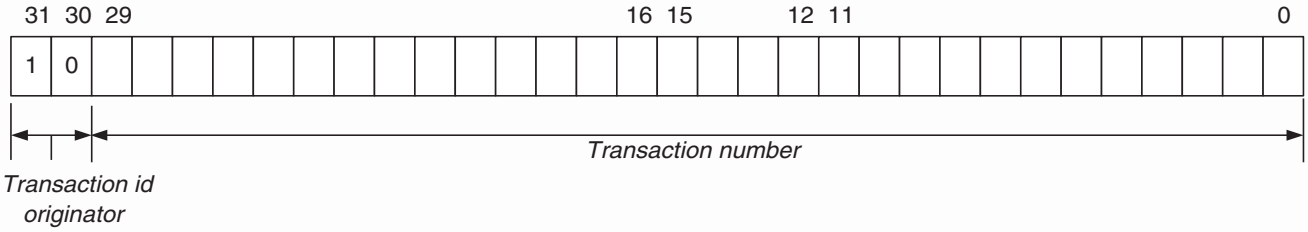


Figura 1 – Formato da *transaction_id*

5.4 Descritores da área de informação do módulo e área privada

5.4.1 Tipos de descritores

Os tipos de descritores usados em uma área de informação do módulo e em uma área privada são mostrados na Tabela 5. Qualquer destes descritores pode ser usado na área de informação do módulo e/ou em uma área privada, conforme necessário. Os descritores contidos em uma área privada em uma DII se aplicam aos módulos na DII. Quando a área de informação do módulo e a área privada têm o mesmo conjunto de descritores, apenas os descritores na área de informação do módulo são habilitados.

Tabela 5 – Tipos de descritores

Valor de *tag*	Descritor	Função	Área de informação do módulo	Área privada
0x01	*type_descriptor*	Tipo de módulo (forma MIME etc.)	X	-
0x02	*name_descriptor*	Nome do módulo (nome do arquivo)	X	-
0x03	*info_descriptor*	Informação do módulo (tipo de caractere)	X	X
0x04	*module_link_descriptor*	Informação do *link* (id do módulo)	X	-
0x05	*CRC32_descriptor*	CRC32 do módulo	X	-
0x06	*location_descriptor*		X	X
0x07	*est_download_time_descriptor*	Tempo estimado de *download*(s)	X	X
0x08 - 0x7F	Reservado para o futuro		-	-
0x80 - 0xBF	Disponível para um *broadcaster*		-	-
0xC0 - 0xC1	Reservado para o futuro		-	-
0xC2	*compression_Type_descriptor*	Algoritmo de compressão quando o modulo é transmitido	X	-
0xC3 - 0xCC	Reservado para o futuro		-	-
0xCD - 0xEE	Reservado para o futuro		-	-

5.4.2 Descritor de tipo

O descritor de tipo (ver Tabela 6) indica o tipo de arquivo transmitido como um módulo único, implementando a transmissão do carrossel de dados com base nesta parte da ABNT NBR 15606, que especifica que um arquivo único é transmitido como um módulo único.

Tabela 6 – Descritor de tipo

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

A semântica de campo no descritor de tipo deve ser a seguinte:

— ***text_char***: campo de 8 bits. A sequência deste campo indica o tipo de mídia de acordo com a RFC 1521.

5.4.3 Descritor do nome

O descritor do nome (ver Tabela 7) indica o nome do arquivo transmitido como um módulo único, implementando a transmissão do carrossel de dados com base nesta parte da ABNT NBR 15606, que especifica que um arquivo único é transmitido como um módulo único. Porém, quando há o descritor *Module Link*, o descritor do nome não pode estar presente em outra posição a não ser no módulo 0x00 na DII.

Tabela 7 – Descritor do nome

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

A semântica de campo no descritor do nome deve ser a seguinte:

— ***text_char***: campo de 8 bits. A sequência deste campo indica o nome do arquivo transmitido como um módulo único usando a especificação de codificação de dados ou um código de caractere especificado em um padrão operacional.

5.4.4 Descritor de informação

O descritor de informação (ver Tabela 8) descreve as informações relacionadas ao módulo.

Tabela 8 – Descritor de informação

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A semântica de campos no descritor de informação deve ser a seguinte:

— ***ISO_639_language_code***: campo de 24 bits que identifica a linguagem usada na área *text_char*. O código da linguagem é representado por três caracteres alfabéticos especificados na ISO 639-2. Cada caractere é codificado dentro de uma representação de 8 bits de acordo com a ISO/IEC 8859-1 e inserido dentro de um campo de 24 bits nessa ordem;

— ***text_char***: campo de 8 bits. A sequência deste campo indica a informação textual relacionada ao arquivo transmitido como um módulo único usando a especificação de codificação de dados ou um código de sinalização especificado em uma norma operacional.

5.4.5 Descritor do *Module_link*

O descritor *Module_link* (ver Tabela 9) gera uma lista de módulos ligados a outros módulos. Por ser a extensão do campo número de blocos de uma mensagem DDB restrita a 16 bits, o tamanho máximo de um módulo na transmissão do carrossel de dados é de 256 Mbytes. Quando é transmitido um arquivo maior que 256 Mbytes, o arquivo é dividido em dois ou mais módulos antes de ser enviado e esta informação é associada ao descritor do *Module_link*.

Tabela 9 – Descritor do *Module_link*

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

A semântica de campos no descritor *Module_link* deve ser a seguinte:

— ***position***: campo de 8 bits que indica a relação de posição com módulo conectado. “0x00” indica que o módulo está localizado no topo do *link*, “0x01 indica que o módulo está no meio e “0x02” indica que está no final;

— ***module Id***: campo de 16 bits que é a identificação do módulo conectado. Quando a posição é “0x02”, o valor deste campo é ignorado.

5.4.6 Descritor da localização

O *location_descriptor* contém a localização do PID onde os blocos, módulos ou grupos podem ser encontrados contendo os dados do carrossel. A Tabela 10 mostra a sintaxe do *location_descriptor*.

Tabela 10 – Sintaxe do *location_descriptor*

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

A semântica do *location_descriptor* deve ser a seguinte:

— ***descriptor_tag***: campo de 8 bits que identifica o descritor. O *location_descriptor* está configurado em 0x06;

— ***descriptor_length***: campo de 8 bits que especifica o número de bytes do descritor imediatamente após este campo;

— ***location_tag***: campo de 8 bits que tem o mesmo valor que o campo *component_tag* no descritor identificador do *stream*.

5.4.7 Descritor CRC

O descritor CRC (ver Tabela 11) descreve o valor CRC do módulo completo.

Tabela 11 – Descritor CRC

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

A semântica de campo no descritor CRC deve ser a seguinte:

— **CRC_32**: campo de 32 bits que armazena o valor CRC calculado para o módulo completo. O valor CRC deve ser calculado como definido na ABNT NBR 15603-2:2007, Anexo B.

5.4.8 Descritor de tempo estimado de *download*

O descritor de tempo estimado de *download* (ver Tabela 12) descreve o período estimado necessário para o *download* do módulo.

Tabela 12 – Descritor de tempo estimado de *download*

Sintaxe	Número de bits	Mnemônico
est_download_time_descriptor() {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
est_download_time	32	*uimsbf*
}		

A semântica do campo no descritor de tempo estimado de *download* deve ser a seguinte:

— ***est_download_time***: campo de 32 bits que indica o período estimado, em segundos, necessário para fazer o *download* do módulo.

5.4.9 Descritor de tipo de compressão

O descritor de tipo de compressão (ver Tabela 13) indica que o módulo foi comprimido no formato zlib baseado na RFC 1950 e mostra seu algoritmo de compressão e o tamanho do módulo antes da compressão em bytes. Um módulo que não tenha sido comprimido não tem esse descritor.

Tabela 13 – Descritor de tipo de compressão

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

A semântica de campos no descritor de tipo de compressão deve ser a seguinte:

— ***compression_type***: campo de 8 bits que define o tipo de compressão usado para comprimir o módulo;

— ***original_size***: campo de 32 bits que indica o tamanho do módulo antes da compressão em bytes.

5.5 Mensagem *DownloadDataBlock* (DDB)

5.5.1 Sintaxe e semântica da mensagem DDB

O conteúdo de uma mensagem DDB é transmitido por armazenamento no campo *downloadDataMessage()* na seção DSM-CC.

Uma mensagem DDB é a estrutura de dados que transmite blocos de dados (ver Tabela 14). Um módulo pode ser dividido com extensão fixada para formar blocos. Nesse caso, cada bloco é representado com um número de bloco na mensagem DDB para permitir que uma unidade receptora reorganize os blocos na ordem pretendida.

De acordo com o especificado na ISO/IEC 13818-6, quando as mensagens DDB são transmitidas em MPEG-2 TS, apenas as mensagens DDB que têm o mesmo *downloadId* devem ser incluídas no mesmo pacote PID. Isso significa que as mensagens DDB em dois carrosséis diferentes não podem ser apresentadas em um único *stream* elementar.

Tabela 14 – Estrutura de dados do bloco de dados de *download*

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

Os campos de DDB devem ser os seguintes:

— ***moduleId***: campo de 16 bits que indica o número de identificação ao qual este bloco pertence;

— ***moduleVersion***: campo de 8 bits que indica a versão do módulo ao qual este bloco pertence;

— ***blockNumber***: campo de 16 bits que indica a posição deste bloco dentro do módulo. O primeiro bloco de um módulo deve ser representado pelo bloco número 0;

— ***blockDataByte***: campo de 8 bits. O tamanho de uma série da área de dados do bloco é igual ao tamanho do bloco da DII, isto é, o tamanho dos blocos divididos a partir de um módulo. Porém, o número do último bloco no módulo pode ser menor que o tamanho de bloco descrito na DII.

5.5.2 Sintaxe e semântica do *dsmccDownloadDataHeader()*

A estrutura de dados do *dsmccDownloadDataHeader()* é definida na Tabela 15.

Tabela 15 – Estrutura de dados do *dsmccDownloadDataHeader*

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

Os campos do *dsmccDownloadDataHeader()* devem ser os seguintes:

— ***protocol discriminator***: campo de 8 bits que é configurado em 0x11 e indica que esta mensagem é uma mensagem DSM-CC MPEG-2 *dsmccType*. Este campo de 8 bits indica o tipo da mensagem DSM-CC MPEG-2 e é configurado em 0x03 (mensagem *download* U-N) para a mensagem DDB na transmissão do carrossel de dados;

— ***messageId***: campo de 16 bits que identifica o tipo da mensagem DSM-CC e é configurado em 0x1003 para uma mensagem DDB;

— ***downloadId***: campo de 32 bits que é configurado no mesmo valor que o identificador de *download* na mensagem DII correspondente;

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

— ***dsmccAdaptationHeader()***: a estrutura de dados deste campo é definida em 5.5.3;

— ***messageLength***: campo de 16 bits que indica a extensão da mensagem, não incluindo este campo e sua área precedente em bytes. O valor é idêntico à soma da extensão de *payload* e a extensão de *dsmccAdaptationHeader*.

5.5.3 Sintaxe do *dsmccAdaptationHeader()*

No *dsmccMessageHeader()*, que é o cabeçalho de uma mensagem DII, e no *dsmccDownloadDataHeader()*, que é o cabeçalho de uma mensagem DDB, pode ser estabelecida a estrutura de dados comum do *dsmccAdaptationHeader()*.

A estrutura de dados do *dsmccAdaptationHeader* é indicada na Tabela 16.

Tabela 16 – Estrutura do *dsmccAdaptationHeader*

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

A semântica do *dsmccAdaptationHeader()* deve ser a seguinte:

— ***adaptationType***: campo de 8 bits que indica o tipo de cabeçalho de adaptação. O valor deste campo indica um formato de adaptação conforme a Tabela 17.

Tabela 17 – Tipo de adaptação

Tipo de adaptação	Formato da adaptação	Definição na ISO/IEC 13818-6
0x00	Reservado	O mesmo que na coluna à esquerda
0x01	Reservado	DSM-CC Acesso condicional
0x02	Reservado	DSM-CC identificador de usuário
0x03	*DIIMsgNumber*	O mesmo que na coluna à esquerda
0x04-0x7F	Reservado	O mesmo que na coluna à esquerda
0x80-0xFF	Definição do usuário	O mesmo que na coluna à esquerda
NOTA Para os tipos de adaptação utilizados nesta parte da ABNT NBR 15606, a operação do formato de adaptação de definição do usuário do tipo de adaptação 0x80 – 0xFF é opcionalmente feita por um operador de serviço.		

5.5.4 Sintaxe da seção DSM-CC

As mensagens DII e DDB são transmitidas usando as seções DSM-CC, como demonstrado na Tabela 18.

Tabela 18 – Seção DSM-CC (transmissão de mensagens DII/DBB)

Sintaxe	Número de bits	Mnemônico
DSMCC_section () {		
table_id	8	*uimsbf*
section_syntax_indicator	1	*bslbf*

Tabela 18 (continuação)

Sintaxe	Número de bits	Mnemônico
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*
}		

A semântica da seção DSM-CC deve ser a seguinte:

— ***table_id***: campo de 8 bits que contém o número de identificação do tipo de dados na seção de *payload* DSM-CC. Com base no valor deste campo, aplica-se uma regra de codificação específica para o campo seguinte na seção DSM-CC. A tabela dos valores de identificação é demonstrada na Tabela 19, como especificado na ISO/IEC 13818-6;

— ***section_syntax_indicator***: campo de 1 bit. Quando é configurado em 1, indica que existe um CRC32 ao final da seção. Quando é configurado em ***0, indica que existe uma soma de verificação. Deve ser configurado em 1 para a transmissão das mensagens DII e DDB;

— ***private_indicator***: campo de 1 bit que armazena o valor complementar do *flag* do *section_syntax_indicator*;

— ***dsmcc_section_length***: campo de 12 bits que indica o número de bytes da área desde o início do campo, imediatamente após esse campo até o fim da seção. O valor neste campo não pode exceder 4 093 bytes;

— ***table_id_extension***: campo de 16 bits que é configurado como demonstrado abaixo, de acordo com o campo *table_id*:

  — quando o valor do campo *table_id* é igual a 0x3B, este campo deve transportar uma cópia dos 2 bytes menos significativos do campo *transaction_id*;

  — quando o valor do campo *table_id* é igual a 0x3C, este campo deve transportar uma cópia do campo *module_id*;

— ***version_number***: campo de 5 bits que é configurado de acordo com o identificador de tabela (*table_id*). Quando o valor do campo *table_id* é igual a 0x3B, este campo deve ser configurado em "0". Quando o valor do campo *table_id* é igual a 0x3C, deve ser configurado nos 5 bits menos significativos do campo versão do módulo;

— ***current_next_indicator***: designação de 1 bit que indica que a subtabela está ativa quando está em "1". Quando está em "0", a subtabela enviada ainda não foi aplicada e usada como a próxima subtabela. Quando o valor do campo *table_id* é igual a um valor na faixa de 0x3A a 0x3C, este campo deve ser configurado em "1";

— ***section_number***: campo de 8 bits que indica o número da seção da primeira seção na subtabela. Quando a seção contém uma mensagem DII, este campo deve ser configurado em 0. Quando esta seção contém uma mensagem DDB, este campo deve transportar uma cópia dos 8 bits menos significativos do número do bloco da DDB;

— ***last_section_number***: campo de 8 bits que indica o número da última seção (seção que tem o número máximo da seção) da subtabela à qual pertence a seção;

— ***userNetworkMessage()***: mensagem DII é armazenada;

— ***downloadDataMessage()***: mensagem DDB é armazenada.

Tabela 19 – *Table_id*

table_id	Tipo de seção DSM-CC	Definição na ISO/IEC 13818-6
0x3A	Reservado	Cápsula multiprotocolo [a]
0x3B	Mensagem DII	Mensagem U-N incluindo DII
0x3C	Mensagem DDB	O mesmo que na coluna à esquerda
0x3D	Descritor de *Stream*	O mesmo que na coluna à esquerda

Tabela 19 (continuação)

table_id	Tipo de seção DSM-CC	Definição na ISO/IEC 13818-6
0x3E	Dados privados	O mesmo que na coluna à esquerda
0x3F	Reservado	O mesmo que na coluna à esquerda
[a] Ver ISO/IEC 13818-6.		

6 Especificação do carrossel de objetos

6.1 Escopo do carrossel de objetos

A especificação do carrossel de objetos foi adicionada para suportar os serviços de transmissão de dados que requerem transmissão periódica de objetos DSM-CC U-U através das redes de transmissão compatíveis com o sistema brasileiro de televisão digital terrestre (SBDTV).

A transmissão de dados de acordo com a especificação do sistema brasileiro de televisão digital terrestre para carrossel de objetos é feita de acordo com a DSM-CC do carrossel de objetos e a especificação de carrossel de dados DSM-CC que são definidas em MPEG-2 DSM-CC (ver ISO/IEC 13818-6:1998, Seção 5).

6.2 Especificação do transporte de dados

6.2.1 Endereço de carrossel NSAP

A especificação SBTVD para carrossel de objetos é baseada na especificação DSM-CC de carrossel de objetos (ver ISO/IEC 13818-6). Um carrossel de objetos SBTVD representa um domínio de serviço particular que consiste em uma coleção de objetos DSM-CC U-U dentro de uma rede SBTVD. O domínio de serviço tem uma porta de serviço que apresenta um gráfico de serviços e nomes de objetos para os receptores.

A única identificação da porta de serviço nas redes de transmissão é feita por meio do endereço *Network Service Access Point* (NSAP) do carrossel, conforme definido em DSM-CC (ver ISO/IEC 13818-6).

Este endereço contém uma parte específica da rede que deve tornar o endereço único dentro do ambiente de rede usado. O endereço NSAP do carrossel é usado para referir-se ao carrossel de objetos a partir de outro domínio de serviço. Para os ambientes do SBTVD, a sintaxe e a semântica do endereço NSAP do carrossel são definidas abaixo.

6.2.2 Estrutura do endereço NSAP do carrossel

O endereço NSAP do carrossel tem uma estrutura conforme a mostrada na Figura 2 (ver ISO/IEC 13818-6).

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

Figura 2 – Formato do endereço NSAP do carrossel

A semântica do AFI (identificador de autorização e formato), tipo, *carouselId* e especificador são definidos na ISO/IEC 13818-6. Em particular:

— **AFI**: campo de 8 bits que deve ser configurado no valor de 0x00 para indicar o uso privado do formato NSAP (ver ETSI EN 301 192);

— ***Type* (Tipo)**: campo de 8 bits que deve ser configurado em 0x00 para indicar o uso do endereço NSAP para carrosséis de objetos;

— ***carouselId***: campo de 32 bits que deve ser configurado no identificador do carrossel de objetos, ou seja, o campo *carouselId*;

— ***specifier* (especificador)**: campo de 32 bits que deve transportar o campo *specifierType* (configurado no valor de 0x01) e o código OUI (Identificador Único Organizacional) como definido na DSM-CC (ver ISO/IEC 13818-6:1998, Seção 5);

— ***privateData***: campo que deve transportar a estrutura *ginga_service_location* que é definida na Tabela 20.

