BRAZILIAN STANDARD

ABNT NBR 15606-3

First edition
2007.11.30

Valid from
2007.12.01

Corrected version
2008.08.22

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

Palavras-chave: Digital terrestrial television. Digital broadcasting. Data transmission.
Descriptors: Digital terrestrial television. Digital broadcasting. Data transmission.

ICS 33.160.01

ISBN 978-85-07-00615-2

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Reference number
ABNT NBR 15606-3:2007
79 pages

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

Impresso no Brasil

Contents

Page

Foreword

Associação Brasileira de Normas Técnicas (ABNT) is the Brazilian Standardization Forum. Brazilian Standards, which content is responsability of the Brazilian Committees (Comitês Brasileiros – ABNT/CB), Sectorial Standardization Bodies (Organismos de Normalização Setorial – ABNT/ONS), and Special Studies Committees (Comissões de Estudo Especiais – ABNT/CEE), are prepared by Study Committees (Comissões de Estudo – CE), made up of representants from the sectors involved including: producers, consumers and neutral entities (universities, laboratories, and others).

Brazilian Standards are drafted in accordance with the rules given in the ABNT Directives (Diretivas), Part 2.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ABNT shall not be held responsible for identifying any or all such patent rights.

ABNT NBR 15606-3 was prepared within the purview of the Special Studies Committees of Digital Television (ABNT/CEE-00:001.85). The Draft Standard was circulated for National Consultation in accordance with ABNT Notice (Edital) nº 07, from June 29, 2007 to August 28, 2007 with the number Draft 00:001.85-006/3.

Should any doubts arise regarding the interpretation of the English version, the provisions in the original text in Portuguese shall prevail at all time.

This standard is based on the work of the Brazilian Digital Television Forum as established by the Presidential Decree number 5.820 of June, 29th 2006.

ABNT NBR 15606 consists of the following parts, under the general title "*Digital terrestrial television — Data coding and transmission specifications for digital broadcasting*":

— Part 1: Data coding specification;

— Part 2: Ginga-NCL for fixed and mobile receivers – XML application language for application coding;

— Part 3: Data transmission specification;

— Part 4: Ginga-J – The environment for the execution of procedural applications;

Part 5: Ginga-NCL for portable receivers – XML application language for application coding.

This English version is equivalent to the corrected version 2 of ABNT NBR 15603-1:2007, from 2008.08.22.

This corrected version of ABNT NBR 15603-1:2007 includes the Technical Corrigendum 1 from 2008.08.22.

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

1 Scope

This part of ABNT NBR 15606 provides a data transmission specification for the data broadcasting scheme, part of the digital broadcasting scheme specified as the standard in Brasil.

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

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ABNT NBR 15603-1, *Digital terrestrial television – Multiplexing and service information (SI) – Part 1: SI for digital broadcasting systems*

ABNT NBR 15603-2:2007, *Digital terrestrial television – Multiplexing and service information (SI) – Part 2: Data structure and definition of basic information of SI*

ABNT NBR 15603-3, *Digital terrestrial television – Multiplexing and service information (SI) – Part 3: Syntaxe and definition of extension information of SI*

ABNT NBR 15606-1, *Digital terrestrial television – Data coding and transmission specifications for digital broadcasting - Part 1: Data coding specification*

ABNT NBR 15606-2:2007, *Digital terrestrial television – Data coding and transmission specifications for digital broadcasting – Part 2: Ginga-NCL for fixed and mobile receivers – XML application language for application coding*

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

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

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

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

ISO/IEC 13818-1:2000, *Information technology – Generic coding of moving pictures and associated audio information: Systems*

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

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

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

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

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

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

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

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

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

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

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

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

RFC 791, DARPA *Internet program protocol specification*

RFC 2396, *URI Generic Syntax*

RFC 1112, *Host extensions for IP multicasting*

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

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

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

3 Terms and definitions

For the purposes of this part of ABNT NBR 15606, apply to the following terms and definitions.

3.1
multi carrier code division multiple access
MC-CDMA
a Code Division Multiple Access (CDMA) scheme that employs multi carrier systems

3.2
transport buffer
a buffer that decodes a transport stream packet in a target decoder of a MPEG2 transport stream Buffer transport Buffer that decodes a package of transport in a stream decoder target of a stream of transport MPEG2

3.3
PES packet header
a field that comprises the first part of a PES packet

3.4
data carousel
a method that sends out any set of data repeatedly so that the data can be downloaded via broadcasting in any timing as needed

NOTE This method is specified in ISO/IEC13818-6.

3.5
digital storage media command and control
DMS-CC
this method supports accessing files and streams in digital interactive service

3.6
advanced television system committee
ATSC
a committee for the purpose of digital broadcasting system standardization in the USA

3.7
digital audio visual council
DAVIC
an industrial consortium for interactive multimedia service standardization

3.8
content
a group of data transmitted through a data broadcasting program or a bidirectional communications service that is caused by the data broadcasting program to serve as part of the data broadcasting program

NOTE More specifically as a term in the broadcasting context, “content” indicates a set of streams in the program sending the group of data. A single data broadcasting program can be composed of multiple contents.

3.9
local content
of the content of transmission of data contained in a single event data

NOTE Generally, a local content is a group of pooled data based on context or to greater convenience of generating program.

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

3.11
address
protocol used to establish a name server using PPP

NOTE This definition this according to RFC 1877.

3.12
ethernet
LAN standard that defines a network based on bus employing the CSMA / CD (multiple of access to carrier/collision detection) for access control

NOTE This definition is in line with IEEE 802.

3.13
event data
set of streams of data transmission which represents a group of content for transmission of data to be distributed with the start and end times of pre-configured

NOTE The concept of event data (data event) is introduced to allow a group of content, data transmission is rotated to another, or if they are not in the same program as needed. In other words, an event data is independent of an event.

3.14
format adaptation
format used in a header DSM-CC and is a form of information inserted in an area of adjustment that encodes the information to meet a request, depending on the distribution system

3.15
hash function
function wrapping
Mathematical function that maps a large area (huge in some cases) within a small range

NOTE A hash function is unique and hand-free collision.

3.16
host
machine
device to access point or device server, required for bidirectional transmission services

3.17
hypertext transfer protocol
HTTP
Layer of application pra transmitted via the World Wide Web

NOTE This definition is in accordance with RFC 1954.

3.18
identifier package
PID
identifier of a package of transport stream MPEG-2

3.19
information service
SI
digital data that describe a array of programs, a distribution system for transmission of streams of data, descriptions of programs, information-grade programming/time of duration

NOTE These data also transporting MPEG-2 PSI (Information Specific Program), as well as parts of the extension set independently.

3.20
specific information program
PSI
control of information transmission, which provides the information necessary to enable a receiver automatically demultiplexar and decode multiple streams of program that were multipleksne

3.21
MIME
layer protocol for the application that provides an architecture of content that allows data such as multimedia files of text, audio and pictures, as that is not US-ASCII to be transmitted via e-mail

3.22
PES package
the data format used to transmit streams basic consisting of a header package PES and a payload PES immediately following the header

3.23
private_stream_1
kind of stream transmitted using PES, which used to send a stream private synchronized with other streams

3.24
private_stream_2
kind of stream transmitted using PES, which is used to send a private stream that does not need to be synchronized with other streams

3.25
procedure for the control of a data connection, the high-level
HDLC procedure
Procedure for the control of transmission with high reliability, used for communication between computers, especially in LAN and internet

3.26
user datagram protocol
UDP
transport layer protocol, which promotes delivery of data without connection between two machines

NOTE 1 Although the UDP does not support messages of recognition, it minimizes the high protocol for greater efficiency in transmission services.