Tabela 20 – Sintaxe para a estrutura *ginga_service_location*

Sintaxe	Número de bits	Mnemônico
ginga_service_location() {		
transport_stream_id	16	*uimsbf*
org_network_id	16	*uimsbf*
service_id	16	*uimsbf*
Reserved	32	*bslbf*
}		

A semântica da estrutura *ginga_service_location* deve ser a seguinte:

— ***transport_stream_id***: campo de 16 bits que identifica o *stream* de transporte no qual o carrossel é transmitido;

— ***org_network_id***: campo de 16 bits que identifica o *network_id* do sistema de entrega do qual se origina o carrossel;

— ***service_id***: campo de 16 bits que fornece o identificador do serviço que contém o carrossel de objetos. O *service_id* é o mesmo que o *program_number* na *program_map_section* associada.

6.3 Descritores

NOTA Todos os descritores do carrossel de dados são os mesmos usados na Seção 5.

6.3.1 Especificação PSI e SI

O serviço de transmissão de dados indica o uso de um carrossel de objeto SBTVD pela inclusão de um ou mais descritores de componentes dos dados – descritor de ID do carrossel – descritor de *tag* de associação, de acordo com a ARIB STD-B23.

Cada descritor deve apontar para um carrossel de objetos e ser associado a um *stream* particular via um identificador *component_tag*. Em particular, o valor do campo *component_tag* é idêntico ao valor do campo *component_tag* de um *stream_identifier_descriptor* (ver ETSI EN 300 468) que pode estar presente na seção de mapa do programa PSI para o *stream* que é usado como *stream* de dados.

Cada descritor de transmissão de dados permite o uso de protocolos de camadas mais altas baseados no critério de linguagem usando uma lista de nomes de objetos.

Um carrossel de objeto pode ser implementado usando serviços de transmissão de dados múltiplos. Os serviços de transmissão de dados podem publicar que eles são parte de um carrossel de objeto particular pela inclusão do *carousel_identifier_descriptor* como definido pela DSM-CC (ver ISO/IEC 13818-6) no primeiro enlace do descritor da tabela de mapa de programa.

Além disso, os carrosséis-objetos usam o conceito de *taps* (ver ISO/IEC 13818-6) para identificar os *streams* nos quais os objetos são transmitidos. A associação entre os *taps* e os *streams* do serviço de dados pode ser feita por um ou outro, usando o descritor *association_tag* definido na ISO/IEC 13818-6 ou o *stream_identifier_descriptor* definido na ETSI EN 300 468.

Em último caso, presume-se que o campo *component_tag* do descritor *stream_identifier* seja o byte de menor significação do valor *association_tag* indicado que tem o byte mais significativo configurado em 0x00.

Finalmente, os objetos de *stream* dentro dos carrosséis de objetos U-U podem ser ligados aos próprios *streams* elementares do serviço de transmissão de dados, aos *streams* elementares de outros serviços ou para completar serviços SBTVD. Se o objeto de *stream* for ligado aos streams elementares de outros serviços ou para completar os serviços SBTVD, a tabela de mapa de programa do serviço de transmissão de dados deve incluir o *deferred_association_tags_descriptor* no primeiro enlace do descritor.

6.3.2 *Deferred_association_tags_descriptor*

A sintaxe e a semântica do *deferred_association_tags_descriptor()* nas redes compatíveis com o SBTVD são descritas na Tabela 21.

Tabela 21 – *Deferred_association_tags_descriptor*

Sintaxe	Número de bits	Mnemônico
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*

Tabela 21 (continuação)

Sintaxe	Número de bits	Mnemônico
for (i=0;i<N2;i++){		
private_data_byte	8	*uimsbf*
}		
}		

A semântica do *deferred_association_tags_descriptor* deve ser a seguinte:

— ***descriptor_tag***: campo de 8 bits que deve ter o valor de 0x15;

— ***descriptor_length***: campo de 8 bits que especifica a extensão do descritor em bytes;

— ***association_tags_loop_length***: campo de 8 bits que define a extensão em bytes do enlace das tags de associação que seguem este campo;

— ***association_tag***: campo de 16 bits que contém a *association_tag* que está associada a um *stream* que não faz parte do serviço de transmissão de dados ou a outro serviço SBTVD;

— ***transport_stream_id***: campo de 16 bits que indica o *stream* de transporte no qual reside o serviço que está associado às *tags* de associação listadas;

— ***program_number***: campo de 16 bits que deve ser configurado no *service_id* do serviço que está associado às *tags* de associação listadas;

— ***private_data_byte:*** campo que deve conter a estrutura *deferred_service_location* definida na Tabela 22.

Tabela 22 – Sintaxe para a estrutura *deferred_service_location*

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

A semântica da estrutura *deferred_service_location* deve ser a seguinte:

— ***org_network_id***: campo de 16 bits que identifica o *network_id* do sistema de entrega a partir do qual se origina o serviço;

— ***private_data_byte***: campo de 8 bits que não é especificado nesta parte da ABNT NBR 15606.

6.3.3 Tipo de fluxo

A presença de um carrossel de objetos em um serviço deve ser indicada na tabela de mapa de programa desse serviço, colocando o tipo de *stream* que contém o carrossel de dados no valor de 0x0B (ver ISO/IEC 13818-1) ou em um valor definido pelo usuário.

7 Encapsulamento multiprotocolo (MPE)

7.1 Especificação de transporte de dados

Os datagramas são encapsulados nas *datagram_sections* que são compatíveis com o formato *DSMCC_section* para dados privados (ver ISO/IEC 13818-6). O mapeamento da seção dentro dos pacotes MPEG-2 de *stream* de transporte é definido em sistemas MPEG-2 (ver ISO/IEC 13818-1).

A sintaxe e a semântica do *datagram_section* são definidas na Tabela 23.

Tabela 23 – Sintaxe do *datagram_section*

Sintaxe	Número de bits	Mnemônico
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*

Tabela 23 (continuação)

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A semântica do *datagram_section* deve ser a seguinte:

— ***table_id***: campo de 8 bits que deve ser configurado em 0x3E, seções DSM-CC com dados privados (ver ISO/IEC 13818-6:1998, Seção 5);

— ***section_syntax_indicator***: campo que deve ser configurado conforme definido na ISO/IEC 13818-6:1998, Seção 5;

— ***private_indicator***: campo que deve ser configurado conforme definido na ISO/IEC 13818-6:1998, Seção 5;

— ***reserved***: campo de 2 bits que deve ser configurado em "11";

— ***section_length***: campo que deve ser configurado conforme definido na ISO/IEC 13818-6:1998, Seção 5;

— ***MAC_address_[1..6]***: campo de 48 bits que contém o endereço MAC do destino. O endereço MAC é fragmentado em seis campos de 8 bits, rotulados como *MAC_address_1* a *MAC_address_6.* O campo *MAC_address_1* contém o byte mais significativo do endereço MAC, enquanto o *MAC_address_6* contém o byte menos significativo. A Figura 3 ilustra o mapeamento dos bytes do endereço MAC nos campos da seção. A ordem dos bits nos bytes não está reservada e o MSB (Bit Mais Significativo) de cada byte também é transmitido primeiro. Os campos *MAC_address* contêm um endereço MAC claro ou embaralhado, como indicado pelo campo *address_scrambling_control*;

— ***payload_scrambling_control***: campo de 2 bits que define o modo de embaralhamento do *payload* da seção. Isso inclui o começo do *payload* depois do *MAC_address_1*, mas exclui a *checksum* ou campo CRC32 (ver Tabela 24). O método de embaralhamento aplicado é privativo ao usuário;

— ***address_scrambling_control***: campo de 2 bits que define o modo de dispersão do endereço MAC nesta subseção (ver Tabela 25). Este campo permite uma mudança dinâmica dos endereços MAC. O método de embaralhamento aplicado é privativo ao usuário;

— ***LLC_SNAP_flag***: *flag* de 1 bit. Se o *flag* estiver configurado em "1", a *payload* carrega um datagrama seguindo o campo *MAC_address_1*. A estrutura LLC/SNAP deve indicar o tipo de datagrama transportado. Se o *flag* estiver configurado em "0", a seção deve conter um datagrama IP sem encapsulamento LLC/SNAPP;

— ***current_next_indicator***: campo de 1 bit que deve ser configurado no valor de "1";

— ***section_number***: campo de 8 bits. Se o datagrama for carregado em seções múltiplas, então este campo indica a posição da seção dentro do processo de fragmentação. Do contrário, este campo deve ser configurado em zero;

— ***last_section_number***: campo de 8 bits que deve indicar o número da última seção usada para carregar o datagrama, ou seja, o número da última seção do processo de fragmentação;

— **LLC_SNAP**: estrutura que deve conter o datagrama de acordo com as especificações da ISO/IEC 8802-2 LLC (*Logic Link Control*) e da ISO/IEC TR 8802-1 SNAP (*Subnetwork Acces Control*). Se a *payload* da seção estiver embaralhada (ver *payload_scrambling_mode*), estes bytes devem estar dispersos;

— ***IP_datagram_data_byte***: bytes que contêm os dados do datagrama. Se a *payload* da seção estiver embaralhada (ver *payload_scrambling_mode*), estes bytes devem estar embaralhados;

— ***stuffing_byte***: campo opcional de 8 bits cujo valor não é especificado. Se a *payload* da seção for embaralhada (ver *payload_scrambling_mode*), este campo é embaralhado. Eles devem auxiliar a codificação do bloco e o processamento de dados nos ambientes de *wide bus*. O número de *stuffing_bytes* usados deve adequar-se às exigências de alinhamento dos dados definidos no *data_broadcast_descriptor*;

— ***checksum***: campo que deve ser configurado conforme definido na ISO/IEC 13818-6:1998, Seção 5. Ele é calculado sobre o *datagram_section* completo;

— **CRC_32**: campo que deve ser configurado conforme definido na ISO/IEC 13818-6:1998, Seção 5. Ele é calculado sobre o *datagram_section* completo.

Tabela 24 – Codificação do campo *payload_scrambling_control*

Valor	Controle de embaralhamento do *payload*
00	Não embaralhado
01	Definido pelo serviço
10	Definido pelo serviço
11	Definido pelo serviço

Tabela 25 – Codificação do campo *address_scrambling_control*

Valor	Controle de endereço de embaralhamento
00	Não embaralhado
01	Definido pelo serviço
10	Definido pelo serviço
11	Definido pelo serviço

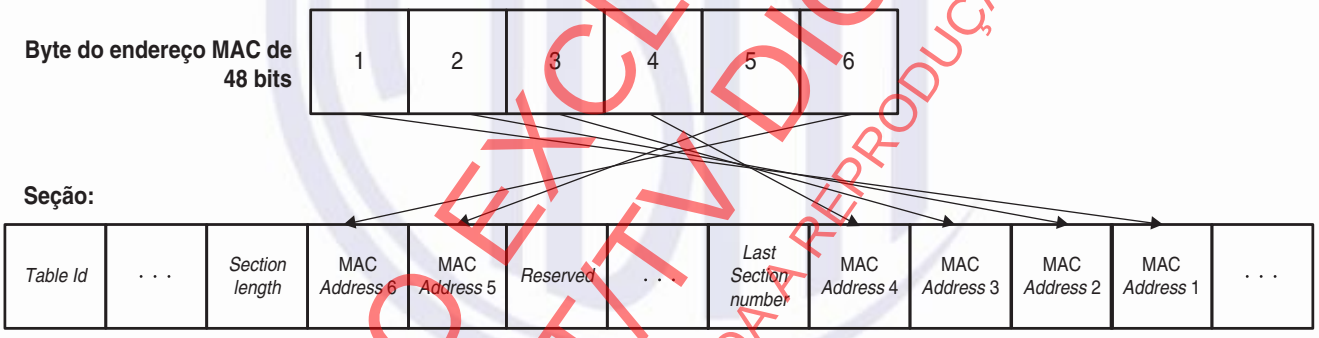


Figura 3 – Mapeamento dos bytes do endereço MAC para os campos da seção

7.2 Especificações PSI e SI

O serviço de transmissão de dados deve indicar a transmissão de datagramas pela inclusão de um ou mais descritores de transmissão de dados em SI (ver ARIB STD-B23). Cada descritor deve ser associado a um *stream* via um identificador *component_tag*. Em particular, o valor do campo *component_tag* deve ser idêntico ao valor do campo *component_tag* de um *stream_identifier_descriptor* (ver ETSI EN 300 468:2010, 6.2.39) que pode estar presente na tabela de mapa de programa PSI (PMT) para o *stream* usado para transmitir os datagramas.

7.3 Descritor de protocolo de transporte

O descritor de protocolo de transporte é utilizado da seguinte maneira:

— ***protocol_id***: campo que deve ser configurado em 0x0002 para indicar o uso da especificação de encapsulamento multiprotocolo;

— ***component_tag***: campo que deve ter o mesmo valor de um campo *component_tag* de um *stream_identifier_descriptor*, que possa estar presente na seção de mapa de programa PSI para o *stream* no qual os dados são transmitidos;

— ***selector_byte***: bytes seletores que devem transportar a estrutura *multiprotocol_encapsulation_info* que é definida na Tabela 26.

Tabela 26 – Sintaxe para a estrutura *multiprotocol_encapsulation_info*

Sintaxe	Número de bits	Mnemônico
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*
}		

A semântica da estrutura *multiprotocol_encapsulation_info* deve ser a seguinte:

— ***MAC_address_range***: campo de 3 bits que deve indicar o número de bytes do endereço MAC que o serviço usa para diferenciar os receptores de acordo com a Tabela 27;

— ***MAC_IP_mapping_flag***: *flag* de 1 bit. O serviço deve configurar esse *flag* em “1”, se usar o IP para mapeamento MAC (ver RFC 1112). Se esse *flag* estiver configurado em “0”, o mapeamento dos endereços IP para endereços MAC é feito fora do escopo desta Norma;

— ***alignment_indicator***: campo de 1 bit que deve indicar o alinhamento que existe entre os bytes do *datagram_section* e os bytes do *stream* de transporte, de acordo com a Tabela 28;

— ***reserved***: campo de 3 bits que deve ser configurado em “111”;

— ***max_sections_per_datagram***: campo de 8 bits que deve indicar o número máximo de seções que pode ser usado para carregar uma única unidade de datagrama.

Tabela 27 – Codificação do 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

Tabela 28 – Codificação do campo *alignment_indicator*

Valor	Alinhamento em bits
0	8 (padrão)
1	32

7.4 Tipo de *stream*

A presença de um *stream* de dados de multiprotocolo em um serviço deve ser indicada na seção de mapa de programa desse serviço pela configuração do tipo de *stream* para o valor de 0x0A (ver ISO/IEC 13818-6:1998, Seção 5) ou um valor definido pelo usuário.

8 Especificação da transmissão do *data piping*

8.1 Especificação do transporte de dados

O serviço de transmissão de dados deve inserir os dados a serem transmitidos diretamente na *payload* dos pacotes MPEG-2 TS.

O serviço de transmissão de dados pode usar o campo *payload_unit_start_indicator* e o campo *transport_priority* dos pacotes do *stream* de Transporte MPEG-2 na forma de serviço privado. O uso do *adaptation_field* deve ser compatível com MPEG-2.

A entrega dos bits em tempo através de um *data pipe* é um serviço privado e não é especificada nesta parte da ABNT NBR 15606.

8.2 Especificações PSI e SI

O serviço de transmissão de dados deve indicar o uso de um *data pipe* (canal de dados), incluindo um ou mais descritores de transmissão de dados em SI (ver ETSI EN 300 468). Cada descritor deve ser associado a um canal de dados particular via um *identificador component_tag*.

Em particular, o valor do campo *component_tag* deve ser idêntico ao valor do campo *component_tag* de um *stream_identifier_descriptor* (ver ETSI EN 300 468) que pode ser apresentado na seção de mapa de programa PSI para o *stream* que é usado como um *data pipe*.

8.3 Descritor de protocolo de transporte

O descritor de transmissão de dados deve ser usado da seguinte forma:

— ***protocol_id***: campo que deve ser configurado em 0x0005 para indicar um canal de dados Ginga. Os outros campos estão presentes.

8.4 Tipo de *stream*

A especificação do *stream_type* na seção de mapa de programa deve ser 0x7E (ver ABNT NBR 15603-2:2007, Tabela J.1).

9 Especificação de transmissão de PES independente

9.1 Transmissão de PES independente

A especificação de transmissão de PES independente é um método utilizado para implementar o *streaming* para serviços de transmissão de dados. Há dois tipos de especificação de transmissão PES: sincronizada e assíncrona.

O sistema de transmissão de PES sincronizada é utilizado quando é necessário sincronizar dados em um *stream* com outros *streams*, incluindo vídeo e áudio. A especificação de transmissão PES assíncrona é utilizada quando a sincronização não é necessária. Como um exemplo de aplicação importante, espera-se que o tipo sincronizado seja utilizado para transmitir *closed caption*, e o tipo assíncrono para transmissão de caracteres sobrepostos (*superimposed*). Para informações relacionadas ao PES independente, ver Anexo A.

9.2 PES sincronizada

De acordo com a especificação de transmissão PES sincronizada, os dados são transmitidos utilizando um pacote PES especificado na ISO/IEC 13818-1. Qualquer mapeamento de pacote PES para um *stream* de transporte MPEG-2 deve atender à ISO/IEC 13818-1.

De acordo com a especificação de transmissão do tipo sincronizada, um pacote PES com as seguintes restrições é utilizado além da sintaxe e semântica especificadas na ISO/IEC 13818-1.

Para o cabeçalho do pacote PES correspondente ao private_*stream_1*, deve ser utilizado o seguinte:

— ***stream_id***: no caso de um *stream* do tipo sincronizado, este deve ser configurado em '0xBD' (private_*stream_1*);

— ***PES_packet_length***: campo de 16 bits que deve ter um valor que não seja zero.

A estrutura de dados da PES sincronizada demonstrada na Tabela 29 deve ser inserida no campo *PES_packet_data_bytes*.

Tabela 29 – Estrutura de dados da PES sincronizada

Sintaxe	Número de bits	Mnemônico
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*
}		

Tabela 29 (continuação)

Sintaxe	Número de bits	Mnemônico
for(i=0;i<N2;i++){		
syncronized_PES_data_byte	8	*bslbf*
}		
}		

A semântica de campos em um pacote PES sincronizado é:

— ***data_identifier***: campo de 8 bits que deve ser configurado em '0x80';

— ***private_stream_id***: não utilizado (0xFF);

— ***PES_data_packet_header_length***: campo de 4 bits que indica a extensão em bytes do *PES_data_private_date_bytes*;

— ***PES_data_private_data_byte***: campo de 8 bits que fornece uma descrição mais detalhada dos dados contidos na PES sincronizada e depende do serviço. Uma unidade receptora pode omitir este campo;

— ***synchronized_PES_data_byte***: campo de 8 bits contendo os dados transmitidos.

9.3 PES assíncrono

De acordo com a especificação de PES assíncrono, os dados são transmitidos utilizando um pacote PES especificado na ISO/IEC 13818-1. Qualquer mapeamento de pacote PES para um *stream* de transporte MPEG-2 deve atender à ISO/IEC 13818-1.

De acordo com a especificação de transmissão assíncrona, um pacote PES com as seguintes restrições é utilizado, além da sintaxe e semântica especificadas na ISO/IEC 13818-1.

Para o cabeçalho do pacote PES correspondente ao *private_stream_2*, deve ser utilizado:

— ***stream_id***: em caso de um *stream* do tipo assíncrono, deve ser configurado a '0xBF' (*private_stream_2*);

— ***PES_packet_length***: campo de 16 bits que deve ter um valor que não seja zero.

A estrutura de dados da PES assíncrona demonstrada na Tabela 30 é inserida no campo do *PES_packet_data_bytes*.

Tabela 30 – Estrutura de dados de PES assíncrona

Sintaxe	Número de bits	Mnemônico
Asynchronous_PES_data() {		
data_identifier	8	*uimsbf*
private_stream_id	8	*uimsbf*

Tabela 30 (continuação)

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A semântica de campos em um pacote PES assíncrono é:

— ***data_identifier***: campo de 8 bits que deve ser configurado em '0x81';

— ***private_stream_id***: não utilizado (0xFF);

— ***PES_data_packet_header_length***: campo de 4 bits que indica a extensão em bytes do *PES_data_private_date_bytes*;

— ***PES_data_private_data_byte***: campo de 8 bits que é uma utilização mais detalhada desta área e depende de um serviço. Uma unidade receptora pode omitir este campo;

— ***asynchronous_PES_data_byte***: campo de 8 bits contendo os dados transmitidos.

10 Protocolos de transporte

10.1 Protocolo do canal de transmissão

10.1.1 *Stream* de transporte MPEG-2

O *stream* de transporte MPEG-2 deve estar de acordo com a ISO/IEC 13818-1 e ABNT NBR 15602-3.

10.1.2 Seção MPEG-2

A seção MPEG-2 deve estar de acordo com a ISO/IEC 13818-1 e ABNT NBR 15602-3.

10.1.3 Dados privados DSM-CC

Os dados privados DSM-CC devem estar de acordo com a ISO/IEC 13818-6.

10.1.4 Carrossel de dados DSM-CC

O carrossel de dados DSM-CC deve estar de acordo com a ISO/IEC 13818-6.

10.1.5 Carrossel de objetos DSM-CC

Para o protocolo de transporte de transmissão que transmite conteúdos de aplicação Ginga, dois sistemas são especificados: o sistema de transmissão de carrossel de objetos e o de carrossel de dados.

Cada sistema está de acordo com a ISO/IEC 13818-6 (para equivalências funcionais, ver Tabela 31).

Tabela 31 – Equivalentes funcionais

Nome	GEM	Implementação na ARIB STD-B23	Observações
Carrossel	Ver GEM 1.3:2011, 6.2.5 e 11.7.2	Ver ARIB-STD-B23:2004, Anexo B	Em caso de utilização de carrossel de dados, aplica-se a ARIB STD-B23:2004, Anexo B Ver ARIB STD-B24
		transport_stream_id, original_network_id, service_id de *dvb_service_location()* na ARIB STD-B23:2004, Tabela B.26. DVB endereço carrossel NSAP deve seguir a semântica da ARIB-SI Qualquer sistema-padrão de carrossel é selecionável	
		ETSI TS 101 812:2006, Anexo B, e ISO/IEC 13818-6 (DSM-CC carrossel de objetos)	Em caso de utilização de um carrossel de objetos, a ETSI TS 101 812:2006, Anexo B, se aplica

10.1.6 Protocolo IP de transporte de *multicast* em um canal de transmissão

O protocolo IP de transporte de *multicast* em um canal de transmissão deve estar de acordo com a ETSI EN 301 192.

10.1.7 Protocolo IP

O protocolo IP deve estar de acordo com a RFC 791.

10.1.8 Protocolo UDP

O protocolo UDP deve estar de acordo com a RFC 768.

10.1.9 Informações de serviço

As informações de serviço devem estar de acordo com as ABNT NBR 15603-1, ABNT NBR 15603-2 e ABNT NBR 15603-3.

10.1.10 Sinalização de IP

A sinalização de IP deve estar de acordo com a ETSI EN 301 192.

10.2 Protocolos de canal de interatividade

10.2.1 Pilha de protocolo do canal interatividade

Os protocolos de canal de interatividade devem estar de acordo com a ABNT NBR 15607-1.

A Figura 4 ilustra o conjunto de protocolos de canal de interatividade SBTVD que são acessíveis por aplicativos Ginga em alguns ou todos os perfis (ver ABNT NBR 15606-1). Os detalhes completos das API que oferecem acesso a estes protocolos de interação estão na ETSI TS 101 812:2006, Seção 11.

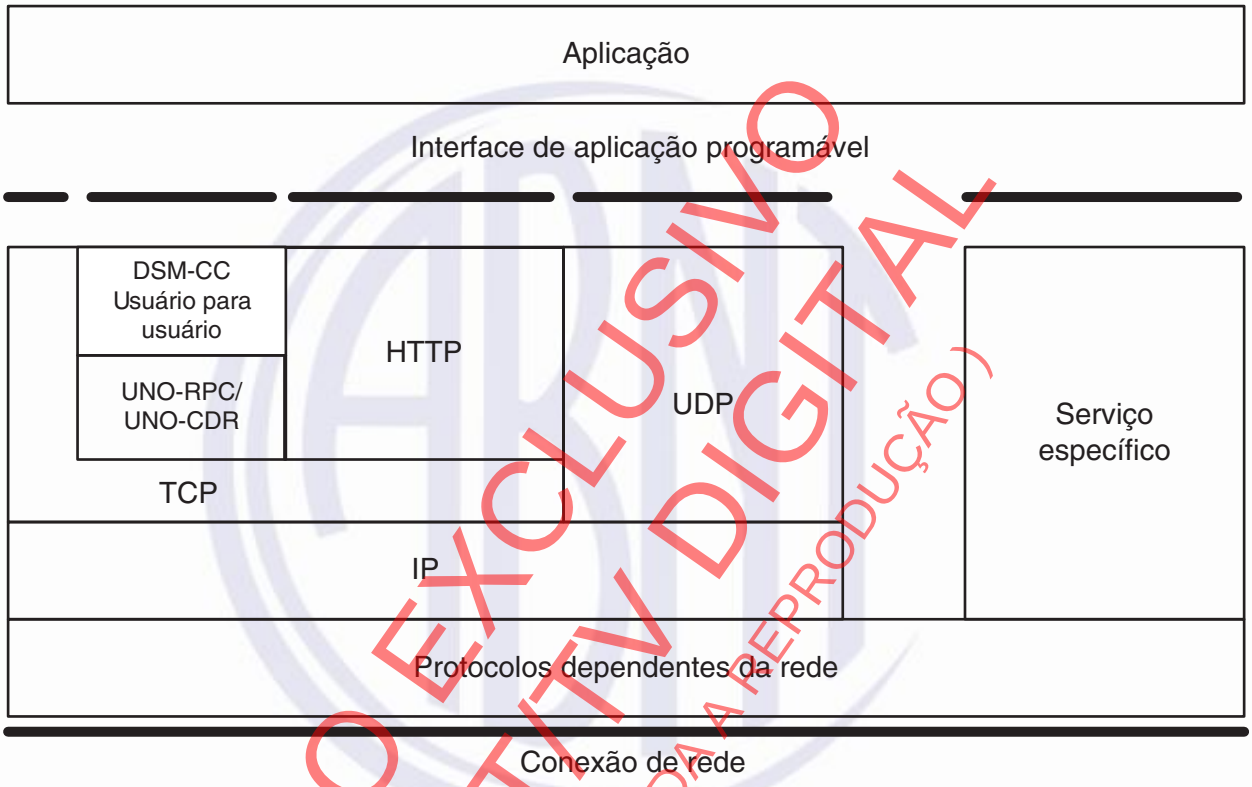


Figura 4 – Pilha de protocolo do canal de interatividade

10.2.2 Protocolo dependente da rede

Os protocolos dependentes da rede devem estar de acordo com as ETS 300 800, ETS 300 801, EN 301 193, EN 301 195, EN 301 199, TR 101 201, EN 301 790, respectivamente para CATV, PSTN/ISDN, DECT, GSM, LMDS (*Local Multipoint Distribution Service*), SMATV e redes de satélite.

Para conexões baseadas nos canal de interatividade, o protocolo PPP é utilizado conforme definido nas RFC 1332, RFC 1661 e RFC 1717. Redes que utilizam endereçamento a servidores DNS devem estar de acordo com a RFC 1877.

10.2.3 Protocolo de *internet* (IP)

O protocolo de internet está definido na RFC 791.

10.2.4 Protocolo de controle de transmissão (TCP)

O protocolo de controle de transmissão está definido na RFC 793.

10.2.5 UNO-RPC

O UNO-RPC consiste em *Internet Inter-ORB Protocol* (IIOP) e deve estar de acordo com a CORBA/IIOP.

10.2.6 UNO-CDR

O UNO-CDR deve estar de acordo com a CORBA/IIOP.

10.2.7 DSM-CC usuário para usuário

O DSM-CC usuário para usuário deve estar de acordo com a ISO/IEC 13818-6, com restrições e extensões definidas na ETSI EN 301 192 e ETSI TR 101 202.

10.2.8 Protocolo HTTP

O protocolo HTTP deve estar de acordo com a RFC 2616 para HTTP 1.1, RFC 1945 para HTTP 1.0 e RFC 2818 para HTTPS.

10.2.9 Protocolo específico para o serviço

O protocolo específico ou proprietário deve ser suportado através de provedores registrados para este serviço.

10.2.10 Protocolo de datagrama do usuário (UDP)

O protocolo de datagrama do usuário (UDP) deve estar de acordo com a RFC 768.

10.2.11 DNS

Os terminais que implementam o DNS devem estar de acordo com as RFC 1034, RFC 1035, RFC 1982 e RFC 2181.

10.3 Protocolos de transporte para aplicativos sendo carregados no canal de interação

10.3.1 Considerações gerais

No caso híbrido entre o fluxo de transmissão e o canal de interação, todas as informações dos diretórios são fornecidas no fluxo de transmissão, sendo que alguns ou todos os conteúdos dos arquivos podem ser fornecidos através do canal de interação.

10.3.2 Entrega do arquivo pelo canal de transmissão

Em terminais que usam o DSM-CC carrossel de objetos de usuário para usuário, o conteúdo do arquivo é transportado através de uma *BIOP::File*, como é o caso normal para um carrossel de objetos DSM-CC usuário para usuário. O IOR do arquivo de ligação para o arquivo de conteúdo usa um *BIOPProfileBody* ou um *LiteOptionsProfileBody*.

10.3.3 Entrega do arquivo pelo canal de interação

Em terminais que usam o DSM-CC carrossel de objetos de usuário para usuário, o IOR do arquivo de ligação para o arquivo de conteúdo usa o *HTTPProfileBody* Esta forma de IOR deve ser usada apenas para objetos *BIOP::File*.

10.3.4 *HTTPProfileBody*

O *HTTPProfileBody* especifica *host*, *port* e também *path_segments*, que em solicitações HTTP são concatenados para formar uma URL da forma: *http://host:port/path_segments*. A sintaxe desse perfil é apresentada na Tabela 32.

Tabela 32 – Sintaxe do *HTTPProfileBody*

Sintaxe	Número de bits	Mnemônico	Valor	Comentários
HTTProfileBody {				
profileId_tag	32	*uimsbf*	0x53425444	
profile_data_length	32	*uimsbf*	*	
profile_data_byte_order	8	*uimsbf*	0x00	*Big endian byte order*
version.major	8	*uimsbf*	0x01	*Protocol major version 1*
version.minor	8	*uimsbf*	0x00	*Protocol minor version 0*
host_data_length	8	*uimsbf*	N1	
for (k=0;k<N1;k++) {				
host_data	8	*uimsbf*	+	
}				
port	16	*uimsbf*		
objectKey_length	16	*uimsbf*	N2	
for (k=0;k<N2;k++) {				
objectKey_data	8	*uimsbf*	+	
}				

A semântica do *HTTPProfileBody* deve ser a seguinte:

— ***tag_value***: estes campos de 8 bits estabelecem a indicação para SBTVD (Sistema Brasileiro de Televisão Digital). Onde, em ASCII, 0x53 é “S”, 0x42 é “B”, 0x54 é “T” e 0x44 é “D”;

— ***version***: estes campos de 8 bits indicam a versão do protocolo que o servidor usa para entregar o arquivo especificado. O valor da versão 1.0 indica que o protocolo de transporte é o definido como perfil de HTTP 1.0”;

— ***host_data***: estes bytes transmitem o identificador do *host* da *internet* para que as mensagens sejam enviadas. Pode ser o nome de domínio qualificado ou Internet-padrão na forma “decimal pontilhada”;

EXEMPLO “192.231.79.52”

— ***port***: estes 16 bits inteiros representam o número da porta TCP/IP no host especificado onde os agentes-alvos ficam escutando os pedidos;

— ***objectKey_data***: estes bytes em forma de *strings* levam a parte *path_segments* da URL que exclusivamente identifica o objeto no servidor de acordo com a RFC 2396.

11 Modelo de aplicação

11.1 Aplicação Ginga-J

A aplicação Ginga-J deve estar de acordo com a ABNT NBR 15606-4. Nesta parte da ABNT NBR 15606, este modelo é utilizado como modelo Ginga-J; para o modelo Ginga-NCL, a aplicação Ginga-J deve estar de acordo com a ABNT NBR 15606-2.

11.2 Modelo Ginga-J

O modelo é utilizado como o modelo Ginga-J e deve estar de acordo com a ABNT NBR 15606-4:2010, Seção 7.

11.3 Como lidar com o modelo NCL

O modelo NCL deve estar de acordo com os detalhes da ABNT NBR 15606-2.

11.4 Gerenciamento de recursos entre aplicações

O gerenciamento de recursos entre aplicações deve estar de acordo com a ABNT NBR 15606-2.

12 Transmissão de informações de aplicação

12.1 Descritores AIT e valores constantes

Os descritores AIT e os valores constantes devem estar de acordo com a ARIB STD-B23 e com a Tabela 33.

Tabela 33 – Descritores AIT e valores constantes

Onde é utilizado	Tipo	Valor	Onde é definido	Foco	Descritores
Data contents descriptor	*Descriptor tag*	0xC7	EIT	Ver Seção 12, 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 (carrossel de dados e objetos)	Ver GEM - *Middleware*
Caching priority descriptor		0x71			
Content type descriptor		0x72	*BIOP objectinfo*		
Reservado para futuro descritor OC		0x73-0x7F	OC		
Application information table (AIT)	*Table ID* no PID AIT	0x74		Ver Seção 12	

Tabela 33 (continuação)

Onde é utilizado	Tipo	Valor	Onde é definido	Foco	Descritores
Application descriptor	*Descriptor tag*	0x00	AIT	Ver Seção 12 e Anexo C	Ver GEM - *Middleware*
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 uso futuro)		0x08-0x0A			
Application icons descriptor		0x0B			
Pre-fetch descriptor		0x0C			
DII location descriptor		0x0D			
Reservado para uso futuro		0x0E - 0x010			
IP signalling descriptor		0x11			
Reservado para uso futuro		0x12-0x5E			
Private data specifier descriptor		0x5F			
Reservado para uso futuro		0x60-07F			
Definido pelo usuário		0x80-0xFE			

Tabela 33 (continuação)

Onde é utilizado	Tipo	Valor	Onde é definido	Foco	Descritores
Sistema de codificação Ginga	Sistema de codificação de dados (*data_component_id*)	A0 (FullSeg) A1(OneSeg)	PMT	Ver 12.6, e ABNT NBR 15608-3	Ver ARIB STD-B23
Sistema de transmissão AIT		A3 (FullSeg) A4 (OneSeg)			
Sistema de transmissão de carrossel de dados	Formato de transmissão (*transmission_format*)	1	Área de sistema de codificação de dados	Ver Seção 12 e ARIB STD-B23	
Sistema de transmissão de carrossel de objetos		10			
Carrossel de objetos Ginga	Identificação de protocolo (*protocol_id*)	0x0001	AIT	Ver Seção 12 e ARIB STD-B23	
Carrossel de dados Ginga		0x0004			
Ginga-J *application type*	Tipo de aplicação (*application_type*)	0x0001	AIT	Ver Seção 12 e ARIB STD-B23	

12.2 Execução da aplicação Ginga

Para realizar a execução da aplicação Ginga é necessário especificar a aplicação e transmitir as informações adicionais da aplicação para controlá-la.

O sistema de transmissão da informação da aplicação a ser utilizado nesta parte da ABNT NBR 15606 deve estar de acordo com a Seção 12.

As informações adicionais de acordo com a ARIB STD-B23 são as seguintes:

— valores de identificação referentes ao Ginga e à AIT, para identificar o armazenamento do componente de dados do *additional_ ginga_info()* a partir do *additional identifying information* no *data component descriptor* para o ES que transmite a aplicação Ginga na PMT;

— armazenamento do *ginga_info()* a partir do *additional information* dentro do data *contents descriptor*, para ser armazenado com a área dos descritores de um evento de programa que utilize o aplicações Ginga na EIT;

— armazenamento da *ait_identifier_info()* a partir do *additional information* dentro do *data component descriptor* para o ES que transmite a AIT na PMT;

— nas estruturas *additional_ginga_info()* e *ginga_info()*, o campo *transmission_format* deve ser identificado com o valor '10', que indica o formato de transmissão como carrossel de objetos.

Além das informações acima, a *Application Information Table* (Tabela de Informações da Aplicação), especificada em 12.16.1, deve ser transmitida em um ES que engloba o programa na forma de seções privadas. Utilizando a AIT, a informação da aplicação é armazenada (ver 12.17.1) sobre a estrutura dos grupos de descritores armazenados na AIT. Para informações adicionais relacionadas às tabelas PMT e EIT, ver o Anexo C.

12.3 Sinalizações comuns das aplicações

As sinalizações comuns das aplicações são os requisitos mínimos de sinalização capazes de identificar alguns aspectos como a fonte do código da aplicação, o nome da aplicação, o ID da aplicação e ID da organização referentes à aplicação.

As informações de aplicações são transmitidas em seções privadas, a partir da Tabela 46 (ver 12.16.1), como um ES que engloba o programa. Desta forma, a informação adicional necessária para cada aplicação é transmitida.

Como complemento, os valores de identificação do componente de dados são determinados de forma a indicar a existência da transmissão da AIT, assim como da aplicação Ginga, e a estrutura da seleção de área do *data component descriptor* (ver 12.7 para detalhes).