NOTE 2 This definition this according to the RFC 768.

3.27
protocol for authentication, password NNTP
layer protocol for the application used to distribute, post and retrieve news (Net News) on the internet

NOTE This definition is according to RFC 1334.

3.28
protocol for authentication, password PAP
Component of the point-to-point protocol (PPP) which supports authentication

NOTE 1 This protocol makes no identification and password to send.

NOTE 2 This definition this according to RFC 1334.

3.29
protocol for the control of transmission
TCP
protocol for transport layer that promotes distribution of data highly reliable, end of the tip, guided by connection, using a mechanism for the detection and correction of errors

NOTE This definition this according to the RFC 793.

3.30
IP protocol control
IPCP
protocol used to establish multiple settings required to use IP in the phase of protocol-layer network

NOTE This definition this according to the RFC 1332.

3.31
internet protocol
IP
protocol-layer network that defines the mechanism for addressing on the Internet to enable data to be transmitted

NOTE This definition this according to the RFC 791.

3.32
protocol for resolution of address
ARP
protocol used in a TCP / IP to get the physical address of the node Ethernet based on your IP address

3.33
protocol for the transmission of data in simple mode
of communications protocol developed for basic transmission of data between a host and a terminal

NOTE The protocol employs a method to minimize the errors of transmission.

3.34
integrated services digital network
ISDN
Integrated services digital network

3.35
booked
term that, when used in sentences that define the encrypted bit stream of shows that the value can be used in the future for extensions defined by ISO

NOTE The reserved bits are set to 1.

3.36
reserved_future_use
term that, when used in sentences setting the encrypted bit stream of shows that the value may be used in extensions defined by ISO in the future

NOTE The reserved bits are set to 1.

3.37
section
syntactic structure used to map the information service and other data within a packet stream of transport

3.38
service domain name
DNS
protocol used by the service which maps a name from machine to a network within your IP address

NOTE This definition is in accordance with RFC 1034 and RFC 1035.

3.40
table of mapping program
PMT
table that is part of the PSI

NOTE This table indicates a location (PID package of transport stream), a table of mapping program for each service in miltiplexado stream.

3.41
transmission of digital video
DVB
project to standardize the system of digital transmission in Europe

4 Types of data transmission specification

The types of the data transmission specifications and the stream type identifications contained in a PMT are shown in Table 1.

Table 1 — Transmission specification types

Transmission specification	Major functions and usages	Stream type
Independent PES	Used for streaming synchronous and assynchronous data for broadcasting services	0x06
Object and Data carousel	Used for streaming synchronous and asynchronous data for broadcasting services. Applied to data transmission for download services and multimedia services	0x0B, 0X0D [a]
Event message	Used for synchronous and asynchronous message notification to na application on the receiver unit from the broadcasting station. Used in multimedia services	0x0C, 0X0D [b]
Interaction channel protocols	Transmission protocols used in a fixed network such as a PSTN/ISDN network and a mobile phone/PHS network when bidirectional communication is also used in a broadcasting service [d]	—
Multiprotocol encapsulation	Datagrams are encapsulated in datagram_sections which are compatible with the DSMCC_section format for the private data	0x0A [c]
Data piping	Protocol which allow data insertion from a broadcasting network directly in the MPEG-2 packet payload	0x7E

[a] When a stream don't contain any data type different of the data carousel, 0x0B or 0x0D are used and when it contains other data DSM-CC, 0x0D is used.

[b] When a stream don't contain any data type different of a event message, 0x0C or 0x0D are used and when it also contains other data DSM-CC, 0x0D is used.

[c] When a stream don't contain any data type different of *multiprotocol encapsulation* (MPE), 0x0A is used and when it also contains other DSM-CC data, 0x0D is used.

[d] PSTN: Public switched telephone network

5 Data carousel transmission specification

5.1 Transmission with DSM-CC data carousel

The data carousel transmission specification is designed to implement general synchronized or asynchronous data transmission without a need of streaming data such as data download to a receiver unit or content transmission for multimedia services. The data carousel transmission specification defined in this standard is based on the DSM-CC data carousel specification specified in ISO/IEC 13818-6.

Transmitting data repeatedly, as defined in the DSM-CC data carousel specification, allows a receiver unit to obtain data on demand at anytime during a transmission period.

Data is transmitted in a module unit that consists of blocks, which all blocks other than that at the end of the module have the same size and each block is transmitted in sections.

This data transmission are used the download data block message (hereafter referred to as DDB message) and the download info indication message (hereafter referred to as DII message). The two messages are components of the User-to-Network download protocol specified in ISO/IEC 13818-6. The data body is transmitted by the DDB message with each module divided into blocks. For additional information related to PSI/SI trasmission see Annex B.

5.2 DSM-CC control message

5.2.1 Download info indication (DII) message

A DII message is part of a DSM-CC control message. Therefore, a DII message transmits message content by containing it in the userNetworkMessage() in the DSM-CC section.

The DII message version is indicated by transaction_number in the transaction_id field of dsmccMessageHeader. This version number is common to all DII messages of the data carousel and the version number is incremented by one when content of one or more DII messages have been changed.

5.2.2 Syntax and semantics of DII message

The data structure of a DII message is shown in Table 2.

Table 2 — Data structure of download infoindication message

Syntax	Number of bits	Mnemonic
DownloadInfoIndication() {		
dsmccMessageHeader()		
downloadId	32	uimsbf
blockSize	16	uimsbf
windowSize	8	uimsbf
ackPeriod	8	uimsbf
tCDownloadWindow	32	uimsbf
tCDownloadScenario	32	uimsbf
compatibilityDescriptor()		
numberOfModules	16	uimsbf
for(i=0;i< numberOfModules;i++) {		
moduleId	16	uimsbf
moduleSize	32	uimsbf
moduleVersion	8	uimsbf
moduleInfoLength	8	uimsbf
for(i=0;i< moduleInfoLength;i++){		
moduleInfoByte	8	uimsbf
}		
}		
privateDataLength	16	uimsbf
for(i=0;i<privateDataLength;i++){		
privateDataByte	8	uimsbf
}		
}		

The semantic fields of DII shall be the following:

— **dsmccMessageHeader () (DSM-CC message header):** as specified in 5.3;

— **downloadId (download identifier):** 32 bits field which serves as a label to uniquely identify the data carousel. In the case of a data event operation, data_event_id shall be inserted into bits 28-31 of downloadId. Otherwise, the range and values to ensure the uniqueness is specified in an operational standard;

— **windowSize:** 8 bits field which is not used for data carousel transmission and the value shall be set to 0;

— **ackPeriod:** 8 bits field which is not used for data carousel transmission and the value shall be set to 0;

— **tCDownloadWindow:** 32 bits field which is not used for data carousel transmission and the value shall be set to 0;

— **tCDownloadScenario:** 32 bits field which indicates the timeout period in which the download is assumed to be completed in microseconds;

— **compatibilityDescriptor():** structure of the compatibility descriptor that is specified in ISO/IEC 13818-6 and shall be configured in this field. When the content of the compatibilityDescriptor() structure is not needed, descriptorCount shall be set to 0x0000 and then the field length shall be 4 bytes;

— **numberOfModules (number of module):** 16 bits field which indicates the number of the modules described in the following loop in this DII message;

— **moduleId (module identifier):** 16 bits field which indicates the identification of the module described in the following moduleSize, moduleVersion, and moduleInfoByte fields;

— **moduleSize (module length):** 32 bits field which indicates the byte length of the module. When the byte length of the module is not known, it shall be set to 0;

— **moduleVersion:** 8 bits field indicates the version of this module;

— **moduleInfoLength (module information length):** 8 bits field which indicates the byte length of the following module information area;

— **moduleInfoByte (module information):** 8 bits unit field which may be used to insert descriptors related to the module. These descriptors are defined in 5.4;

NOTE The tag values of the descriptors to be inserted are defined in Table 5.