Os seguintes descritores devem ser armazenados na AIT como informação comum, sem levar em consideração o formato da aplicação:

— ***transport protocol descriptor***: todas as aplicações devem estar no escopo de pelo menos um *transport protocol descriptor*. Este descritor pode ser armazenado tanto no *common information descriptor loop* como no *application information descriptor loop*;

— ***application descriptor***: um único descritor de aplicação deve ser armazenado em um enlace de descritor de informação de aplicação para cada uma das aplicações;

— ***application name descriptor***: uma única aplicação deve ser armazenada em um enlace de descritor de informação de aplicação para cada uma das aplicações.

12.4 Sinalizações adicionais das aplicações Ginga

Os parâmetros de sinalização da aplicação e a indicação do *initial entity* são requisitos mínimos que devem ser transmitidos via AIT.

No enlace do descritor de informações de aplicação AIT Ginga, os seguintes descritores devem ser armazenados pelo menos para uma aplicação.

— Ginga-J *application descriptor*;

— Ginga-J *application location descriptor*;

ou

— Ginga-NCL *application descriptor*;

— Ginga-NCL *application location descriptor*.

12.5 Informações adicionais em PSI/SI

Como para as informações de aplicação, a Tabela 33 de informações de aplicação (AIT) é transmitida em modo seção privada como um ES que compreende o programa. As informações adicionais referentes à transmissão de informações de aplicação são as seguintes:

— definição de valores de identificação correspondentes a Ginga e AIT para identificar o armazenamento do componente de dados de *additional_ginga_info()* de informações adicionais, identificando as informações do descritor de componente de dados para ES que transmite Ginga na PMT;

— armazenamento de *ginga_ info()* em informação adicional dentro do descritor de conteúdos de dados a serem armazenados na área do descritor de um evento de programa que utiliza a aplicação Ginga na AIT;

— armazenamento de *ait_identifier_info()* em informação adicional dentro do descritor de componente de dados para ES que transmite AIT de PMT;

— diferenciação de outros sistemas de transmissão atribuindo 0x0004 como o *protocol_id* que corresponde aos dados de transmissão de carrossel. Para detalhes do *selector_byte*, ver 12.17.6;

— atribuição de um sistema de transmissão de carrossel de objetos em '10' no campo de *additional_ginga_info()* e *ginga_info()* para identificar o sistema de transmissão de conteúdos em nível de PMT;

— no caso de *transmission_format*='10' (sistema de transmissão de carrossel de objetos), o descritor *association_tag* (valor de *tag*: 0x14), o descritor *deffered_Association_tag* (valor de *tag*: 0x15) ou o descritor *Carousel_id* (valor: 0x13), especificados na ISO/IEC 13818-6, devem ser armazenados em PMT, conforme necessário.

12.6 Identificação do componente de dados

Um *data_component_id* é atribuído à aplicação Ginga. Enquanto isso, um valor de identificação de componente de dados é atribuído à transmissão AIT e um *stream* elementar a ser transferido no modo de seção privada é adicionado ao PMT.

12.7 Descritor de componente de dados e descritor de conteúdos de dados

12.7.1 Referência indireta

A partir do carrossel que transmite a aplicação Ginga, deve ser feita referência indireta ao descritor de componente de dados relevante ao sistema de codificação Ginga pelo componente de *tag* (*component_tag*.)

12.7.2 Descritor de componente de dados em aplicação Ginga – Sistema de codificação de dados

Quando a identificação da codificação de dados é feita pelo sistema de codificação Ginga, a estrutura *additional_ginga_info()*, conforme demonstrado na Tabela 34, é descrita dentro da área de informações adicionais de identificação no descritor de componente de dados.

Tabela 34 – *Additional_ginga_info()*

Sintaxe	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*
}		
}		

A descrição do *additional_ginga_j_info( )* deve ser a seguinte:

— ***transmission_format*** (formato de transmissão): a área de 2 bits especifica o sistema de transmissão da aplicação Ginga (ver Tabela 35);

— ***application_identifier_flag*** (*flag* identificador da aplicação): *flag* de 1 bit que indica se o identificador de aplicação está incluído na área de seleção (ver Tabela 36);

— ***recommended_resolution*** (resolução recomendada): A resolução da aplicação Ginga (correspondente às características de resolução) e a taxa de aspecto (correspondente às características de *display-aspect-ratio*) são indicadas neste campo. No SBTVD o campo *recommended resolution* é especificado na tabela AIT sendo obrigatório seu uso nos dispositivos *one-seg* conforme Tabela 37;

— ***independent_flag*** (*flag* independente de disponibilidade de áudio e vídeo): indica se presume-se que o programa de transmissão de dados seja ouvido e visto independentemente; 0 = Impossível e 1 = Possível;

— ***application_identifier()*** (identificador de aplicação): um valor para identificar exclusivamente a aplicação. Ver Tabela 39 para os detalhes.

Escolher um dos valores da Tabela 38.

Tabela 35 – Formato de transmissão

Valor	Descrição
00	Carrossel de dados e mensagem de eventos (exceto serviço de dados só para armazenamento)
00	Carrossel de dados (serviço de dados só para armazenamento)
10	Carrossel de objetos
11	Reservado para o futuro

Tabela 36 – *Flag* identificador da aplicação

default_version_flag	Descrição
0	Não utilizar o valor-padrão para o número da versão
1	Utilizar o valor-padrão para o número da versão

Tabela 37 – Sintaxe do campo *recommended resolution*

Sintaxe	Número de bits	Mnemônico
recommended_resolution	4	*bslbf*

Tabela 38 – Opções de resolução

Valor	Descrição
0000	Aplicações Ginga com múltiplos tamanhos e resoluções
0001	1920 x 1080 (16:9)
0010	1280 x 720 (16:9)

Tabela 38 (continuação)

Valor	Descrição
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* orientação "retrato")
1100	n (min = 320) x 240 (*one-seg* orientação "paisagem")
1101	480 x n (min = 640) (*one-seg* orientação "retrato") – VGA
1110	n (min = 640) x 480 (*one-seg* orientação "paisagem") – VGA
1111	Reservado para o futuro

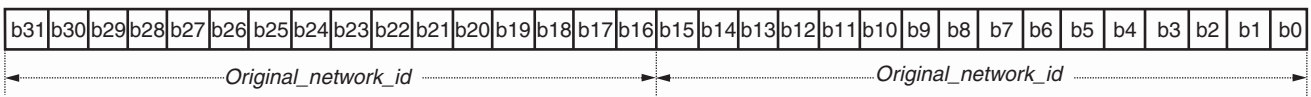
As resoluções VGA retrato ou paisagem são opcionais em dispositivos *one-seg* e são destinadas aos dispositivos de tamanho maior. Quando uma aplicação *one-seg* for destinada a estes dispositivos, e utilizar o conteúdo de mídia nesta resolução, deve ser sinalizada a resolução VGA retrato ou paisagem. É permitido aos dispositivos de tamanho menor receber esta aplicação desde que seja implementado o *scaling*.

Tabela 39 – Codificação de identificador de aplicação

Estrutura de dados	Número de bits	*String* de bits
application_identifier() {		
organization_id	32	*bslbf*
application_id	16	*bslbf*
}		

— ***organization_id*** (ID da organização): campo de 32 bits que indica a organização que preparou a aplicação. Este ID armazena um número único de 32 bits atribuído no SBTVD pela emissora matriz e cujo valor é representado pelos 16 bits do *network_id* a partir do bit mais significativo, completando os 16 bits restantes com o mesmo valor;

EXEMPLO



Uma emissora matriz que possui o *Original_network_id* igual a (0000000010000001)B utiliza o *organization_id* igual a (00000000100000010000000010000001)B, identificação única para todas as emissoras de sua rede.

— ***application_id*** (ID da aplicação): campo de 16 bits que armazena o número exclusivamente atribuído no sistema para identificar a aplicação. Se a aplicação descrita pelo descritor for um serviço adicional a um programa de televisão ou de rádio, ela é utilizada para especificar a aplicação que realmente se associa ao programa de televisão ou rádio;

— ***download_id*** (ID do *download*): campo de 32 bits que serve como rótulo que identifica o carrossel de forma única. Ele mostra o carrossel que deveria estar montado como configuração-padrão;

— ***ondemand_retrieval_flag*** (*flag* de disponibilidade de recepção de áudio e vídeo por demanda): área de 1 bit que indica, para a recepção da aplicação transmitida pelo referido *elementary stream*, se a aquisição da aplicação a partir do carrossel em cada caso da operação de audiência está prevista. A capacidade de recepção é regulada pela operação de cada entidade de mídia; 0 = Não disponível e 1 = Disponível;

— ***file_storable_flag*** (*flag* de arquivo armazenável): indica se o armazenamento do arquivo do programa de transmissão de dados correspondente é possível. Por exemplo, o armazenamento de arquivo é difícil se a informação for atualizada durante o programa. A capacidade de armazenamento é regulada pela operação de cada entidade de mídia. 0 = Arquivo não armazenável e 1 = Arquivo armazenável;

— ***event_section_flag*** (*flag* de evento da seção de transmissão): campo de 1 bit que indica se a mensagem de evento é distribuída por este componente. 0 = A mensagem de evento não é distribuída e 1 = A mensagem de evento é distribuída;

— ***carousel_id*** (ID do carrossel): campo de 32 bits que é o valor de identificação que especifica de forma exclusiva o carrossel de objetos. Este valor de identificação é especificado pelo descritor *carousel_id* (descritor do identificador do carrossel), que está armazenado no PMT.

12.7.3 Descritor de conteúdos dos dados na aplicação Ginga – Sistema de conteúdo de dados

Se a identificação da codificação de dados for feita conforme o sistema de codificação Ginga, a estrutura *ginga_info()* demonstrada na Tabela 40 deve ser descrita na área de seleção do descritor de conteúdos de dados no AIT Isso permite que a notificação avançada da aplicação Ginga seja agendada para uso pela unidade de evento de programa.

As informações a respeito da aplicação Ginga e sinais de controles são armazenados na AIT. Não se presume que a aplicação seja controlada pela unidade de evento de programa. Consequentemente, não há mecanismo na AIT que compreenda o cronograma pelo qual a aplicação Ginga será utilizada para cada unidade de programa (ver ABNT NBR 15603-2:2007, 8.3.28).

Tabela 40 – *Ginga_j_info()*

Estrutura de dados	Número de bits	Mnemônico
ginga_info(){		
transmission_format	2	*bslbf*
reserved_future_use	1	*bslbf*
recommended_resolution	4	*bslbf*

Tabela 40 (continuação)

Estrutura de dados	Número de bits	Mnemônico
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()		

Tabela 40 (continuação)

Estrutura de dados	Número de bits	Mnemônico
} else if (transmission_format == '10') {		
ginga_object_carousel_info()		
ondemand_retrieval_flag	1	*bslbf*
file_storable_flag	1	*bslbf*
reserved_future_use	6	*bslbf*

A descrição da *ginga_info( )* deve ser a seguinte:

— ***transmission_format*** (formato de transmissão): área de 2 bits que especifica o sistema de transmissão da aplicação Ginga (ver Tabela 41);

Tabela 41 – Formato de transmissão

Valor	Descrição
00	Carrossel de dados e mensagem de eventos (exceto serviço de dados só para armazenamento)
00	Carrossel de dados (serviço de dados só para armazenamento)
10	Carrossel de objetos
11	Reservado para o futuro

— ***recommended_resolution*** (resolução recomendada): A resolução da aplicação Ginga e *display-aspect-ratio* que são indicadas na Tabela 38. As resoluções VGA retrato ou paisagem são opcionais em dispositivos *one-seg* e são destinadas aos dispositivos de tamanho maior. Quando uma aplicação *one-seg* for destinada a estes dispositivos, e utilizar o conteúdo de mídia nesta resolução, deve ser sinalizada a resolução VGA retrato ou paisagem. É permitido aos dispositivos de tamanho menor receber esta aplicação desde que seja implementado o *scaling*;

— ***default_version_flag*** (*flag* de utilização da versão-padrão): *flag* de 1 bit que indica que o valor-padrão especificado pela operação é utilizado como um perfil para a execução da aplicação Ginga que deve ser transmitida pelo ES correspondente. Deve estar de acordo com a Tabela 42;

Tabela 42 – *default_version_flag*

default_version_flag	Descrição
0	Não utilizar o valor-padrão para o número da versão
1	Utilizar o valor-padrão para o número da versão

— ***independent_flag*** (*flag* independente de disponibilidade de áudio e vídeo): indica se presume-se que o programa de transmissão de dados seja ouvido e visto independentemente; 0 = Impossível e 1 = Possível;

— ***application_identifier_flag*** (*flag* identificador da aplicação): *flag* de 1 bit que indica se o identificador de aplicação está incluído na área de seleção; 0 = Não Incluído e 1 = Incluído;

— ***content_id_flag*** (*flag* de ID de conteúdos): *flag* de 1 bit que indica se a ID dos conteúdos e sua versão estão incluídas no descritor; 0 = Não Incluído e 1 = Incluído;

— ***associated_application_flag*** (*flag* de aplicação associada): *flag* de 1 bit que indica os conteúdos associados ao programa de televisão ou rádio, quando a aplicação descrita por este descritor for um serviço adicional de dados para o programa de televisão ou rádio. Para uma aplicação que não seja um serviço adicional, o valor deve sempre ser 0;

— ***update_flag*** (*flag* de atualização): indica se há distribuição diferencial para esta aplicação no futuro; 0 = Não há distribuição diferencial e 1 = Há distribuição diferencial;

— ***ISO_639_language_code*** (código de linguagem): código de linguagem usado para a aplicação Ginga;

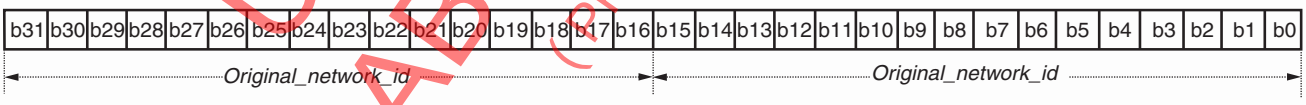
— ***application_identifier()*** (identificador de aplicação): valor para identificar exclusivamente a aplicação (ver Tabela 43);

Tabela 43 – Estrutura do identificador de aplicação

Estrutura de dados	Número de bits	*String* de bits
application_identifier() {		
organization_id	32	*bslbf*
application_id	16	*bslbf*
}		

— ***organization_id*** (ID da organização): campo de 32 bits que indica a organização que preparou a aplicação. Este ID armazena um número único de 32 bits atribuídos no SBTVD pela emissora matriz e seu valor é representado pelos 16 bits do *network_id* a partir do bit mais significativo, completando os 16 bits restantes com o mesmo valor;

EXEMPLO



Uma emissora matriz que possui o *Original_network_id* igual a (0000000010000001)B utiliza o *organization_id* igual a (00000000100000010000000010000001)B, identificação única para todas as emissoras de sua rede:

— ***application_id*** (ID da aplicação): campo de 16 bits que armazena o número que identifica a aplicação. O número é atribuído de forma exclusiva no sistema. Quando a aplicação descrita pelo descritor for um serviço adicional ao programa de televisão ou rádio, é utilizada para especificar a aplicação que de fato se associa ao programa de televisão ou rádio;

— ***content_id*** (ID de conteúdos): campo de 32 bits que é um rótulo que identifica o programa de transmissão de dados e é atribuído de forma exclusiva na companhia de transmissão. Em caso de armazenamento de dados, se o *content_id* tiver o mesmo valor de um programa de transmissão de dados anterior, os dados podem ser sobrescritos;

— ***content_version*** (versão dos conteúdos): campo de 16 bits que indica o número da versão entre o programa de transmissão de dados que tem uma ID de conteúdos idêntica;

— ***application_profiles_length*** (especificação de extensão do perfil da aplicação): indica a extensão do campo que especifica o perfil receptor com o qual a aplicação é executável;

— ***profile_major_version*** (número principal do perfil): campo de 8 bits que indica que o número principal entre os números de versão dos perfis de recepção deve corresponder pelo menos à execução de aplicação Ginga relevante;

— ***profile_minor_version*** (número secundário de perfil): campo de 8 bits que indica o número secundário entre os números de versão dos perfis de recepção que deve corresponder a pelo menos a execução de aplicação Ginga relevante;

— ***profile_micro_version*** (micronúmero de perfil): campo de 8 bits que indica o micronúmero entre os números de versão dos perfis de recepção que devem corresponder a pelo menos à execução de aplicação Ginga relevante;

— ***ginga_carousel_info()***: estrutura de dados especificada na ARIB STD-B24:2007, Anexo C2;

— ***ondemand_retrieval_flag*** (*flag* de disponibilidade de recepção de áudio e vídeo por demanda): área de 1 bit que indica, para a recepção da aplicação transmitida pelo referido *elementary stream*, se a aquisição da aplicação a partir do carrossel em cada caso da operação de audiência está prevista. A capacidade de recepção é regulada pela operação de cada entidade de mídia; 0 = Não disponível e 1 = Disponível;

— ***file_storable_flag*** (*flag* de arquivo armazenável): indica se o armazenamento do arquivo do programa de transmissão de dados correspondente é possível. Por exemplo, o armazenamento de arquivo é difícil se a informação for atualizada durante o programa. A capacidade de armazenamento é regulada pela operação de cada entidade de mídia;

— ***ginga_stored_carousel_info()***: estrutura de dados especificada na ARIB STD-B24:2007, Anexo C2;

— ***ginga_object_carousel_info()***: estrutura de dados especificada em 12.7.4.2.

12.7.4 Descritor de componente de dados para transmissão AIT

12.7.4.1 *Ait identifier inf*

Quando a ID da codificação de dados é transmissão AIT, a estrutura *ait_identifier_info()* demonstrada na Tabela 44 deve ser descrita dentro da área de seleção do descritor de componente de dados no PMT (ver ABNT NBR 15603-2:2007, Subseção 7.2.3).

Tabela 44 – *Ait_identifier_info()*

Estrutura de dados	Número de bits	*String* de bits
ait_identifier_info(){		
for (i=0; i<N; i++) {		
application_type	16	*uimsbf*

Tabela 44 (continuação)

Estrutura de dados	Número de bits	*String* de bits
reserved_future_use	3	*bslbf*
AIT_version_number	5	*uimsbf*
}		
}		

A descrição da *ait_identifier_info( )* deve ser a seguinte:

— ***application_type*** (tipo de aplicação): indica o valor do tipo de aplicação a ser transmitido na AIT. O valor atribuído a este campo deve estar conforme a Tabela 46;

— ***AIT_version_number*** (número da versão AIT): versão atual *version_number* armazenada.

12.7.4.2 Área de seleção do descritor de conteúdos de dados na transmissão do carrossel de objetos

Quando as informações para o controle de recepção do carrossel de objetos estão dentro da área de seleção do descritor de conteúdo de dados, as informações demonstradas na Tabela 45 devem ser adicionadas na posição da área de seleção para cada esquema de codificação de dados.