— **privateDataLength:** 16 bits field which indicates the byte length of the following PrivateDataByte field;

— **privateDataByte (private data):** 8 bits unit field which may be used to contain a data structure in a descriptor format. The data structure is defined based on a data coding format or by a service operator;

The semantics of the tag values of the descriptors for this field is defined in Table 3. The possible descriptors for this field are those defined in 5.4 and by a data coding format.

Table 3 — Semantics of descriptor tags of module information area and private area in DII

Value of descriptor tag	Semantics
0x01 - 0x7F	Reserved tag values of DVB-compatible descriptors to be inserted into the module information area and the private area (see 5.4)
0x80 - 0xBF	Available tag values of descriptors defined by a service operator
0xC0 - 0xEF	Reserved for tag values of descriptors to be inserted into the module information area and the private area (see 5.4)
0xF0 - 0xFE	Reserved tag values of descriptors defined base don a data coding format

5.3 Syntax and semantics of dsmccMessageHeader()

The data structure of dsmccMessageHeader() is defined in Table 4.

Table 4 — Data structrure of dsmccMesageHeader

Syntax	Number of bits	Mnemonic
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
}		
}		

The semantic fields of dsmccMessageHeader() shall be the following:

— **protocolDiscriminator:** 8 bits field which shall be set to 0x11 and indicates that this message is a MPEG-2 DSM-CC message;

— **dsmccType (DSM-CC type):** 8 bits field which indicates the type of the MPEG-2 DSM-CC message. In a DII message for data carousel transmission, it shall be set to 0x03 (U-N download message);

— **messageId (message type identifier):** 16 bits field which identifies the DSM-CC message type. In a DII message, it shall be set to 0x1002;

— **transaction_id (Transaction identifier):** 32 bits which field identifies the message and has a versioning function. The format of transaction_id is shown in Figure 1. The Transaction number field in bits 0-29 shall be used to identify the version of the DII, as specified in ISO/IEC13818-6. The value of bits 30-31 shall be set to '10' (TransactionId allocated by the network) as defined in Transaction Id Originator, as specified in ISO/IEC 13818-6;

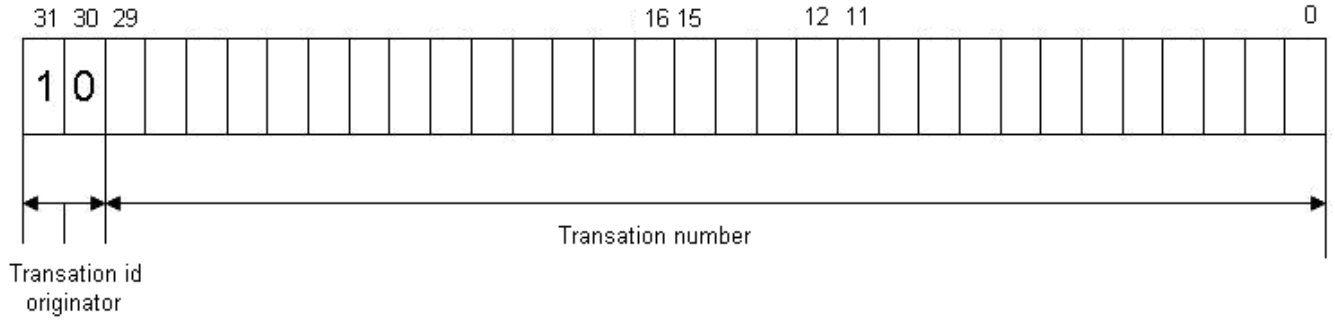

Figure 1 — Format of transaction_id

— **adaptationLength:** 8 bits field which indicates the byte number of the dsmccAdaptationHeader() field;

— **messageLength:** 16 bits field which indicates the number of bytes of the message immediately following this field. That is, the value is a sum of the payload length tand the dsmccAdaptationHeader() length;

— **dsmccAdaptationHeader():** the data structure of this field is defined in 5.5.3.

5.4 Descriptors of module information area and private area

5.4.1 Types of the descriptors

The types of the descriptors used in a module information area and a private area are shown in Table 5. Any of these descriptors may be used in the module information area and/or the private area as needed. The descriptors contained in the private area in a DII apply to the modules in the DII. When the module information area and the private are have the same set of descriptors, only the descriptors in the module information area are enabled.

Table 5 — Types of descriptors

Tag value	Descriptor	Function	Module information area	Private area
0x01	type_descriptor	Module type (MIME form etc.)	o	
0x02	name_descriptor	Module name (file name)	o	
0x03	info_descriptor	Module information (character type)	o	o
0x04	module_link_descriptor	Link information (module id)	o	
0x05	CRC32_descriptor	CRC32 of total module	o	
0x06	location_descriptor		o	o
0x07	est_download_time_descriptor	Estimated download time (s)	o	o
0x08 - 0x7F	Reserved for the future			
0x80 - 0xBF	Available to a broadcaster			
0xC0 - 0xC1	Reserved for the future			
0xC2	Compression_Type_descriptor	Compression algorithm when the module is transmitted	o	
0xC3 - 0xCC	Reserved for the future			
0xCD - 0xEE	Reserved for the future			

5.4.2 Type descriptor

The type descriptor (see Table 6) indicates the file type of the file transmitted as a single module, implementing data carousel transmission based on this Standard that specifies that a single file is transmitted as a single module.

Table 6 — Type descriptor

Syntax	Number of bits	Mnemonic
Type_descriptor(){		
descriptor_tag	8	*uimsbf*
descriptor_length	8	*uimsbf*
for(i=0;i<N;i++) {		
text_char	8	*uimsbf*
}		
}		

The semantic fields of type descriptor shall be the following:

— **text_char:** 8 bits field. The sequence of this fields indicates a media type in accordance with RFC 1521.

5.4.3 Name descriptor

The name descriptor (see Table 7) indicates the file name of the file transmitted as a single module, implementing data carousel transmission based on this standard that specifies that a single file is transmitted as a single module. However, when the Module Link descriptor exists, the name descriptor shall not be present other than in the position = 0x00 module in the DII.

Table 7 — Name descriptor

Syntax	Number of bits	Mnemonic
Name_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0;i<N;i++) {		
text_char	8	uimsbf
}		
}		

The semantic fields of name descriptor shall be the following:

— **text_char:** 8 bits field. The sequence of this field indicates the file name of the file transmitted as a single module using the data coding specification or a character code specified in an operational standard.

5.4.4 Info descriptor

The info descriptor (see Table 8) describes information related tothe module.

Table 8 — Info descriptor

Syntax	Number of bits	Mnemonic
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
}		
}		

The semantic fields of info descriptor shall be the following:

— **ISO_639_language_code:** 24 bits field which identifies the language used in the following text_char area. The language code is represented in three alphabetic characters specified in ISO 639-2. Each character is coded into a 8 bits representation according to ISO 8859-1 and inserted into the 24 bits field in that order;

— **text_char:** 8 bits field. The sequence ot this fields indicates textual information related to the file transmitted as a single module using the data coding specification or a character code specified in an operational standard.

5.4.5 Module_link descriptor

The Module_link descriptor (see Table 9) generates a list of modules by linking with other modules. Because the length of the block number field of a DDB message is restricted to 16 bits, the maximum size of one module in data carousel transmission is 256 Mbytes. When a file larger than 256 Mbytes is transmitted, the file is divided into two or more modules before being sent.