Tabela 45 – Estrutura *Ginga_object_carousel_info()*

Estrutura de dados	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*
}		

Tabela 45 (continuação)

Estrutura de dados	Número de bits	*String* de bits
if (default_timeout_DSI_flag == '1') {		
timeout_value_DSI	32	*uimsbf*
}		
if (default_leak_rate_flag == '1') {		
leak_rate	22	*uimsbf*
reserved	2	*bslbf*
}		
}		
}		

A descrição da *ginga_object_carousel_info()* deve ser a seguinte:

— ***num_of_carousels*** (número de carrosséis): campo de 8 bits que indica o número de carrosséis de objetos incluídos em uma volta do enlace;

— ***association_tag*** (*tag* de associação): campo de 16 bits que especifica o componente de *stream* para o qual a mensagem DSI com o *ServiceGatewayInfo* do carrossel de objeto é armazenada, pelo *tag* de associação que é atribuído pelo descritor *association_tag* de PMT;

— ***event_section_flag***: com este componente, a distribuição de mensagem de evento é indicada;

— ***component_size_flag*** (*flag* de tamanho de componente): campo de 1 bit que indica se o tamanho do componente é codificado na estrutura de dados. Quando o valor do campo *component_size* ainda não está definido, não está codificado (0: não codificado; 1: codificado);

— ***default_transaction_id_flag***: campo de 1 bit que indica se a ID de transação está codificada na estrutura de dados. Quando a aquisição de DII para ID opcional de transação está especificada, a ID de transação não está codificada (0: não codificada; 1: codificada);

— ***default_timeoutDSI_flag***: campo de 1 bit que indica se o valor de intervalo DSI está codificado na estrutura de dados. Quando o valor-padrão definido pela operação é utilizado como o valor de intervalo DSI, ele não está codificado (0: não codificado; 1: codificado);

— ***default_leak_rate_flag***: campo de 1 bit que indica se a taxa de vazão está codificada na estrutura de dados. Quando o valor-padrão definido pela operação é utilizado como valor de taxa de vazão, ele não está codificado (0: não codificado; 1: codificado);

— ***component_size*** (tamanho de componente): campo de 32 bits que indica o tamanho total dos dados (unidade: byte) a serem transferidos pelo referido carrossel de objetos;

— ***transaction_id*** (ID da transação): valor de ID da transação a ser transmitido pelo componente. A ID não codificada de transação mostra a necessidade de aquisição DSI que tem ID de transação opcional;

— ***time_out_value_DSI*** (valor de intervalo DSI): campo de 32 bits que indica o valor de intervalo recomendado (unidade em milissegundos) para a recepção de toda a seção DSI do carrossel relevante. Quando o valor for 0xFFFFFFFF, será uma indicação de que não é valor de intervalo recomendável;

— ***leak_rate*** (taxa de vazão): campo de 22 bits que indica a taxa de vazão do *buffer* de transporte do receptor. A unidade é de 50 byte/s.

12.8 Localizador em descrição de aplicação

Alguns campos da descrição da aplicação podem conter localizadores (*locators*) para indicação em certos tipos de arquivos em diretórios, ou em transportes HTTP, url etc. O localizador utilizado na descrição da aplicação deve estar de acordo com a ARIB STD-B23:2004, Seção 14.

12.9 Descrição da aplicação

A descrição da aplicação deve estar de acordo com 12.10 a 12.15. A estrutura de sessão AIT a ser transmitida no modo de transmissão de sessão privada deve estar de acordo com 12.16. O campo de *string* na AIT pode ser codificado por um código de caracteres da ISO/IEC 8859-15. Por outro lado, os *strings* adquiridos pelo método *getName()* podem ser automaticamente convertidos em *strings* utilizáveis em Java.

Os tipos de aplicações devem ser modificados, inclusive o tipo de aplicação 0x0008, como aplicação reservada Ginga, e 0x0009, como aplicação Ginga-NCL, como demonstrado na Tabela 45.

Tabela 46 – Tipo de aplicação

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

12.10 Transmissão e monitoramento de descrição de aplicação

A transmissão e a monitoração da descrição de aplicação devem ocorrer no intervalo entre a atualização e a detecção de uma nova versão. Esse intervalo não pode exceder 30 s. Como para o processo de transmissão, a seção da tabela que está de acordo com a estrutura AIT é transmitida como ES que compreende o modo de transmissão via seção privada do programa.

Para a operação, o valor 0x0001, 0x0008 e 0x0009 são atribuídos como o *application_type* de Ginga-J, Ginga e Ginga-NCL, e 0x0001 é atribuído como o valor de *protocol_id* para significar o sistema de transmissão de carrossel de objetos.

O *selector_byte* no descritor de protocolo de transporte para o sistema de transmissão do carrossel de dados deve ser da mesma sintaxe, como para o caso do “protocolo de transporte de carrossel de dados” (*protocol_id* =0x0004). Para os detalhes da estrutura, ver Tabela 57.

12.11 Visibilidade da descrição de aplicação

Enquanto não for selecionado um novo serviço e a aplicação estiver recebendo sua sinalização é permitida a sua exibição mesmo se a descrição não estiver disponível.

12.12 Detalhes da descrição de aplicação

As descrições de aplicação são transmitidas com base no sistema de transmissão descrito em 12.3. Os descritores de aplicação especificados em 12.17 devem ser utilizados para o armazenamento das descrições de aplicação.

12.13 Tratamento da aplicação a partir de serviço previamente selecionado

A partir de um serviço previamente selecionado, se um aplicativo com um *service_bound_flag* 0 for executado quando uma seleção de serviços é realizada, ele deve continuar a execução em um novo serviço, se o mesmo pedido for sinalizado.

12.14 Descrição de aplicação específica ao Ginga-J

A descrição de aplicação específica deve estar de acordo com a Tabela 46.

12.15 Detalhes da descrição de aplicação Ginga

As descrições de aplicação Ginga são transmitidas com base no sistema de transmissão descrito em 12.3.

A estrutura AIT e a estrutura do descritor em AIT devem estar de acordo com 12.6. Os descritores de aplicação especificados em 12.18 devem ser usados para o armazenamento dos descritores da aplicação Ginga.

12.16 Sistema de codificação de informação de aplicação

12.16.1 Informação de aplicação

Na AIT, todas as informações relevantes à aplicação e exigências para *status* de partida estão armazenadas. Também é possível instruir o receptor para alterar o *status* de partida a partir da estação de TV através dos dados na AIT.

Todas as seções AIT que tenham o mesmo *Application_Type* no PID idêntico compreendem uma subtabela. O valor de *tag* de descrição na AIT deve ser único na AIT.

A estrutura de dados da informação de aplicação AIT é demonstrada na Tabela 47.

Tabela 47 – Tabela de informação da aplicação (AIT)

Estrutura de dados	Número 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*
}		

A descrição da *application_information_section ()* deve ser a seguinte:

— ***table_id*** (ID tabela): neste campo de 8 bits, 0x74 é armazenado para indicar que esta é a tabela AIT;

— ***section_syntax_indicator*** (indicador de sintaxe da seção): a indicação de sintaxe da seção é sempre “1” em um campo de 1 bit;

— ***section_length*** (extensão da seção): campo de 12 bits. Os primeiros 2 bits devem sempre ser “00.” Isso especifica o número de bits do campo de extensão de seção para a última seção, incluindo CRC32. O valor deve ser inferior a 1021 (0x3FD em hexadecimal);

— ***application_type*** (tipo de aplicação): campo de 16 bits que indica o valor do tipo de aplicação a ser transmitido na AIT. O valor atribuído a este campo deve estar conforme a Tabela 46;

— ***version_number*** (número da versão): campo de 5 bits que é o número da versão da subtabela. Deve-se adicionar um ao número da versão, quando uma alteração for feita na informação dentro da subtabela. Quando o valor alcançar “31,” o próximo valor será novamente “0.”

— ***current_next_indicator*** (próximo indicador atual): esta indicação de 1 bit é sempre “1.”;

— ***section_number*** (número de seção): campo de 8 bits que mostra o número da seção. O número da seção da primeira seção na subtabela é 0x00. Cada adição de uma seção que tem a ID de tabela e tipo de aplicação idênticos adiciona “1” ao número da seção;

— ***last_section_number*** (número da última seção): campo de 8 bits que especifica o número da última seção na subtabela à qual as seções pertencem;

— ***common_descriptors_length*** (extensão do enlace de descritores comuns): campo de 12 bits que especifica a extensão do byte da área comum subsequente de descritores. Os descritores dentro da área de descritores são aplicáveis a todas as aplicações na subtabela AIT;

— ***application_control_code*** (código de controle da aplicação): campo de 8 bits que especifica o código de controle do status da aplicação. O campo é dependente do valor de tipo de aplicação. Ver 12.16.4 para mais detalhes;

— ***recommended_resolution*** (resolução recomendada): a resolução da aplicação Ginga (correspondente às características de resolução) e a taxa de aspecto (correspondente às características de *display-aspect-ratio*) são indicadas neste campo. O campo *recommended resolution* é obrigatório nos serviços one seg e deve estar conforme a Tabela 37.

— ***application_loop_length*** (extensão do enlace de informação de aplicação): campo de 12 bits que especifica a extensão total do enlace em byte onde é armazenada a informação da aplicação subsequente;

— ***application_identifier ()*** (identificador de aplicação): ver Tabela 48;

— ***application_descriptors_loop_length*** (extensão do enlace de descritores de informação de aplicação): campo de 12 bits que especifica a extensão do byte da área subsequente de descritores. Estes descritores na área de descritores se aplicam somente à aplicação designada.

12.16.2 *Application ID* – Identificação de codificação da aplicação

Uma aplicação é identificada exclusivamente pelo identificador de aplicação demonstrado na Tabela 48. Este identificador é composto de uma estrutura de 6 bytes (48 bits).

Tabela 48 – Identificador da aplicação

Estrutura de dados	Número de bits	*String* de bits
application_identifier () {		
organization_id	32	*bslbf*
application_id	16	*bslbf*
}		*bslbf*

A descrição da *application_identifier ()* deve ser a seguinte:

— ***organization_id*** (ID da organização): campo de 32 bits que indica a organização que criou a aplicação. Esta ID armazena o número atribuído exclusivamente no mundo;

— ***application_id*** (ID da aplicação): campo de 16 bits que armazena o número que identifica a aplicação e que é atribuído exclusivamente na ID da organização. A ID de aplicação é dividida em duas faixas. Uma é a faixa de aplicação não assinada e a outra é a faixa de aplicação assinada. Esta divisão é feita para fins de segurança (ver Anexo B). A faixa do valor é respectivamente demonstrada na Tabela 49.

Tabela 49 – Valor de ID da aplicação

Valor de ID da aplicação	Tipo de aplicação
0x0000...0x3fff	Faixa de aplicação não assinada
0x4000...0x7fff	Faixa de aplicação assinada
0x8000...0xfffd	Reservado
0xfffe	*Wild card value* (indica todas as aplicações assinadas da mesma ID de organização)
0xffff	*Wild card value* (indica todas as aplicações da mesma ID de organização)

Na ID da aplicação, os valores 0xffff e 0xfffe são para *wild card value*. Estes não são utilizados para especificar a aplicação, mas, por exemplo, são usados como descritor para autorização de aplicação externa. 0xffff corresponde a todas as aplicações que têm a mesma ID de organização (*organization_id*). 0xfffe corresponde a todas as aplicações assinadas que têm a mesma ID de organização.

Algumas vezes o mesmo identificador de aplicação é usado entre aplicações de diferentes tipos, como no caso da execução da mesma função por diferentes tipos de aplicação, por exemplo.

NOTA Apenas um tipo aparece na coleção de subtabelas AIT do mesmo tipo de aplicação em um serviço.

12.16.3 Efeito sobre o ciclo de vida

A diretriz básica do efeito sobre o ciclo de vida é apresentada para a ocasião em que o serviço é alterado ou em que as aplicações que têm o mesmo identificador de aplicação são iniciadas, conforme a seguir:

— no *changeover* de serviço, se o *service_bound_flag* na aplicação ativa no serviço anterior for “0” e o identificador da aplicação existir na AIT do serviço recentemente selecionado, então a aplicação funciona continuamente, a menos que haja alguma restrição de recurso;

— no *changeover* de serviço, se o *service_bound_flag* na aplicação ativa no serviço anterior for “0” e se apenas a aplicação estiver na lista de descritores de autorização de aplicação externa, a aplicação trabalha continuamente, mesmo se a aplicação não fizer parte do serviço recentemente selecionado, a menos que haja alguma restrição de recurso;

— como para uma aplicação que tem um identificador de aplicação, apenas uma instância é ativada. Mesmo se uma outra aplicação tiver o mesmo identificador, ela não pode ser ativada se uma aplicação com o mesmo identificador de aplicação já tiver existido;

— isso afeta o comportamento da API de início de aplicação ou início automático da aplicação. Se o *service_bound_flag* “1” for configurado para a aplicação, ela terminará (*KILL*) a cada seleção de serviço.

12.16.4 Controle de aplicações de ciclo de vida

Para *Application Life Cycle Control*, mecanismos de sinalização devem ser fornecidos a partir da estação de transmissão, para controlar o ciclo de vida das aplicações do tipo padrão.

12.16.5 Acesso e saída do domínio da aplicação

Entering and Leaving the Application Domain Application é o domínio definido como uma coletânea de serviços com aplicações listadas na AIT. Isso significa que as aplicações são as listadas nos enlaces de informação de aplicação de AIT ou as listadas nos descritores de autorização de aplicação externa. Os serviços cujas aplicações não estão listadas como acima mencionado são considerados fora do domínio da aplicação.

12.16.6 Controle dinâmico do ciclo de vida das aplicações Ginga

O controle dinâmico do ciclo de vida das aplicações Ginga deve estar de acordo com 12.16.1 a 12.16.4.

Ginga define o “*application_control_code*” valor 0x07 como aplicações UNBOUND (ver Tabela 50).

Tabela 50 – Valores de código de controle de aplicação Ginga

Código	Identificador	Semântica
0x00		Reservado para uso futuro
0x01	AUTOSTART	Aplicativos com o código de controle AUTOSTART são iniciados automaticamente quando o receptor muda para este serviço
0x02	PRESENT	Os aplicativos com código de controle PRESENT não são iniciados automaticamente, mas são adicionados à lista de aplicativos disponíveis do receptor. O usuário pode então escolher iniciar este aplicativo, selecionando-o na lista

Tabela 50 (continuação)

Código	Identificador	Semântica
0x03	DESTROY	Aplicativos com código de controle DESTROY devem ser finalizados pelo gerenciador de aplicações logo que possível. Caso seja lançada uma exceção do tipo *javax.microedition.xlet.XletState ChangeException* durante uma tentativa de finalização, o aplicativo deve continuar em execução. Aplicativos sinalizados previamente com código de controle STORE podem opcionalmente ser mantidos no *cache*
0x04	KILL	Os aplicativos com o código de controle KILL devem ser incondicionalmente finalizados pelo gerenciador de aplicações. Aplicativos sinalizados previamente com o código STORE devem ser removidos do *cache*
0x05	PREFETCH	A aplicação NCL é carregada e a máquina Ginga NCL é preparada. A aplicação espera por um comando de edição NCL [a] antes de passar para o estado ativo
0x06	REMOTE	Identifica um aplicativo remoto que somente é lançado após a seleção do serviço
0x07	UNBOUND	Aplicativos com código de controle UNBOUND são similares a aplicativos sinalizados com PRESENT, exceto que o usuário decide se a aplicação pode ser armazenada para execução posterior. Caso o receptor não possua capacidade de armazenagem disponível para armazenar a aplicação ou o usuário escolha não instalar a aplicação, esta deve ser tratada como não disponível (não pode ser listada entre as aplicações disponíveis). Receptores sem suporte a armazenamento de aplicações devem ignorar as aplicações sinalizadas com este código de controle
0x08	STORE	Aplicativos com código de controle STORE não podem ser iniciados automaticamente, mas indicam que técnicas de *caching* podem ser utilizadas de modo a acelerar o carregamento de seus recursos durante a inicialização. Caso a plataforma não tenha capacidade para realizar técnicas de *caching* pró-ativo ou armazenamento de dados, aplicações sinalizadas como STORE devem ser tratadas de modo idêntico às aplicações sinalizadas com PRESENT
0X09 - 0Xff		Reservado para uso futuro

[a] Ver ABNT NBR 15606-2:2011, Tabela 55.

12.17 Descritores para AIT – Descritores para transmissão de informações das aplicações

12.17.1 Descritores comuns

Os descritores a serem comumente utilizados na AIT, independentemente do tipo de aplicação, são descritos em 12.17.2 a 12.17.12.

12.17.2 Descritor de aplicação

Um descritor de aplicação é armazenado no enlace do descritor de informação da aplicação AIT para cada aplicação (ver Tabela 51).

Tabela 51 – Estrutura do descritor de aplicação

Estrutura de dados	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	
}		
}		

A descrição da *application_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (*tag descritor*): campo de 8 bits; 0x00 é armazenado para indicar que este é o descritor mencionado;

— ***application_profiles_length*** (extensão de informação do perfil da aplicação): campo de 8 bits que indica a extensão total de informação do perfil da aplicação que está incluída no enlace subsequente;

— ***application_profile*** (perfil da aplicação): campo de 16 bits. O perfil da aplicação que pode executar a aplicação é armazenado. Se o perfil for montado no receptor, significa que a aplicação é executável. Os detalhes de perfil são definidos para cada tipo de aplicação;

— ***version.major*** (versão principal): campo de 8 bits que indica a versão principal do perfil acima mencionado;

— ***version.minor*** (versão secundária): campo de 8 bits que indica a versão secundária do perfil acima mencionado;

— ***version.micro*** (microversão): campo de 8 bits que indica a microversão do perfil acima mencionado;

Os quatro primeiros campos descritos acima compreendem um perfil mínimo para execução desta aplicação. O terminal Ginga inicia essa aplicação se qualquer uma das seguintes fórmulas teóricas for aplicável e qualquer perfil evidenciar “verdadeiro” na informação de perfil da aplicação:

(perfil da aplicação $\in$ coleta dos perfis montados no terminal)

E {(versão principal da aplicação < versão principal do terminal para o perfil)

OU [(versão principal da aplicação – versão principal do terminal para o perfil)

E ({versão secundária da aplicação < versão secundária do terminal para o perfil }

OU {[versão secundária da aplicação = versão secundária do terminal para o perfil]

E [microversão da aplicação ≤ microversão do terminal para o perfil]})] }

As definições do perfil detalhado da plataforma em aplicação Ginga-J e Ginga-NCL devem estar de acordo com a ABNT NBR 15606-1.

A codificação do perfil em aplicação Ginga-J deve estar de acordo com a ABNT NBR 15606-4.

A codificação do perfil da aplicação Ginga-NCL deve estar de acordo com a ABNT NBR 15606-2.