Table 9 — Module_link descriptor

Syntax	Number of bits	Mnemonic
module_link_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
position	8	uimsbf
moduleId	16	uimsbf
}		

The semantic fields of module link descriptor shall be the following:

— **position:** 8 bits field which indicates the position relation to the linked module. “0x00” indicates that it is located at the head of the link, “0x01” indicates the middle and “0x02” indicates the end;

— **module Id:** 16 bits field is the module identification of the linked module. When the position is “0x02”, the value of this field is ignored.

5.4.6 Location descriptor

The location_descriptor contains the location of the PID where blocks, modules or groups can be found containing data of the carousel. Table 10 shows the syntax of the location_descriptor.

Table 10 — Syntax of location_descriptor

Syntax	Number of bits	Valor
location_descriptor(){		
descriptor_tag	8	0x06
descriptor_length	8	
location_tag	8	
}		

The semantic fields of location descriptor shall be the following:

— **descriptor_tag:** 8 bits field which identifies the descriptor. For the location_descriptor it is set to 0x06;

— **descriptor_length:** 8 bits field which specifies the number of bytes of the descriptor immediately following this field;

— **location_tag:** 8 bits field which has the same value as the component_tag field in the stream identifier descriptor.

5.4.7 CRC descriptor

THe CRC descriptor (see Table 11) describes the CRC value of the whole module.

Table 11 — CRC descriptor

Syntax	Number of bits	Mnemonic
CRC32_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
CRC_32	32	rpchof
}		

The semantic fields of CRC descriptor shall be the following:

— **CRC_32:** 32 bits which stores the CRC value calculated against the whole module. The CRC value shall be calculated as defined in ABNT NBR 15603-2:2007, Annex B.

5.4.8 Estimated download time descriptor

The estimated download time descriptor (see Table 12) describes a estimated period require to download the module.

Table 12 — Estimated download time descriptor

Syntax	Number of bits	Mnemonic
est_download_time_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
est_download_time [a]	32	uimsbf
}		

The semantic fields of estimated download time descriptor shall be the following:

— **est_download_time:** 32 bits field indicates the estimated period (in seconds) to require to download the module.

5.4.9 Compression type descriptor

The compression type descriptor (see Table 13) indicates that the module has been compressed and tells its compression algorithm and the module size before compression in bytes. A module that has not been compressed does not have this descriptor.

Table 13 — Compression type descriptor

Syntax	Number of bits	Mnemonic
Compression_Type_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
compression_type	8	uimsbf
original_size	32	uimsbf
}		

The semantic fields of compression type descriptor shall be the following:

— **compression_type:** 8 bits field which designates the compression type used to compress the module;

— **original_size:** 32 bits field which indicates the module size before compression in bytes.

5.5 DownloadDataBlock (DDB) message

5.5.1 Syntax and semantics of DDB message

The content of a DDB message in transmitted by storing in the downloadDataMessage() field in the DSM-CC section.

A DDB message is the data structure transmitting data blocks (see Table 14). A module can be divided with a fixed length to form blocks. In this case, each block is represented with a block number in the DDB message to allow a receiver unit to rearrange the blocks in the intended order.

As is specified in ISO/IEC13818-6, when DDB messages are transmitted in MPEG-2 TS, only the DDB messages having the same downloadId shall be included in the same PID packet. That means that DDB messages in two different carousels shall not present in a single elementary stream.

Table 14 — Data structure of download data block

Syntax	Number of bits	Mnemonic
DownloadDataBlock() {		
dsmccDownloadDataHeader()		
moduleId	16	uimsbf
moduleVersion	8	uimsbf
reserved	8	bslbf
blockNumber	16	uimsbf
for(i=0;i<N;i++) {		
blockDataByte	8	uimsbf
}		
}		

The fields of DDB shall be the following:

— **moduleId:** 16 bits field which indicates the identification number of the module, to which this block belongs;

— **moduleVersion:** 8 bits field which indicates the version of the module, to which this block belongs;

— **blockNumber:** 16 bits field which indicates the position of this block within the module. The first block of a module shall be represented by the block number 0;

— **blockDataByte:** 8 bits field. The size of a series of the block data area is as same as the block size of the DII, that is, the size of the blocks divided from a module. However, the block number of the last block in the module may be smaller than the block size described in DII.

5.5.2 Syntax and semantics of dsmccDownloadDataHeader()

The data structure of the dsmccDownloadDataHeader() is defined in Table 15.

Table 15 — Data structureof the dsmccDownloadDataHeader

Syntax	Number of bits	Mnemonic
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
}		
}		

The fields of dsmcc DownloadData Header shall be the following:

— **protocol discriminator:** 8 bits field which is set to 0x11 and indicates that this message is a MPEG-2 DSM-CC message dsmccType This 8-bit field indicates the type of the MPEG-2 DSM-CC message and is set to 0x03 (U-N download message) for the DDB message in data carousel transmission;

— **messageId:** 16 bits field which identifies the type of the DSM-CC message and sets to 0x1003 for a DDB message;

— **downloadId:** 32 bits field which is set to the same value as the download identifier in the corresponding DII message;

— **adaptationLength:** 8 bits field which indicates the number of bytes of the dsmccAdaptationHeader() field;

— **dsmccAdaptationHeader():** the data structure of this field is defined in 5.5.3;

— **messageLength:** 16 bits field which indicates the length of the message, not including this field and the area preceding this field in bytes. The value is identical to the sum of the payload length and the dsmccAdaptationHeader length.

5.5.3 Syntax of dsmccAdaptationHeader()

In dsmccMessageHeader() which is the header part of a DII message and dsmccDownloadDataHeader() which is the header part of a DDB message, the common data structure dsmccAdaptationHeader() may be placed.

The data structure of dsmccAdaptationHeader is indicated in Table 16.

Table 16 — DsmccAdaptationHeader structure

Syntax	Number of bits	Mnemonic
dsmccAdaptationHeader() {		
adaptationType	8	uimsbf
for (i=0; i<(adaptationLength-1);i++) {		
adaptationDataByte	8	uimsbf
}		
}		
}		

The semantics of dsmccAdaptationHeader() shall be the following:

— **adaptationType:** 8 bits field which indicates the type of the adaptation header. This field value indicates an adaptation format as shown in Table 17.

Table 17 — Adaptation type

Adaptation type	Adaptation format	Definiton ISO/IEC 13818-6
0x00	Reserved	Same as in the left column
0x01	Reserved	DSM-CC conditional access
0x02	Reserved	DSM-CC user identifier
0x03	DIIMsgNumber	Same as in the left column
0x04-0x7F	Reserved	Same as in the left column
0x80-0xFF	User definition	Same as in the left column
NOTE For Adaptation types used on this standard the operation of the user definition adaptation format of the adaptation type 0x00 – 0xFF has the option of a operation by a service operator.		

5.5.4 Syntax of DSM-CC section

DII messages and DDB messages are transmitted using DSM-CC sections shown in Table 18.

Table 18 — DSM-CC section (transmission of DII/DBB message)

Syntax	Number of bits	Mnemonic
DSMCC_section () {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
dsmcc_section_length	12	uimsbf
table_id_extension	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf

Table 18 (continuação)

Syntax	Number of bits	Mnemonic
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
}		
}		

Semantics of DSM-CC section shall be the following:

— **table_id:** 8 bits field which contains the number identifying the type of data in the DSM-CC section payload. Based on the value in this field, a coding rule specific to the following field in the DSM-CC section is applied. The table identification values are shown in the Table 19, as specified in ISO/IEC13818-6;

Table 19 —Table_id

table_id	DSM-CC section type	Definition of ISO/IEC13818-6
0x3A	Reserved	Multi-protocol capsule [a]
0x3B	DII message	U-N message including DIII
0x3C	DDB message	Same as in the left column
0x3D	Stream descriptor	Same as in the left column
0x3E	Private data	Same as in the left column
0x3F	Reserved	Same as in the left column
[a] See ISO/IEC13818-6.		