— ***service_bound_flag*** (*service bound flag*): campo de 1 bit que indica se a aplicação está efetiva apenas no presente serviço. Se o campo for “1,” a aplicação é relevante apenas ao serviço atual. Quando o serviço é alterado para outro serviço, a finalização do processamento da aplicação mencionada será iniciada;

— ***visibility*** (visibilidade): campo de 2 bits que indica se a aplicação é visível aos usuários finais quando está ativada. As definições de *status* para o valor de visibilidade são demonstrados na Tabela 52;

— ***application_priority*** (prioridade da aplicação): campo de 8 bits que indica a prioridade relativa entre as aplicações notificadas no serviço;

— ***transport_protocol_label*** (rótulo de protocolo de transporte): campo de 8 bits que armazena o valor para identificação única do protocolo de transporte que transporta a aplicação. O valor corresponde ao valor do campo *transport_protocol_label* do descritor do protocolo de transporte.

Tabela 52 – Visibilidade

Valor visível	Descrição
0	Esta aplicação não é visível às outras aplicações via enumeração de aplicação API, nem aos usuários via navegador, exceto por erro de informação de *logout* e similares
1	Esta aplicação não é visível para os usuários, mas é visível a partir de outras aplicações via enumeração de aplicação API
10	Reservado para uso futuro
11	Esta aplicação é visível tanto para os usuários como para as outras aplicações

12.17.3 Descritor do nome de aplicação

Um descritor de nome da aplicação é armazenado para cada aplicação no enlace do descritor de informação da aplicação AIT. O nome da aplicação é usado para diferenciação e fornece a informação aos usuários (ver Tabela 53).

Tabela 53 – Estrutura do descritor de nome da aplicação

Estrutura de dados	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*
}		
}		
}		

A descrição da *application_name_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits; 0x01 é armazenado para indicar que este é o descritor de nome da aplicação;

— ***ISO_639_language_code*** (código da linguagem): campo de 24 bits que identifica a linguagem do descritor do nome da aplicação. O código da linguagem é indicado pelo código *three-alphabetical -letter* como disposto na ISO 639-2. Cada letra é codificada em 8 bits, de acordo com a ISO/IEC 8859-1, e inserida de acordo com a ordem no campo de 24 bits;

— ***application_name_length*** (descrição do comprimento do nome da aplicação): campo de 8 bits que indica o comprimento do byte da subsequente descrição do nome da aplicação;

— ***application_name_char*** (descrição do nome da aplicação): campo que especifica a texto de descrição do nome da aplicação. Este texto de descrição é usada como informação da aplicação pelos usuários e não pode ter valor nulo.

12.17.4 Descritor da informação dos ícones da aplicação

O descritor da informação dos ícones da aplicação é armazenado apenas uma vez para cada aplicação (ver Tabela 54). O descritor indica a informação sobre o ícone relevante para a aplicação. O formato dos conteúdos do ícone é codificado pelo PNG e usa um sistema proporcionado de acordo com a ABNT NBR 15606-1.

Tabela 54 – Estrutura do descritor da informação dos ícones da aplicação

Estrutura de dados	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*
}		
}		

A descrição da *application_icons_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits que armazena 0x0B, que indica o descritor dito;

— ***icon_locator_length*** (comprimento do localizador de ícone): campo de 8 bits que indica o comprimento de byte do subsequente locador do ícone;

— ***icon_locator_byte*** (localizador de ícone): campo de 8 bits que armazena o localizador do arquivo de imagem estática como um ícone. Este localizador deve ser colocado antes do nome do arquivo de ícone e de acordo com 12.16.1, 12.16.2, 12.16.3 e 12.16.5. Para a aplicação Ginga-J, existem regras no caso em que *application_type* for 0x0008 e há um caminho relativo da base do diretório definido pela aplicação Ginga-J armazenado;

— ***icon_flags* (ícone de *flag*)**: este campo de 16 bits armazena a *flag* que indica o tamanho e o aspecto da relação do ícone usável. Os detalhes devem estar de acordo com 12.16.1, 12.16.2, 12.16.3 e 12.16.5, e a codificação de cada caso é conforme mostrado na Tabela 55. O valor é armazenado após o OR (endereço lógico) pela unidade.

Tabela 55 – Bits do ícone da *flag*

Bits de ícone de *flag*	Tamanho dos ícones e relação de aspecto
0000 0000 0000 0001	32 x 32 para exibir pixels quadrados
0000 0000 0000 0010	32 x 32 para transmitir pixels em uma tela 4:3
0000 0000 0000 0100	24 x 32 para transmitir pixels em uma tela 16:9
0000 0000 0000 1000	64 x 64 para exibir pixels quadrados

Tabela 55 (continuação)

Bits de ícone de *flag*	Tamanho dos ícones e relação de aspecto
0000 0000 0001 0000	64 x 64 para transmitir pixels em uma tela 4:3
0000 0000 0010 0000	48 x 64 para transmitir pixels em uma tela 16:9
0000 0000 0100 0000	128 x 128 para exibir pixels quadrados
0000 0000 1000 0000	128 x 128 para transmitir pixels em uma tela 4:3
0000 0001 0000 0000	96 x 128 para transmitir pixels em uma tela 16:9
xxxx xxx0 0000 0000	*reserved_future_use*

O nome do arquivo do ícone é codificado de acordo com o valor do ícone da *flag* descrito acima. A codificação do nome do arquivo segue na descrição abaixo:

filename = *icon_locator* “/ginga.icon.” *hex_string*

hex_string = 4*4*hex*

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 arquivo deve conter um ícone. O ícone contido no arquivo deve ter o formato especificado pelos quatro dígitos *postscript* deste nome do arquivo. No caso do campo *icon_flags* do a*pplication_icons_descriptor* indicando a presença de múltiplos ícones, cada um dos arquivos indicados deve ter seu próprio ícone.

EXEMPLO *icon_flags* com valor 0x0005, o diretório especificado pelo *icon_locator* contém 2 arquivos nomeados ginga.0004 (para 24 x 32 *square pixel*) e ginga.0001 (para 32 x 32 *square pixel*)

12.17.5 Descritor de autorização de aplicação externa

O descritor de autorização de aplicação externa pode ser armazenado em um ou mais no primeiro enlace descritor comum do AIT, conforme necessidade. Em cada descritor há informação relevante armazenada das aplicações externas. Uma aplicação externa é aquela que pode operar continuamente com as aplicações listadas na subtabela AIT, mas que não podem ser ativadas pelo serviço.

A autorização externa é aplicável para a aplicação que tem *application_identifier()* na *application_type* especificada pela AIT que inclui este descritor (ver Tabela 56).

Tabela 56 – Estrutura do descritor de autorização de aplicação externa

Estrutura de dados	Número de bits	*String* de bits
external_application_authorisation_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*

Tabela 56 (continuação)

Estrutura de dados	Número de bits	*String* de bits
for (i=0; i<N ; i++) {		
application_identifier ()		
application_priority	8	*uimsbf*
}		
}		

A descrição do *external_application_authorisation_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits onde 0x05 é armazenado para indicar o descritor de autorização de aplicação;

— ***application_identifier()*** (identificador de aplicação): campo de 48 bits que indica a aplicação em que a referência externa está disponível. Para detalhes da estrutura do campo, ver 12.16.1;

— ***application_priority*** (prioridade da aplicação): campo de 8 bits que indica a prioridade da presente aplicação na suposição do contexto do serviço dito. Para detalhes da prioridade, ver 12.17.2.

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

O *transport protocol descriptor* indica a identificação do protocolo de transporte relevante para o componente do serviço e armazena informação sobre o protocolo. Este protocolo é armazenado no primeiro enlace do descritor comum ou enlace do descritor da informação da aplicação. Quando isto é armazenado no enlace de descritor comum, é aplicável para todas as subtabelas do AIT. O descritor do protocolo de transporte no enlace do descritor da informação da aplicação descreve o protocolo de transporte adicional para ser usado especificamente na aplicação (ver Tabela 57).

Tabela 57 – Estrutura do descritor de protocolo de transporte

Estrutura de dados	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*
}		
}		

A descrição do *transport_protocol_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits que armazena 0x02, que indica que este é o atual descritor de transporte;

— ***protocol_id*** (identificação de protocolo): campo de 16 bits que indica o protocolo que transporta a aplicação (ver Tabela 58);

Tabela 58 – Valores da identificação de protocolo

Valores	Descrição
0x0000	Reservado para o futuro
0x0001	Protocolo de transporte do carrossel de objetos
0x0002	IP através de *multiprotocol* e encapsulamento definido na EN 301192 e ETSI TR 101202
0x0003	Transporte via HTTP sobre o canal de interatividade conforme descrito na seção 12.17.9
0x0004	Protocolo de transporte do carrossel de dados
0x0005...0xFFFF	Reservado para o futuro

— ***transport_protocol_label*** (rótulo de protocolo de transporte): campo de 8 bits que indica o valor que identifica unicamente o protocolo de transporte na seção AIT. Para o descritor da aplicação, ver o dispositivo de conexão (por exemplo, o *stream* elementar do carrossel) que transporta a aplicação com estes valores;

— ***selector_byte*** (seletor de área): campo de 8 bits que armazena informação adicional provida por cada identificação de protocolo. A estrutura de dados descrita na área é especificada separadamente por cada *protocol_id* (ver Tabela 59).

Tabela 59 – Estrutura do *selector_byte*

Valores do *protocol_id*	Estrutura do seletor de área
0x0000	Reservado para o 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 o futuro

12.17.7 Transporte através do OC (carrossel de objeto)

A estrutura de dados é mostrada na Tabela 60 para o protocolo de transporte do carrossel de objetos (*protocol_id* =0x0001) e o protocolo de transporte do carrossel de dados (*protocol_id* =0x0004).

Tabela 60 – Estrutura do protocolo de transporte do descritor do seletor de área (no caso de protocolo de transporte do carrossel de objetos/protocolo de transporte de carrossel de dados)

Estrutura de dados	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*

A descrição deve ser a seguinte:

— ***remote_connection*** (conexão remota): se este campo de 1 bit estiver em "1," isto mostra que o componente de serviço vigente é transmitido de outra fonte que aquela que transmite para AIT. Um serviço como este não é executado automaticamente, mas é visível e viável de se iniciar através da API. Além disso, *REMOTE* é armazenado de tal maneira na aplicação em *application_control_code*;

— ***original_network_id*** (identificador da rede original): quando *remote_connection* estiver "1," o identificador da rede original para o serviço de transmissão vigente é armazenado;

— ***transport_stream_id*** (identificador de transport *stream*): quando *remote_connection* é "1," o identificador de *transport stream* da transmissão de serviço atual é armazenado;

— ***service_id*** (identificador de serviço): quando *remote_connection* estiver "1," o serviço (identificado pelo identificador de serviço) da transmissão vigente é armazenado;

— ***component_tag*** (componente de *tag*): indica o componente do serviço principal que transmite a aplicação. No caso de carrossel de dados, o *stream* elementar que transmite o carrossel automaticamente montado no início da aplicação é indicado. No caso de carrossel de objetos, o *stream* elementar que transmite DSI é indicado.

12.17.8 Transporte através de IP

Quando o identificador de protocolo for 0x0002, o seletor de bytes no descritor do protocolo de transporte deve ser conforme a Tabela 61, para prover toda a informação necessária à obtenção das aplicações Ginga e componentes de dados das aplicações entregues pelo protocolos de IP.

Tabela 61 – Sintaxe do seletor de bytes para o transporte de IP

Estrutura de dados	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*
}		

A descrição deve ser a seguinte:

— ***remote_connection***: este e os três campos associados (*original_network_id, transport_stream_id* e *service_id*) têm semântica e sintaxe idênticas aos campos com os mesmos nomes, de acordo com 12.17.7;

— ***alignment_indicator***: campo de 1 bit que indica o alinhamento que existe entre os bytes do *datagram_section* e os bytes de transporte de *stream*;

— ***URL_length***: campo de 8 bits que indica o número de bytes na URL;

— ***URL_byte***: estes bytes formam uma URL, conforme RFC 2396.

Para a URL usando o campo do "servidor" incluindo a notação *host:port*, conforme definido na RFC 2396, somente endereços numéricos IP serão usados para identificar as transmissões de IP transportadas no canal de radiodifusão uma vez que não há *Domain Name Service* específico para a radiodifusão. IP para mapeamento MAC será feito conforme descrito na RFC 1112.

NOTA Esta Norma não define o formato de URL a ser sustentado por este descritor. Por isso, o formato de URL não pode ser usado de maneira interoperável.

12.17.9 Transporte via canal de interatividade

Quando o valor do protocol ID é 0x0003, o *selector byte* no descritor de protocolo de transporte (*Transport protocol descriptor*) deve estar de acordo com a Tabela 62. Desta forma é permitido codificar um número de URL. Para eficiência na codificação de URL similares, a codificação divide a URL

dentro de uma parte comum e um conjunto de extensões de URL. O conjunto de URL pode identificar arquivos ZIP ou URL bases terminadas no caracter "/", que encapsula parte do sistema de arquivos.

Múltiplos descritores (*Transport protocol descriptor*) com o valor de ID 0x0003 com o mesmo rótulo de protocolo de transporte podem ser fornecidos para definir um conjunto maior de URL para descrição do sistema de arquivos.

Tabela 62 – Sintaxe do *selector bytes* para o transporte no canal de interatividade

Sintaxe	Bits	Mnemônico
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*
}		
}		
}		

A sintaxe detalhada do *selector bytes* para o transporte no canal de interatividade é:

— ***URL_base_length***: este campo de 8 bits fornece o número de bytes na parte da URL base;

— ***URL_base_byte***: estes bytes da primeira parte de uma URL HTTP de acordo com o HTTP 1.0 (ver RFC 1945), ou a primeira parte de outra URL HTTPS de acordo com a HTTPS (ver RFC 2818). Quando uma HTTP URL é encontrada, HTTP deve ser usado de acordo com a RFC 2616.

Quando a URL HTTPS é encontrada, HTTPS deve esta de acordo com a RFC 2818.

Para qualquer outra URL, a implementação provedora de serviço de transporte de protocolos no canal de interatividade registrada é usada para o esquema de URL associada, caso exista. No caso de não existir provedor para o esquema, o descritor *transport protocol descriptor* é ignorado.

— ***URL_extension_count***: este campo de 8 bits indica o número de URL relacionadas pelo descritor;

— ***URL_extension_length***: este campo de 8 bits indica o número de bytes na parte de extensão da URL;

— ***URL_extension_byte***: estes bytes formam parte posterior de uma URL HTTP de acordo com HTTP 1.0 (ver RFC 1945), ou a parte posterior de uma URL HTTPS de acordo com a RFC 2818 ou outra URL na qual o esquema seja suportado por uma a implementação provedora de serviço de transporte de protocolos no canal de interatividade registrada.

URL são formadas pela concatenação de extensão de URL com o URL base anterior. A URL formada identifica o diretório do sistema de arquivo ou o arquivo ZIP especifico.

12.17.10 Descritor de sinalização de IP

O descritor de sinalização IP é definido para o uso tanto no “comum” quanto na “aplicação” do enlace da AIT. Este descritor indica a identificação da organização que provê os *streams multicast* usados por todas as aplicações (quando presente no enlace “comum”) ou pela aplicação de sinalização particular (quando presente no enlace da “aplicação”). Para a definição do INT, ver ETSI EN 301 192.

Este descritor e o INT com *action_type* 0x01 devem ser usados pelas aplicações confiando na presença dos *streams multicast* IP no *link* da difusão. O conhecimento da identificação presente no descritor habilita a recuperação da tabela apropriada de notificação de IP (INT) com *action_type* 0x01 que contém a correspondência entre o endereço do IP *multicast*, *port* e localização do *stream* (ver Tabela 63).

Tabela 63 – Sintaxe do descritor de sinalização de IP

Sintaxe	Número de bits	Mnemônicos
ip_signalling_descriptor () {		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
platform_id	24	*uimsbf*

A descrição do *ip_signalling_descriptor ()* deve ser a seguinte:

— ***descriptor_tag***: campo de 8 bits que com valor 0x11 identifica o descritor;

— ***descriptor_length***: campo de 8 bits que identifica o número de bytes, seguindo o comprimento do campo;

— ***platform_id***: campo de 24 bits contendo um *platform_id* da organização provendo streams IP/MAC no transporte de *streams*/serviços. Alocações do valor de *platform_id* são encontradas no ETSI TS 101 162.

12.17.11 *Pre-fetch descriptor* (descritor de pré-busca)

Somente um descritor *pre-fetch* é armazenado no enlace do descritor da informação da aplicação AIT, conforme necessidade. Este descritor é definido para ser usado pelo carrossel de objetos (*protocol_id*=0x0001). Cada descritor é associado a um descritor de protocolo de transporte através do rótulo do protocolo de transporte (ver Tabela 64).

O terminal Ginga pode adquirir adiantadamente módulos denotados pelo rótulo terminal para acelerar o tempo de início da aplicação. Assim como para os rótulos, os descritores de rótulos especificados pela transmissão do carrossel de objetos são usados de acordo com a ISO/IEC 13818-6:1998, Anexo B. A difusão desta sinalização é opcional.

Tabela 64 – Estrutura do descritor de *pre-fetch*

Estrutura de dados	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*
}		
}		

A descrição do *prefetch_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (*descriptor tag*): campo de 8 bits onde 0x0C é armazenado para mostrar o descritor de *pre-fetch*;

— ***transport_protocol_label*** (rótulo do protocolo de transporte): campo de 8 bits que armazena o rótulo do protocolo de transporte para especificar o carrossel de objetos que transmite os módulos referidos no descritor *pre-fetch* acima. Para o rótulo do protocolo de transporte, ver 12.17.6;

— ***label_length*** (comprimento de rótulo): campo de 8 bits que indica o comprimento de bytes do rótulo de descrição a ser incluído no consequente enlace;

— ***label_char*** (descrição de rótulo): campo de 8 bits. Um rótulo do módulo está armazenado. Isto corresponde à descrição de rótulo transmitida pelo descritor de rótulos que está armazenado no *userInfo* do *moduleInfo* do DII no carrossel de objetos;

— ***prefetch_priority*** (prioridade *pre-fetch*): campo de 8 bits que armazena os valores de 1 a 100. Estes valores mostram a prioridade do *pre-fetch*. Maior valor mostra a maior prioridade.

12.17.12 Descritor de localização DII

Para cada aplicação somente um descritor de localização DII pode ser dado, se necessário. Este descritor pode ser armazenado tanto no enlace de descritor comum quanto no enlace do descritor da

informação da aplicação. Isto é definido para ser usado pelo sistema de transmissão do carrossel de objetos (*protocol_id*=0x0001). Cada descritor é associado a um descritor de protocolo de transporte através de um rótulo de protocolo de transporte.

O grupo de módulos como uma parte do carrossel de objetos DSM-CC é notificado pelo *DownloadInfoIndication* (DII). Todas as mensagens DII que mostram a existência do carrossel de objetos não podem ser listadas em localização uma por uma.

É necessário fazer todas as mensagens DII disponíveis em ordem para encontrar os módulos correspondentes aos rótulos para fazer o *pre-fetch* (ver 12.17.11). O descritor de localização DII lista a localização destes DII.