— **section_syntax_indicator:** 1 bit field. When it is set to 1, it indicates that a CRC32 exists at the end of the section. When it is set to ***0, it indicates that check sum exists. It shall be set to 1 to transmito DII messages and DDB messages;

— **private_indicator:** 1 bit field which stores the complement value of the section_syntax_indicator flag;

— **dsmcc_section_length:** 12 bits field which indicates the number of bytes of the area from the beginning the field immediately following this field to the end of the section. The value in this field shall not exceed 4093 bytes;

— **table_id_extension:** 16 bits field which is set as shown below according to table_id field:

  - when the value of the table_id field equals 0x3B, this field shall convey a copy of the least significant two bytes of the transaction_id field;
  - when the value of the table_id field equals 0x3C, this field shall convey a copy of the module_id field;

— **version_number:** 5 bits field is set as shown below according to table identifie;

— **value:** when the value of tabel_id field equals 0x3B, this field shall be set to 02. When the value of table_id field equals 0x3C, shall be set to the least significant 5 bit of the module version field;

— **current_next_indicator:** 1 bit designation which indicates that the sub-table is the current sub-table when it is "1". When it is "0", the sent sub-table is not applied yet and used as the next sub-table. When the value of table_id field equals a value in the range from 0x3A to 0x3C, this field shall be set to "1";

— **section_number:** 8 bits field which indicates the section number of the first section in the sub-table. When the section contains a DII message, this filed shall set to be 0. When this section contains a DDB message, this ield shall convey a copy of the least significant eight bits of the block number of the DDB;

— **last_section_number:** 8 bits field which indicates the last section number (the section which has the maximum section number) of the sub-table to which the section belongs;

— **userNetworkMessage():** the DII message is stored;

— **downloadDataMessage():** the DDB message is stored.

6 Especification of object carousel

6.1 Scope of object carousels

The object carousel specification has been added in order to support data broadcast services that require the periodic broadcasting of DSM-CC U-U objects through SBTVD compliant broadcast networks.

Data broadcast according to the SBTVD object carousel specification is transmitted according to the DSM-CC object carousel and DSM-CC data carousel specification which are defined in MPEG-2 DSM-CC (see ISO/IEC13818-6:1998, Section 5).

6.2 Data transport specification

6.2.1 Carousel NSAP address

The specification of SBTVD object carousels is based on the DSM-CC object carousel specification (see ISO/IEC 13818-6). A SBTVD object carousel represents a particular service domain that consists of a collection of DSM-CC U-U objects within a SBTVD network. The service domain has a service gateway that presents a graph of service and object names to receivers.

The unique identification of the service gateway in broadcast networks is done by means of carousel Network Service Access Point (NSAP) address as defined in DSM-CC (see ISO/IEC13818-6). This address contains a network specific part that shall make the address unique within the network environment used. The carousel NSAP address is used to refer to the object carousel from another service domain. For SBTVD system environments, the syntax and semantics of the carousel NSAP address are defined below.

6.2.2 Carousel NSAP address structure

The carousel NSAP address has a structure as defined in Figure 2 (see ISO/IEC13818-6):

Figure 2 — Format of carousel NSAP address

The semantics of AFI (authority and format identifier), type, carouselId, and specifier are defined in ISO/IEC 13818-6. In particular:

— **AFI:** 8 bits field which shall be set to the value of 0x00 to indicate the usage of the NSAP format for private use (ver EN 301 192);

— **Type:** 8 bits field which shall be set to 0x00 to indicate the use of the NSAP address for object carousels;

— **carouselId:** 32 bits field which shall be set to the identifier of the object carousel by the carouselId field;

— **specifier:** 32 bits field shall convey the specifierType field (set to the value of 0x01) and OUI (Organizational Unique Identifier) code as defined in DSM-CC (see ISO/IEC 13818-6:1998, Sectione 5). The OUI code shall be set to a value that has been allocated to DVB by the IEEE 802 registration authority;

— **privateData:** field which shall convey the ginga_service_location structure which is defined in Table 20.

Table 20 — Syntax for DVB_service_location structure

Syntax	Number of bits	Mnemonic
ginga_service_location() {		
transport_stream_id	16	uimsbf
org_network_id	16	uimsbf
service_id	16	uimsbf
reserved	32	bslbf
}		

The semantics of the dvb_service_location structure shall be the following:

— **transport_stream_id:** 16 bits field that identifies the transport stream on which the carousel is broadcast;

— **org_network_id:** 16 bits field that identifies the network_id of the delivery system from which the carousel originates;

— **service_id:** 16 bits field gives the service identifier of the service that contains the object carousel. The service_id is the same as the program_number in the associated program_map_section.

6.3 Descriptors

NOTE All data carousel descriptors are the same used in data carousels transmission in Section 5.

6.3.1 PSI and SI specification

The data broadcast service indicate the use of a SBTVD object carousel by including one or more data component descriptor - carousel ID descriptor – association tag descriptor, in accordance to ARIB STD-B23.

Each descriptor shall point to one object carousel and shall be associated to a particular stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468) that may be present in the PSI program map section for the stream that is used as a data stream.

Each data broadcast descriptor allows for the start up of the higher layer protocols based on a language criterion usinga list of object names.

An object carousels can be implemented using multiple data broadcast services. Data broadcast services may publish that they are part of a particular object carousel by including the carousel_identifier_descriptor as defined by DSM-CC (see ISO/IEC13818-6) in the first descriptor loop of the program map table.

Further, object carousels use the concept of taps (see ISO/IEC13818-6) to identify the streams on which the objects are broadcast. The association between Taps and the streams of the data service may be done by either using the association_tag descriptor defined in (see ISO/IEC13818-6) or the stream_identifier_descriptor in EN 300 468.

In the latter case, it is assumed that the component_tag field of the stream_identifier descriptor is the least significant byte of the referenced association_tag value which has the most significant byte set to 0x00.

Finally, stream objects within U-U object carousels can be bound to elementary streams of the data broadcasting service itself, to elementary streams of other SBTVD services, or to complete SBTVD services. If the stream object is bound to elementary streams of other SBTVD services, or to complete SBTVD services the program map table of the data broadcast service shall include the deferred_association_tags_descriptor in the first descriptor loop.

6.3.2 Deferred_association_tags_descriptor

The syntax and semantics of the deferred_association_tags_descriptor() in SBTVD compliant networks are described in Table 21.

Table 21 — Deferred_association_tags_descriptor

Syntax	Number of bits	Mnemonic
deferred_association_tags_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
association_tags_loop_length	8	uimsbf
for (i=0;i<N1;i++) {		
association_tag	16	uimsbf
}		
transport_stream_id	16	uimsbf
program_number	16	uimsbf
for (i=0;i<N2;i++){		
private_data_byte	8	uimsbf
}		
}		

Semantics of the deferred_association_tags_descriptor shall be the following:

— **descriptor_tag:** 8 bits field which shall have the value of 0x15;

— **descriptor_length:** 8 bits field which specifies the length of the descriptor in bytes;

— **association_tags_loop_length:** 8 bits field which defines the length in bytes of the loop of association tags that follows this field;

— **association_tag:** 16 bits field which contains the association_tag that is associated with either a stream that is not part of this data broadcast service or another SBTVD service;

— **transport_stream_id:** 16 bits field which indicates the transport stream in which the service resides that is associated with the enlisted association tags;

— **program_number:** 16 bits field which shall be set to the service_id of the service that is associated with enlisted association tags;

— **private_data_byte:** field which shall contain the deferred_service_location structure which is defined in Table 22.

Table 22 — Syntax for deferred_service_location structure

Syntax	Number of bits	Mnemonic
deferred_service_location() {		
org_network_id	16	uimsbf
for (i=0;i<N;i++) {		
private_data_byte	8	uimsbf
}		
}		

The semantics of the deferred_service_location structure shall be the following:

— **org_network_id**: 16 bits field that identifies the network_id of the delivery system from which the service originates;

— **private_data_byte:** 8 bits field which is not specified by the present document.

6.3.3 Stream type

The presence of an object carousel in a service shall be indicated in the program map table of that service by setting the stream type of the stream that contains the data carousel to the value of 0x0B (see ISO/IEC 13818-1) or an user defined value.

7 Multiprotocol encapsulation (MPE)

7.1 Data transport specification

Datagrams are encapsulated in datagram_sections which are compliant to the DSMCC_section format for private data (see ISO/IEC13818-6). The mapping of the section into MPEG-2 Transport Stream packets is defined in MPEG-2 Systems (ver ISO/IEC 13818-1).The syntax and semantics of the datagram_section are defined in Table 23.

Table 23 — Syntax of datagram_section

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

The semantics of the datagram_section shall be the following:

— **table_id:** 8 bits field which shall be set to 0x3E, DSM-CC sections with private data (see ISO/IEC13818-6:1998, Section 5);

— **section_syntax_indicator:** field which shall be set as defined by ISO/IEC13818-6:1998, Section 5;

— **private_indicator:** field which shall be set as defined by ISO/IEC13818-6:1998, Section 5;

— **reserved:** 2 bits which field that shall be set to "11";

— **section_length:** field which shall be set as defined by ISO/IEC13818-6:1998, Section 5;

— **MAC_address_[1..6]:** 48 bits field which contains the MAC address of the destination. The MAC address is fragmented in 6 fields of 8 bits, labelled MAC_address_1 to MAC_address_6. The MAC_address_1 field contains the most significant byte of the MAC address, while MAC_address_6 contains the least significant byte. Figure 3 illustrates the mapping of the MAC address bytes in the section fields.

NOTE The order of the bits in the bytes is not reversed and that the MSB (Most Significant Bit) of each byte is still transmitted first.

The MAC_address fields contain either a clear or a scrambled MAC address as indicated by the address_scrambling_control field;

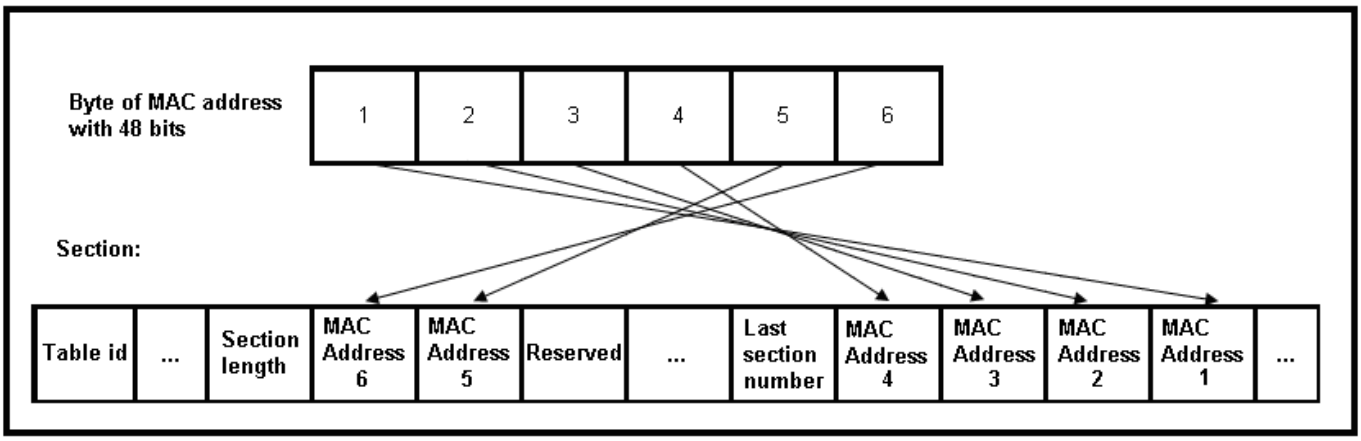

Figure 3 — Mapping of MAC address bytes to section fields

— **payload_scrambling_control:** 2 bits field which defines the scrambling mode of the payload of the section. This includes the payload starting after the MAC_address_1 but excludes the checksum or CRC32 field (see Table 24). The scrambling method applied is user private;

Table 24 — Coding of the payload_scrambling_control field

Value	payload_scrambling_control
0	Unscrambled
1	Defined by service
10	Defined by service
11	Defined by service

— **address_scrambling_control:** 2 bits field which defines the scrambling mode of MAC address in this subsection (see Table 25). This field enables a dynamic change of MAC addresses. The scrambling method applied is user private.

Table 25 — Coding of the address_scrambling_control field

Value	address_scrambling_control
0	unscrambled
1	defined by service
10	defined by service
11	defined by service

— **LLC_SNAP_flag:** 1 bit flag. If this flag is set to "1" the payload carries an LLC/SNAP encapsulated datagram following the MAC_address_1 field. The LLC/SNAP structure shall indicate the type of the datagram conveyed. If this flag is set to "0", the section shall contain an IP datagram without LLC/SNAP encapsulation;

— **current_next_indicator:** 1 bit field which shall be set to a value of "1";

— **section_number:** 8 bits field. If the datagram is carried in multiple sections, then this field indicates the position of the section within the fragmentation process. Otherwise it shall be set to zero;

— **last_section_number:** 8 bits field which shall indicate the number of the last section that is used to carry the datagram, or the number of the last section of the fragmentation process;

— **LLC_SNAP:** structure which shall contain the datagram according to ISO/IEC 8802-2 LLC (Logical Link Control) and ISO/IEC 8802-1 SNAP (SubNetwork Attachment Point). If the payload of the section is scrambled (see payload_scrambling_mode), these bytes shall be scrambled;

— **IP_datagram_data_byte:** bytes contain the data of the datagram. If the payload of the section is scrambled (see payload_scrambling_mode), these bytes are scrambled;

— **stuffing_byte:** optional 8 bits field whose value is not specified. If the payload of the section is scrambled (see payload_scrambling_mode), these bytes are scrambled. They are to assist with block encryption and data processing in wide bus environments. The number of stuffing_bytes used shall meet the data alignment requirements defined in the data_broadcast_descriptor;

— **checksum:** field which shall be set as defined by ISO/IEC13818-6:1998, Section 5. It is calculated over the entire datagram_section;

— **CRC_32:** field which shall be set as defined by ISO/IEC13818-6:1998, Section 5. It is calculated over the entire datagram_section.

7.2 PSI and SI specifications

The data broadcast service shall indicate the transmission of datagrams by including one or more data broadcast descriptors in SI (see ARIB STD-B23). Each descriptor shall be associated with a stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468:2005, Section 2) that may be present in the PSI (PMT) for the stream that is used to transmit the datagrams.

7.3 Transport protocol descriptor

The transport protocol descriptor is used in the following way:

— **protocol_id:** field which shall be set to 0x0002 to indicate the use of the multiprotocol encapsulation specification;

— **component_tag:** field which shall have the same value as a component_tag field of a stream_identifier_descriptor that may be present in the PSI program map section for the stream on which the data is broadcast;

— **selector_byte:** selector bytes which shall convey the multiprotocol_encapsulation_info structure which is defined in Table 26.