No caso de o descritor de localização DII não estar incluído, somente os módulos de indicação DII que incluem o *ServiceGateway* são encontrados.

Os enlaces da identificação DII no descritor estão localizados em ordem de importância. Isto é, DII que tem alta prioridade de *pre-fetch* será localizada no começo do enlace. O receptor montado com mecanismos de módulos-base de *pre-fetch* verifica o DII na ordem listada no descritor de localização de DII (ver Tabela 65).

Tabela 65 – Estrutura do descritor de localização de DII

Estrutura de dados	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*
}		
}		

A descrição do *DII_location_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits que armazena 0x0D que indica que este é o *DII_location_descriptor*;

— ***transport_protocol_label*** (rótulo de protocolo de transporte): campo de 8 bits que armazena o rótulo do protocolo de transporte que especifica o carrossel de objetos que é transmissor do descritor de *pre-fetch modulesreferredin*. Para detalhes do rótulo do protocolo de transporte, ver 12.17.6;

— ***DII_identification*** (identificação DII): campo de 15 bits que armazena o valor que especifica o *DIImessage*. Este valor corresponde ao bit 1-15 da transação ID no *dsmMessageHeader()* do *DII message*;

— ***association_tag*** (associação da *tag*): campo de 16 bits que indica o relacionamento com *DII message* que é transmitido (por exemplo, o *stream* elementar).

12.18 Descritor de aplicação Ginga

12.18.1 Estrutura do descritor de aplicações Ginga

Um descritor é armazenado no enlace do descritor da informação da aplicação AIT para cada aplicação Ginga. Isto indica o parâmetro de informação para o início da aplicação (ver Tabela 66).

Tabela 66 – Estrutura do descritor de aplicação Ginga

Estrutura de dados	Número de bits	Mnemônico
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*
}		
}		
}		

A descrição do *ginga_application_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits onde 0x03 é armazenado para indicar o descritor de aplicação Ginga-J e 0x06 é armazenado para indicar no respectivo descritor de aplicação Ginga NCL;

— ***parameter_length*** (comprimento do parâmetro): campo de 8 bits que mostra o comprimento de bytes da descrição de parâmetro subsequente;

— ***parameter_byte*** (descrição de parâmetro): campo de 8 bits. O *string* a ser dado à aplicação como parâmetro é armazenado.

12.18.2 Descritor de localização de aplicação Ginga

O descritor é armazenado no enlace do descritor da informação da aplicação AIT uma vez para cada aplicação Ginga. Este armazena a informação de caminho necessária na aplicação (ver Tabela 67).

Tabela 67 – Estrutura do descritor de localização de aplicação Ginga

Estrutura de dados	Número de bits	Mnemônico
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*
}		
}		

A descrição do *ginga_application_location_descriptor ()* deve ser a seguinte:

— ***descriptor_tag*** (descritor de *tag*): campo de 8 bits onde 0x04 é armazenado para indicar descritor de localização de aplicação Ginga-J e 0x07 é armazenado para indicar o respectivo descritor de aplicação Ginga NCL;

— ***base_directory_length*** (comprimento do diretório base): campo de 8 bits que indica o comprimento dos bytes a serem incluídos nos enlaces subsequentes. O valor armazenado será 1 ou maior;

— ***base_directory_byte*** (byte do diretório base): campo de 8 bits. O nome do diretório do caminho do arquivo do sistema deve ser armazenado por uma letra delimitadora, usando "/" (0x2F). Este nome de diretório é usado pelo diretório-base no caminho relativo. Se o diretório do caminho do arquivo do sistema for designado conforme um diretório-base, "/" deve ser armazenado;

— ***entitypath_extension_length*** (comprimento do caminho da entidade): campo de 8 bits que indica o comprimento do byte do caminho de classe adicional subsequente;

— ***entitypath_extension_byte*** (caminho da entidade): campo de 8 bits. Quando um caminho de classe é designado por um diretório outro do que o diretório-base, o nome do caminho de classe é armazenamento. O nome do diretório do caminho do arquivo do sistema é armazenado pela letra delimitadora usando "/" (0x2F). Se houver mais de um caminho, eles devem ser armazenados por enumeração com ";" (0x3B);

— ***initial_entity*** (entidade de ativação inicial): campo com 8 bits. O nome da entidade inicial da aplicação que consta no sistema de arquivos armazenados.

13 Especificação da transmissão da mensagem do evento

13.1 Mensagem de evento

A especificação da transmissão da mensagem do evento provê um meio para enviar mensagens de informação imediatamente ou em horários específicos para uma aplicação operada em uma unidade de receptor de uma estação de difusão.

A especificação da transmissão da mensagem do evento definido nesta parte da ABNT NBR 15606 é estendida para negociar os vários tempos, apontando métodos pela aplicação baseada na especificação do descritor de *stream* e sua especificação da transmissão de seção DSM-CC especificada na ISO/IEC 13818-6.

13.2 Descritores de *stream*

13.2.1 Descritor de *stream* DSM-CC

Esta seção está de acordo com os requisitos da ISO/IEC 13818-6.

Descritores de *stream* podem ser usados para prover informação DSM-CC, que é correlacionada com um *transport stream* ou *program stream* MPEG-2. Estes descritores estão em formato de programas e elementos de programas descritores, conforme definido na ISO/IEC 13818-1. Os descritores de *stream* DSM-CC devem ser somente carregados em uma *DSMCC_section* (ver ISO/IEC 13818-6:1998, Seção 9). Isto cria um espaço identificador único (daquele definido por ISO/IEC 13818-1) para valores de descritor de *tag*. O formato geral de todos os descritores de *stream* definidos nesta parte da ABNT NBR 15606 está mostrado na Tabela 68.

Tabela 68 – Descritor de *stream* DSM-CC

Sintaxe	Número de bits	Mnemônico
dsmccStreamDescriptor () {		
descriptorTag	8	*uimsbf*
descriptorLength	8	*uimsbf*
descriptor()		
}		

A descrição do *dsmccStreamDescriptor ()* deve ser a seguinte:

— ***descriptorTag***: campo de 8 bits que identifica cada descritor. A extensão de possíveis valores para o *descriptorTag* é mostrada na Tabela 69;

— ***descriptorLength***: campo de 8 bits que especifica o número de bytes do descritor imediatamente depois do campo *descriptorLength*.

Tabela 69 – Valor do campo *DescriptorTag*

Descritor de *tag*	Descrição
0x16	ISO/IEC 13818-6 reservado
0x17	Descritor de referência NPT
0x18	Descritor de *endpoint* NPT
0x19	Descritor de modo *stream*
0x1A	Descritor de evento de *stream*

13.2.2 Descritor de referência NPT

Para habilitar a determinação de tempo do NPT de um evento, o descritor de referência NPT é definido. O formato do descritor de referência é mostrado na Tabela 70.

Tabela 70 – Descritor de referência NPT

Sintaxe	Número de bits	Mnemônico
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*
}		

A descrição do *NPTReferenceDescriptor ()* deve ser a seguinte:

— ***descriptorTag***: campo de 8 bits que identifica o tipo do descritor de *stream*. O valor do campo *descriptorTag* para o descritor de referência NPT é mostrado na Tabela 69;

— ***descriptorLength***: campo de 8 bits que especifica o número de bytes do descritor imediatamente depois do campo *descriptorLength*;

— ***postDiscontinuityIndicator***: campo de 1 bit. Um valor de 0 indica que o descritor de referência NPT é válido na recepção. Um valor de 1 indica que o descritor de referência NPT torna válido no próximo tempo a base de descontinuidade do sistema, conforme definido na ISO/IEC 13818-1;

— ***dsm_contentId***: campo de 7 bits que identifica que um conjunto aninhado de conteúdo está sendo apresentado. Esse campo pode ser usado para indicar a transição para uma base de tempo diferente NPT dentro de uma base de tempo existente NPT. Por exemplo, este campo pode ser trocado quando um comercial é apresentado dentro de um programa de televisão e então novamente trocado quando o programa de televisão é reiniciado;

— ***STC_Reference***: inteiro sem sinal de 33 bits, que indica que o valor de STC para o qual NPT iguala o campo *NPT_Reference*. O valor do campo *STC_Reference* é especificado em unidades de períodos do sistema de frequência de relógio, conforme definido na ISO/IEC 13818-1, dividido por 300, rendendo unidades de 90 kHz. *STC_Reference* é derivado de um sistema de frequência de relógio, conforme mostrado na equação abaixo:

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

onde

STCNPT(k) é o valor do sistema de tempo do relógio "*System Time Clock*" quando o NPT iguala o valor de *NPT_Reference*;

— ***NPT_Reference***: inteiro com sinal de 33 bits indicando o valor de NPT no valor dado de STC no campo *STC_Reference field*;

— ***scaleNumerator***: inteiro com sinal de 16 bits usado com o *scaleDenominator*, um inteiro sem sinal de 16 bits, para definir a extensão da troca de NPT em relação ao STC. Um valor de 1 para *scaleNumerator* com um valor de 1 para *scaleDenominator* indica que o NPT está trocando em uma taxa equivalente ao STC, rendendo a taxa-padrão da apresentação. Um valor de 0 para *scaleNumerator* com um valor não zero para *scaleDenominator* indica que NPT não está mudando em relação ao STC, rendendo um valor constante do NPT. Um valor de 0 para *scaleNumerator* com um valor de 0 para *scaleDenominator* indica que os campos *scaleNumerator* e *scaleDenominator* não estão providos no descritor de referência NPT. Um valor não zero para *scaleNumerator* com um valor de 0 para *scaleDenominator* não será usado (ver Tabela 71).

Tabela 71 – *ScaleNumerator*

scaleNumerator	***scaleDenominator***	**Semântica**
0	0	Indica que o *scaleNumerator* e o *scaleDenominator* não são usados
0	Outro além de 0	NPT continua o valor constante irrelevante para STC
1	1	NPT e STC avançam na mesma taxa
Outro além de 0	0	Tal combinação não pode ser usada

13.3 Descritor de modo de *stream*

O descritor de modo de *stream* contém informação sobre o modo da máquina de estado do *stream*, permitindo que os clientes sincronizem melhor suas ações com as mudanças de estado do *stream*. O formato do descritor de modo do *stream* é mostrado na Tabela 72.

Tabela 72 – Descritor de modo de *stream*

Sintaxe	Número de bits	Mnemônico
StreamModeDescriptor () {		
descriptorTag	8	*uimsbf*
descriptorLength	8	*uimsbf*
streamMode	8	*uimsbf*
Reserved	8	*bslbf*
}		

A descrição do *StreamModeDescriptor ()* deve ser a seguinte:

— ***descriptorTag***: campo de 8 bits que identifica o tipo do descritor do *stream*. O valor do campo *descriptorTag* para o descritor de modo do *stream* é mostrado na Tabela 69;

— ***descriptorLength***: campo de 8 bits que especifica o número de bytes do descritor imediatamente depois do campo *descriptorLength*;

— ***streamMode***: campo de 8 bits cujo valor indica o estado atual da máquina de estado do *stream*. Os valores para *streamMode* são mostrados na Tabela 73. Os estados da máquina de estado de *stream* são definidos na ISO/IEC 13818-6:1998, Seção 5.

Tabela 73 – Valores do campo *StreamMode*

Descritor de modo	Descrição
0	Aberto
1	Pausa
2	Transporte
3	Pausa no transporte
4	Transporte de busca
5	Pausa de transporte de busca
6	Transporte de pausa da busca
7	Fim de *stream*
8	Transporte de pré-busca
9	Pausa de transporte de pré-busca
10 – 255	ISO/IEC 13818-6 reservado

13.4 Descritores de evento de *stream*

O descritor de evento de *stream* contém informações que permitem a transmissão de eventos específicos, conforme definido na ISO/IEC 13818-6:1998, Seção 5, de modo que eles possam ser

sincronizados com o *stream*. A definição de evento neste contexto não é a mesma se o evento for relacionado com NPT. O formato do descritor de evento de *stream* é mostrado na Tabela 74.

Tabela 74 – Descritor de evento de *stream*

Sintaxe	Número de bits	Mnemônico
StreamEventDescriptor () {		
descriptorTag	8	*uimsbf*
descriptorLength	8	*uimsbf*
eventide	16	*uimsbf*
Reserved	31	*bslbf*
eventNPT	33	*uimsbf*
for(i=0;i<N;i++){		
privateDataByte	8	*uimsbf*
}		
}		

A descrição do *StreamEventDescriptor ()* deve ser a seguinte:

— ***descriptorTag***: campo de 8 bits que identifica o tipo do descritor de *stream*. O valor do campo *descriptorTag* para o descritor de evento de *stream* é mostrado na Tabela 69;

— ***descriptorLength***: campo com 8 bits que especifica o número de bytes do descritor imediatamente depois do campo *descriptorLength*;

— ***eventid***: campo de 8 bits cujo valor é um tipo de evento específico da aplicação;

— ***eventNPT***: inteiro sem sinal, cujo valor é o valor de NPT quando o evento ocorreu, ou o valor de NPT quando o evento ocorrer;

— ***privateDataByte***: campos que permitem a inclusão de dados específicos de aplicação no descritor de eventos de *stream*.

13.5 Descritor de evento geral

O descritor de evento geral (*general_event_descriptor*) é um descritor para comunicar informações aplicáveis a mensagens de eventos.

A estrutura de dados do descritor de evento geral é mostrada na Tabela 75.

Tabela 75 – Descritor de evento geral

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A descrição do *General_event_descriptor ()* deve ser a seguinte:

— ***event_msg_group_id*** (identificador de grupo de mensagem de evento): campo de 12 bits que identifica o grupo de mensagens a serem recebidas pelo aplicador. Detalhes das operações são especificados em cada identificador de codificação de dados. Quando operando um evento de mensagem com a identificação de mais de um grupo de mensagens ao mesmo tempo, somente os descritores de evento geral com o identificador do mesmo grupo de mensagens devem ser incluídos em uma seção DSM-CC;

— ***time_mode*** (modo de tempo): campo de 8 bits que indica o método para designar o tempo quando evento de mensagem é gerado (ver Tabela 76);

Tabela 76 – Modo de tempo

Time_mode	Método de designação de tempo	Semântica
0x00	nenhum	A mensagem de evento é gerada imediatamente após a recepção
0x01	*mjd_utc_time*	A mensagem de evento é gerada no tempo absoluto indicado pelo tempo MJD UTC. A mensagem de evento também é gerada quando o conteúdo gravado do *stream* é reproduzido referindo-se ao tempo de reprodução
0x02	NPT	A mensagem de evento é gerada no tempo específico com os dados de tempo NPT
0x03	*eventRelativeTime*	A mensagem de evento é gerada quando o período é especificado neste campo (em milésimos de segundo) após o tempo de início do programa
0x04	---	Reservado para o futuro
0x05	*MJD_UTC_time*	A mensagem de evento é gerada no tempo absoluto indicado pelo tempo MJD UTC. Quando o conteúdo gravado do *stream* é reproduzido, a mensagem de evento também é gerada referindo-se ao tempo no ar
0x06-0xFF	---	Reservado para o futuro

— ***event_msg_MJD_UTC_time***: campo de 40 bits é codificado no caso do modo de tempo = 0 x 01 ou 0x05 e indica o tempo quando o evento de mensagem é gerado no UTC e MJD (refere-se a ARIB STD-B10:2008, Parte 2, Anexo C). Este campo contém uma cópia dos 16 bits mais baixos de MJD e é seguido por seis representações de 4 bits codificado em binário e com representação decimal (BCD);

— ***event_msg_NPT***: campo de 33 bits que é codificado no caso do modo de tempo = 0 x 02 e indica o tempo usando o *Normal Play Time* de DSM-CC (ver 7.1), quando o evento de mensagem é gerado;

— ***event_msg_relativeTime***: campo de 36 bits que é codificado no caso do modo de tempo = 0 x 03 e indica que o evento de mensagem é gerado quando o período especificado neste campo passa após o horário de início do programa. O valor deste campo é descrito na ordem de hora (dois dígitos), minuto (dois dígitos), segundo (dois dígitos) e milésimos de segundo (três dígitos), para formar nove representações de 4 bits decimais codificados em binário (BCD);

— ***event_msg_type*** **(tipo de mensagem de evento)**: um identificador que indica o tipo de mensagem de evento. O uso e a semântica são especificados em cada especificação de codificação de dados.

— ***event_msg_id*** **(identificador de mensagem de evento)**: campo de 16 bits que contém o identificador para identificar cada mensagem de evento. O uso e a semântica são especificados em cada especificação de codificação de dados;

— ***private_data_byte* (dados privados)**: campo de 8 bits que armazena informação relacionada ao evento de mensagem requerido pela especificação de codificação de dados especificado no *event_msg_type*.

13.6 Sintaxe de seção de DSM-CC transmitindo o descritor de *stream*

O descritor de *stream* é transmitido na seção DSM-CC mostrada na Tabela 77.

Tabela 77 – Seção DSM-CC (transmissão de descritor de *stream*)

Sintaxe	Número de bits	Mnemônico
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*
iIf(table_id == 0x3D){		
for(i=0;i<N;i++){		
stream_descriptor()		
}		
}		
if(section_syntax_indicator == '0'){		
Checksum	32	*uimsbf*
}		
Else{		
CRC_32	32	*rpchof*
}		
}		

A descrição da *DSMCC_section ()* deve ser a seguinte:

— ***table_id*** (tabela de identificação): campo de 8 bits que é regulado para 0x3D, para indicar que o descritor de *stream* está armazenado no *payload* da seção DSM-CC;

— ***section_syntax_indicator***: campo de 1 bit que indica que CRC32 existe no fim da seção quando é 1. Quando ele é 0, isto indica que existe verificação de soma. Para transmitir um evento de mensagem, este campo deve ser regulado em 1;

— ***private_indicator***: campo de 1 bit que armazena o valor complementar da seção de valor indicador de sintaxe;

— ***dsmcc_section_length***: campo de 12 bits que indica o comprimento de byte da área de posição imediatamente seguinte a este campo ao fim da seção. Este valor de campo não pode exceder 4 093;

— ***data_event_id***: campo de 4 bits que é o identificador para identificar o evento de dados entre os dados precedentes e seguintes que usam o evento de mensagem para permitir o conteúdo local pretendido, apesar de serem precedidos e/ou seguidos de outros conteúdos locais, para receber o evento de mensagem. Conteúdos locais adjacentes, quando transmitindo, são alocados com diferentes identificadores;

— ***event_msg_group_id*** (identificador de grupos de mensagem): campo de 12 bits que contém o campo para identificar o evento de mensagens a serem recebidas pela aplicação. A semântica detalhada é especificada em cada especificação de codificação de dados;

— ***version_number***: campo de 5 bits que é o número da versão da subtabela. O número de versão é incrementado por 1 quando qualquer peça de informação na subtabela foi trocada. Os valores disponíveis são de 0 a 31. O valor 0 é usado para atualizar o 31;

— ***current_next_indicator***: indicador de 1 bit que indica que a subtabela é a atual subtabela quando for '1'. Quando for '0', a subtabela enviada não é ainda aplicada e usada como a próxima subtabela;

— ***section_number***: campo de 8 bits que indica o número da seção;

— ***last_section_number***: campo de 8 bits que indica o número da última seção (que é a seção com o número máximo de seções da subtabela para a qual as seções participantes pertencem).