Table 26 — Syntax for multiprotocol_encapsulation_info structure

Syntax	Number of bits	Mnemonic
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
}		

The semantics of the multiprotocol_encapsulation_info structure shall be the following:

— **MAC_address_range:** 3 bits field which shall indicate the number of MAC address bytes that the service uses to differentiate the receivers according to Table 27;

Table 27 — Coding of the MAC_address_range field

MAC_address_range	Valid MAC_address bytes
0x00	Reserved
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	Reserved

— **MAC_IP_mapping_flag:** 1 bit flag. The service shall set this flag to "1" if it uses the IP to MAC mapping (see RFC 1112). If this flag is set to "0", the mapping of IP addresses to MAC addresses is done outside the scope of this Standard;

— **alignment_indicator:** 1 bit field that shall indicate the alignment that exists between the bytes of the datagram_section and the transport stream bytes according to Table 28;

Table 28 — Coding of the alignment_indicator field

Value	Bits alignment
0	8 (standard)
1	32

— **reserved:** 3 bits field that shall be set to "111";

— **max_sections_per_datagram:** 8 bits field that shall indicate the maximum number of sections that can be used to carry a single datagram unit.

7.4 Stream type

The presence of a multiprotocol data stream in a service shall be indicated in the program map section of that service by setting the stream type of that stream to the value of 0x0A (see ISO/IEC13818-6:1998, Section 5) or a user defined value.

8 Data piping transmission specification

8.1 Data transport specification

The data broadcast service shall insert the data to be broadcast directly in the payload of MPEG-2 TS packets.

The data broadcast service may use the payload_unit_start_indicator field and the transport_priority field of the MPEG-2 transport stream packets in a service private way. The use of the adaptation_field shall be MPEG-2 compliant.

The delivery of the bits in time through a data pipe is service private and is not specified in the present document.

8.2 PSI and SI specifications

The data broadcast service shall indicate the use of a data pipe by including one or more data broadcast descriptors in SI (see EN 300 468). Each descriptor shall be associated with a particular data pipe via a component_tag identifier.

In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468) that may be present in the PSI program map section for the stream that is used as a data pipe.

8.3 Transport protocol descriptor

The data broadcast descriptor shall be used in the following way:

— **protocol_id:** field which shall be set to 0x0003 to indicate a DVB data pipe. The other fields are present.

8.4 Stream type

The specification of the stream_type in the program map section shall be 0x7E (see ABNT NBR 15603-2:2007, Tabela J.1).

9 Independent PES transmission specification

9.1 Independent PES transmission

The independent PES transmission specification is a method used to implement streaming for data broadcasting services. The PES transmission specification defined in this chapter has the two types: synchronized type and asynchronous type.

The synchronized PES transmission system is used when it is necessary to synchronize data in a stream with other streams including video and audio. The asynchronous PES transmission specification is used when the synchronization is not necessary. As a major application example, it is expected that the synchronized type is used for transmitting captions and the asynchronous type is used for transmitting superimposed characters. For information related to the PES independent, see Annex A.

9.2 Synchronized PES

According to the synchronized PES transmission specification, data is transmitted using a PES packet specified in ISO/IEC 13818-1. Any mapping a PES packet to an MPEG-2 transport stream shall comply with ISO/IEC 13818-1.

According to the synchronized type transmission specification, a PES packet with the following restrictions is used in addition to the syntax and semantics specified in ISO/IEC 13818-1.

The PES packet header corresponding to the private_stream_1 shall be used the following:

— **stream_id:** in the case of a synchronized type stream, it shall be set to ‘0xBD‘ (private_stream_1);

— **PES_packet_length:** 16 bits field shall have a non-zero value.

The synchronized PES data structure shown in Table 29 shall be inserted into the PES_packet_data_bytes field.

Table 29 — Synchronized PES data structure

Syntax	Number of bits	Mnemonic
syncronized_PES_data() {		
data_identifier	8	uimsbf
private_stream_id	8	uimsbf
reserved_future_use	4	bslbf
PES_data_packet_header_length	4	uimsbf
for (i=0; i<N1; i++) {		
PES_data_private_data_byte	8	bslbf
}		
for(i=0;i<N2;i++){		
syncronized_PES_data_byte	8	bslbf
}		
}		

The semantics of fields in a Synchronized PES packet shall be the following:

— **data_identifier:** 8 bits field which shall be set to ’0x80’;

— **private_stream_id:** unused (0xFF);

— **PES_data_packet_header_length:** 4 bits field indicates the length in bytes of the PES_data_private_date_bytes;

— **PES_data_private_data_byte:** 8 bits field more detailed usage and this field depends on a service. A receiver unit may skip this field;

— **synchronized_PES_data_byte:** 8 bits field containing the transmitted data.

9.3 Asynchronous PES

According to the asynchronous PES specification, data is transmitted using a PES packet specified in ISO/IEC 13818-1. Any mapping a PES packet to an MPEG-2 transport stream shall comply with ISO/IEC 13818-1.

According to the asynchronous type transmission specification, a PES packet with the following restrictions is used in addition to the syntax and semantics specified in ISO/IEC 13818-1.

The PES packet header corresponding to private_stream_2 shall be used.

— **stream_id:** in case of an asynchronous type stream, it shall be set to ‘0xBF' (private_stream_2);

— **PES_packet_length:** 16 bit field which shall have a non-zero value.

The asynchronous PES data structure shown in Table 30 is inserted to the field of the PES_packet_data_bytes.

Table 30 — Asynchronous PES data structure

Syntax	Number of bits	Mnemonic
Asynchronous_PES_data() {		
data_identifier	8	uimsbf
private_stream_id	8	uimsbf
reserved_future_use	4	bslbf
PES_data_packet_header_length	4	uimsbf
for (i=0; i<N1; i++) {		
PES_data_private_data_byte	8	bslbf
}		
for(i=0;i<N2;i++){		
Asynchronous_PES_data_byte	8	bslbf
}		
}		

The semantics of fields in an Asynchronous PES packet shall be the following:

— **data_identifier:** 8 bit field which shall be set to '0x81';

— **private_stream_id:** unused (0xFF);

— **PES_data_packet_header_length:** 4 bits field which indicates the length in bytes of the PES_data_ private_date_bytes;

— **PES_data_private_data_byte:** 8 bits field and a more detailed usage of the area depends on a service. A receiver unit may skip this field;

— **asynchronous_PES_data_byte**: 8 bits field which contains the transmitted data.

10 Transmission protocol

10.1 Broadcast channel protocol

10.1.1 MPEG-2 transport stream

MPEG 2 transport stream shall be according to GEM 1.0:2005, Subsection 6.2.1

10.1.2 MPEG-2 section

MPEG-2 section shall be according to GEM 1.0:2005, Subsection 6.2.2

10.1.3 DSM-CC private data

DSM-CC private data shall be according to GEM 1.0:2005, Subsection 6.2.3

10.1.4 DSM-CC data carousel

DSM-CC data carousel shall be according to GEM 1.0:2005, Subsection 6.2.4

10.1.5 DSM-CC object carousel

For broadcast transport protocol that transmits the Ginga application contents, two systems are specified: the object carousel transmission system and data carousel transmission system.