14 Sistema de arquivo de difusão e transporte de gatilho

O sistema de arquivo de difusão e transporte de gatilho deve estar de acordo com a GEM 1.3:2011, Anexo B.

Anexo A
(normativo)

Vídeo e áudio PES

A.1 Formato de transmissão de dados através do PES de vídeo MPEG-2 codificado

No caso do uso de PES de vídeo codificado com o vídeo MPEG-2 (ver ISO/IEC 13818-2) para transmitir dados, deve ser utilizado o campo de dados do usuário (*user_data_area*), conforme o cabeçalho da tela do *stream* de vídeo. A sintaxe do campo de dados do usuário é mostrada na Tabela A.1. O uso mais detalhado desta área depende do modo de operação das emissoras.

Tabela A.1 – Sintaxe do campo de dados do usuário do *stream* de vídeo

Sintaxe	Número de bits	Mnemônico
User Data () {		
user_data_start_code	32	*bslbf*
while (nextbits() != 0x000001){		
user_data	8	*uimsbf*
}		
next_start_code()		
}		
NOTA Código de início de dados de usuários: 0x000001b2.		

A.2 Formato de transmissão de dados do áudio PES codificado com MPEG-2 BC áudio

Na utilização de pacotes PES de áudio MPEG-2 BC áudio (ver ISO/IEC 13818-3) para transmissão de dados, a área de dados hierárquicos (*ancillary data area*), que pode conter outros dados que não sejam áudio MPEG, deve ser utilizada. A sintaxe da área de dados hierárquicos é mostrada na Tabela A.2. Um uso mais detalhado dessa área depende dos operadores de serviço.

Tabela A.2 – Área de dados subordinada ao *stream* de áudio

Sintaxe	Número de bits	Mnemônico
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 transmissão de dados do áudio PES codificado com MPEG-2 AAC áudio

Na utilização de pacotes PES de áudio MPEG-2 AAC áudio (ver ISO/IEC 13818-3) para transmissão de dados, a área data_*stream_element* deve ser usada para permitir que outros dados, além de dados de áudio MPEG, possam estar contidos em cada bloco de dados puro (*raw_data_block*). A sintaxe da área é mostrada na Tabela A.3. Um uso mais detalhado dessa área depende dos operadores de serviço.

Tabela A.3 – Área base de dados de *stream* do áudio *stream* (MPEG-2 AAC áudio)

Sintaxe	Número de bits	Mnemônico
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)

Informação PSI/SI para transmissão de carrosséis de dados e mensagens de eventos

B.1 Especificação da codificação de dados baseada no carrossel de dados e esquema de evento de mensagem

Em adição à especificação de codificação de dados aplicada, a transmissão de dados baseada no carrossel de dados e o esquema de evento de mensagem serão definidos; uma sintaxe adicional depende do formato de transmissão de dados a ser inserida no *data_component_descriptor* em PMT e *data_content_descriptor* em EIT especificados na ARIB STD-B23.

Esta Norma está baseada nas seguintes suposições sobre operação de transmissão para serviços de difusão de dados, conforme a seguir:

— o DII e o DDB pertencentes a um carrossel são transmitidos em um ES;

— um serviço de difusão de dados pode consistir em dois ou mais carrosséis. Eventos de mensagens podem ser transmitidos.

B.2 Conteúdo de enlace de *additional_data component_info* e *data_component_descriptor*

Para inserir a informação para controle de recepção do carrossel de dados e o possível evento de mensagens no enlace que contém *additional_data_component_info* no fim de *data_component_descriptor*, a seguinte estrutura de dados deve ser colocada no enlace, conforme determinado pela especificação de codificação de dados (ver Tabela B.1).

Tabela B.1 – *Additional ginga carousel info*

Sintaxe	Número de bits	Mnemônico
additional_ginga_carousel_info() {		
data_event_id	4	*uimsbf*
event_section_flag	1	*bslbf*
Reserved	3	*bslbf*
}		

A descrição da *additional_ginga_carousel_info()* deve ser a seguinte:

— ***data_event_id***: identificador de 4 bits que reconhece os eventos de dados precedentes e seguintes, usando o carrossel de dados e mensagens do possível evento para evitar defeito para receber o conteúdo local apropriado transmitido no carrossel de dados e mensagens de possíveis eventos. No caso de todos os bits deste campo regulados em 1, isso significa que os DIIs têm um identificador de *data_event_id* entregue neste serviço e o evento de mensagens é válido;

— ***event_section_flag***: campo de 1 bit indica se um evento de mensagens foi ou não enviado com este componente, conforme a seguir:

0: mensagens do evento não foram transmitidas;

1: mensagens do evento foram transmitidas;

— ***reserved***: reservado.

B.3 *Byte seletor* de *data_contents_descriptor*

B.3.1 *Data structure*

Para inserir informação para controle de recepção de carrossel de dados no byte seletor de descritor de conteúdos de dados como EIT, uma estrutura de dados deve ser colocada no campo *selector_byte*, conforme determinado pela especificação de codificação de dados participante.

B.3.2 Estrutura de dados para controle de recepção de carrossel de dados para serviços de dados não armazenados

Para serviços de dados não armazenados, inserir as informações contidas na Tabela B.2.

Tabela B.2 – Serviços de dados não armazenados

Sintaxe	Número de bits	Mnemônico
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*
}		

Tabela B.2 (continuação)

Sintaxe	Número de bits	Mnemônico
if(default_leak_rate_flag == '1'){		
leak_rate	22	*uimsbf*
Reserved	2	*bslbf*
}		
}		
}		

A descrição da *ginga_carousel_info()* deve ser a seguinte:

— ***num_of carousels***: campo de 8 bits que indica o número de carrosséis, incluídos no enlace seguinte;

— ***component_tag***: campo de 8 bits que designa o componente do *stream* transmitindo os carrosséis com o componente de *tag* dado pelo descritor identificador de *stream* no PMT;

— ***event_section_flag***: campo que indica se o evento de mensagens foi ou não enviado usando este componente;

— ***component_size_flag***: campo de 1 bit que indica se a estrutura de dados contém ou não o componente de tamanho. Quando o valor do campo *component_size* não está disponível, ele deve ser regulado a "0":

0: não codificado;

1: codificado;

— ***default_transaction_id_flag***: campo de 1 bit que indica se o identificador de transação está codificado ou não nesta sintaxe. Para designação de maneira a obter tudo da identificação de transação opcional, identificação de transação não pode ser codificada (0: não codificado; 1: codificado);

— ***default_timeout DII_flag***: campo de 1 bit que indica se o valor de estouro de tempo de DII está codificado nesta sintaxe. Quando o valor-padrão especificado nesta operação como valor de estouro de DII for usado, ele não está codificado (0: não codificado; 1: codificado);

— ***default_leak_rate_flag***: campo de 1 bit que indica se a taxa de vazamento está codificada ou não nesta sintaxe. Quando o valor-padrão especificado nesta operação como valor de estouro de DII for usado, ele não está codificado (0: não codificado; 1: codificado);

— ***component_size***: campo de 32 bits que indica o tamanho total (em bytes) dos dados transmitidos nos carrosséis deste componente;

— ***transaction_id***: identificação do valor da transação DII transmitida neste componente. No casa da identificação de transação não estar presente, uma DII com a identificação de transação deve ser obtida;

— ***time_out_value_DII***: campo de 32 bits que indica o valor de intervalo recomendado (em milésimos de segundo) para receber a seção completa de DII deste carrossel. Quando o valor for 0 x FFFFFFFF, isso significa que não existe valor de intervalo recomendado;

— ***leak_rate***: campo de 22 bits que indica a taxa de dispersão do transporte de *buffer* da unidade do receptor em uma unidade de 50 bytes/s.

B.3.3 Estrutura de dados para o controle da recepção do carrossel de dados para o serviço de dados armazenados

Para o serviço de dados armazenados, inserir as informações contidas na Tabela B.3.

Tabela B.3 – Serviço de dados armazenados

Sintaxe	Número de bits	Mnemônico
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*
}		
}		

A descrição da *ginga_stored_carousel_info ()* deve ser a seguinte:

— ***num_of carousels***: campo de 8 bits que indica o número de carrosséis, incluindo o do enlace seguinte;

— ***component_tag***: campo de 8 bits que designa o componente de *stream* transmitindo os carrosséis com o componente da *tag* dado pelo descritor identificador de *stream* em PMT;

— ***um_dataEvent_flag***: campo de 1 bit que indica se a estrutura de dados contém ou não um número de eventos de dados. Quando o valor do campo *num_dataEvent* não estiver disponível, ele deve ser regulado a 0 (0: não codificado; 1: codificado);

— ***num_modules_flag***: campo de 1 bit que indica se a estrutura de dados contém ou não um número total de módulos. Quando o valor do campo *num_modules* não estiver disponível, ele deve ser regulado a 0 (0: não codificado; 1: codificado);

— ***num_resources_flag***: campo de 1 bit que indica se a estrutura de dados contém ou não um número total de recursos. Quando o valor do campo *num_resources* não estiver disponível, ele deve ser regulado a 0 (0: não codificado; 1: codificado);

— ***compressed_component_size_flag***: campo de 1 bit que indica se a estrutura de dados contém ou não o tamanho comprimido do componente. Quando o valor do campo *compressed_component_size* não estiver disponível, ele deve ser regulado a 0 (0: não codificado; 1: codificado);

— ***component_size_flag***: campo de 1 bit que indica se a estrutura de dados contém ou não o tamanho do componente. Quando o valor do campo *component_size* não estiver disponível, ele deve ser regulado a 0 (0: não codificado; 1: codificado);

— ***default_transaction_id_flag***: campo de 1 bit que indica se o identificador da transação está codificado ou não na sintaxe. Para designar o ganho de DII de identificação de transação opcional, a identificação da transação não pode estar codificada (0: não codificado; 1: codificado);

— ***default_timeout DII_flag***: campo de 1 bit que indica se o valor de intervalo está ou não codificado na sintaxe. Quando o valor básico especificado na operação conforme o valor de intervalo DII for usado, ele não é codificado (0: não codificado; 1: codificado);

— ***default_leak_rate_flag***: campo de 1 bit que indica se a taxa de dispersão está ou não codificada na sintaxe. Quando o valor básico especificado na operação como o valor da taxa de dispersão for usado, ele não é codificado (0: não codificado; 1: codificado);

— ***num_dataEvent***: campo de 32 bits que indica o número total de eventos de dados no componente participante;

— ***num_modules***: campo de 32 bits que indica o número total de módulos nos eventos de dados no componente participante;

— ***num_resources***: campo de 32 bits que indica o número total de recursos nos eventos de dados no componente participante;

— ***compressed_component_size***: campo de 32 bits que indica o tamanho total (em bytes) dos dados nos eventos de dados nos carrosséis de dados deste componente. O tamanho do módulo comprimido é calculado com base no estado comprimido, não no estado extraído;

— ***component_size***: campo de 32 bits que indica o tamanho total (em bytes) dos dados nos eventos de dados nos carrosséis de dados deste componente. O tamanho do módulo comprimido é calculado com base no estado extraído, não no estado comprimido;

— ***transaction_id***: transação de identificação do valor DII transmitido neste componente. No caso da identificação da transação não estar definida, a DII com a identificação da transação deve ser obtida;

— ***time_out_value_DII***: campo de 32 bits que indica o valor de intervalo recomendado (em milésimos de segundos) para receber a seção completa de DII neste carrossel. Quando o valor for 0xFFFFFFFF, isto significa que não existe valor de intervalo recomendado;

— ***leak_rate***: campo de 22 bits que indica a taxa de dispersão do transporte de *buffer* da unidade do receptor em uma unidade de 50 byte/s.

Anexo C
(informativo)

Relação entre o descritor PMT/EIT e AIT

A Figura C.1 mostra a relação de AIT ao descritor componente de dados PMT e o descritor de conteúdos de dados EIT em Ginga.

Os itens de extensão para Ginga são os seguintes:

— nova atribuição para a identificação de valor a ser usado em Ginga (por exemplo, componentes de dados ID, *transport_id, application_id*);

— para o componente ES que transmite a aplicação Ginga ou para o componente ES que transmite AIT, as áreas do seletor são especificadas para os descritores de componentes de dados que armazenam informação adicional a não ser transmitida através de AIT;

— áreas de seletor são especificadas para os descritores de conteúdos de dados em ordem para armazenar os detalhes da aplicação Ginga na base o evento do programa;

— em um descritor, *application_identifier_flag* é regulada em vez de *entry_point_flag* em ordem para armazenar informação especificando uma aplicação que é um ponto de entrada em PMT/EIT.

As informações adicionais são para as áreas do seletor dos descritores de componentes de dados, assim como descritores de conteúdos de dados relevantes para transmissão Ginga e AIT. O restante supõe-se que esteja conforme a transmissão AIT sob MHP1.0.

É possível que o carrossel de dados que não transmite aplicação Ginga refira-se aos conteúdos especificando o locador da aplicação Ginga.

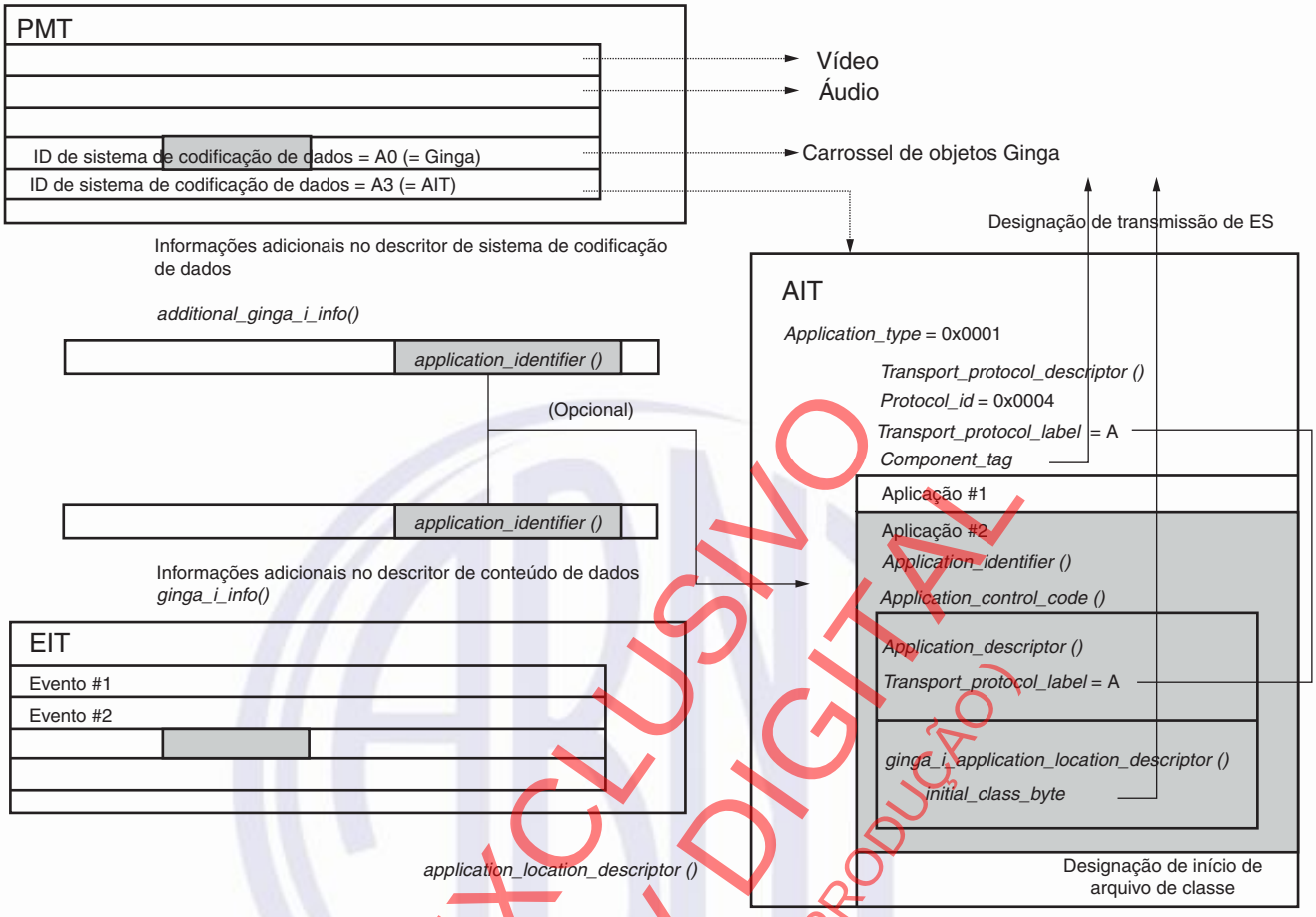


Figura C.1 – Relação da transmissão AIT e descritor de componente de dados

Anexo D
(informativo)

Informações adicionais sobre trasmissões utilizando independentes PES

Um *data_identifier* deve estar presente no início de *PES_packet_data_byte* para identificar o tipo de dados (ver ETSI EN 301 192, ATSC DVS 161 e DAVIC 1.4:1998, parte 9).

NOTA No Japão, o Ministério de Telecomunicações (Notificação Hei 10/260) estipula o uso de *data_component_id* na PMT.

Para assegurar a conformidade DVB, ATSC e DAVIC, o campo *data_identifier* contém uma cópia do valor alocado para a área do usuário definido naqueles padrões.

As razões para empregar as especificações independentes PES como um padrão para o método de transmissão PES são as seguintes:

— tamanho (menor de restrições) permite uma liberdade maior;

— é permitido que dados de vídeo e áudio sejam produzidos separadamente antes de serem multiplexados por menos esforço;

— o compartilhamento de dados é permitido em múltiplas peças de dados de vídeo e áudio para um acesso mais fácil.

A transmissão de carrossel de dados é baseada no *downloading* U-N (carrossel de dados DSM-CC) estipulado na ISO/IEC 13818-6, no qual o seguinte foi adicionado:

— em relação à área de dados do módulo, assumindo seu uso para a transmissão de arquivos etc., foram adicionados os descritores de *Activation time*, de *Expire* e de *CompressionType*.

— exceto pelos serviços de *download* assumidos quando o original ISO/IEC 13818-6 foi desenvolvido, estas estipulações permitem o uso das transmissões do carrossel de dados que são eficientes e que têm a recepção mínima processando cargas em uma ampla variedade de aplicações, como serviços de multimídia.

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 PUB 140-1:1994, http://www-09.nist.gov/div897/pubs/fip140-1.htm, *Security requirements for cryptographic modules*

[14] FIPS PUB 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*

[22] IEEE 802-2001, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*