Each system conforms to GEM 1.0:2005 Subsection 6.2.5 and is accessible via org.dvb.dsmcc package. (See table 31)

Table 31 — Equivalent functional

<table>
<tr><th>Nome</th><th>GEM</th><th>Implementação na ARIB STD-B23</th><th>Observações</th></tr>
<tr><td rowspan="3">Carousel</td><td rowspan="3">See GEM 1.0:2005, Subsection 6.2.5 and 11.7.2</td><td>See ARIB-STD-B23:2004, Annex B</td><td>In case of using a data carousel, see ARIB STD-B23:2004, Annex B and ARIB STD-B24</td></tr>
<tr><td colspan="2">*transport_stream_id, original_network_id*, *service_id* de *dvb_service_location()* na ARIB STD-B23:2004, table B.26: DVB endereço carrossel NSAP shall follow the semantics of ARIB-SI.
Any standard carousel is selectable.</td></tr>
<tr><td>ETSI TS 101 812:2003, Annex B, and ISO/IEC 13818-6 (DSM-CC carousel of objcts)</td><td>If using a object carousel, ETSI TS 101812 2003, Appendix B, applies</td></tr>
</table>

10.1.6 IP multicast transport protocol over broadcasting channel

IP multicast transport protocol over broadcasting channel shall be according to GEM 1.0:2005 Subsection 6.2.6.

10.1.7 IP protocol

IP protocol shall be according to GEM 1.0:2005, Subsection 6.2.7.

10.1.8 UDP protocol

UDP protocol shall be according to GEM 1.0:2005, Subsection 6.2.8.

10.1.9 Service information

Service information shall be according to ABNT NBR 15603-1, ABNT NBR 15603-2 and ABNT NBR 15603-3.

10.1.10 IP signaling

IP signalling shall be according to GEM 1.0:2005, Subsection 6.2.10.

10.2 Interaction channel protocols

10.2.1 Interaction channel protocol stack

Interation channel protocol stak conforms to GEM 1.0:2005, Subsection 6.3. This standard does not consider other protocols and the APIs that would provide access to them. Other private protocols and possibly APIs are not precluded and are outside of the scope of this specification.

Figure 4 illustrates the set of SBTVD defined interaction channel protocols that are accessible by MHP applications in some or all profiles (see ABNT NBR 15606-1). The full details of the API that provide access to these interaction protocols are in ETSI TS 101 812:2003, Section 11.

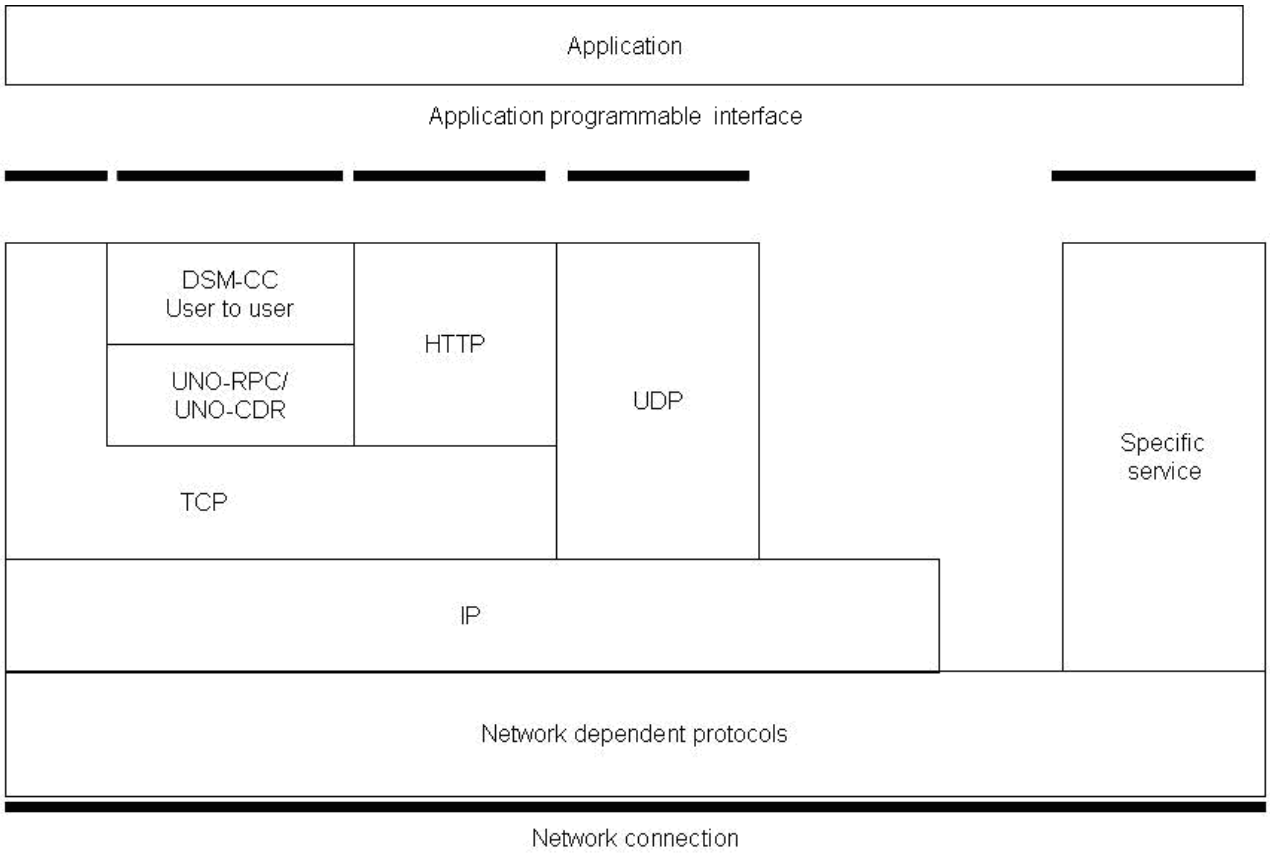

Figura 4 — Interaction channel protocol stack

10.2.2 Network dependent protocol

Network dependent protocol shall be according to GEM 1.0:2005 Subsection 6.3.1.

10.2.3 Internet protocol (IP)

Internet protocol is defined in RFC 791.

10.2.4 Transmission control protocol (TCP)

Transmission control protocol as defined in RFC 793

10.2.5 UNO-RPC

UNO-RPC shall be according to GEM 1.0:2005, Subsection 6.3.4.

10.2.6 UNO-CDR

UNO-CDR shall be according to GEM 1:2005, Subsection 6.3.5.

10.2.7 DSM-CC user to user

DSM-CC user to user shall be according to GEM 1.0:2005, Subsection 6.3.6.

10.2.8 HTTP protocol

HTTP protocol shall be according to GEM 1.0:2005, Subsection 6.3.7.

10.2.9 Specific service Protocol

Specific service Protocol shall be according to GEM 1.0:2005, Subsection 6.3.8.

10.2.10 User datagram protocol (UDP)

User datageam protocol (UDP) shall be according to GEM 1.0:2005, Subsection 6.3.9.

10.2.11 DNS

DNS shall be according to GEM 1.0:2005 Subsection 6.3.10.

10.3 Transport protocols for application loading over the interaction channel

Transport protocols for application loading over the interaction channel shall be according to GEM 1.0:2005, Subsection 6.4.

The file system implemented only by the interation channel shall be according to GEM 1.0:2005, Subsection 6.4.1.

The hybrid between broadcast stream and interaction channel shall be according to GEM 1.0:2005, Subsection 6.4.

11 Application model

11.1 Ginga application

The Ginga application shall be in accordance to GEM 1.0:2005 Section 9. In this standard the model é used as Ginga-J model, for the model NCL-Ginga the application shall be in accordance with the ABNT NBR 15606-2.

The Ginga application is defined as an extended application of the GEM application by adding the provisions of GEM 1.0:2005. The GEM application is a subset of the Ginga-J application. Therefore, the Ginga-J application prepared without the additional provisions is equivalent to the GEM application.

11.2 Ginga-J model

The Ginga-J model shall be in accordance to GEM 1.0:2005, Subsection 9.2.

11.3 How to handle NCL model

The NCL model shall be in accordance to ABNT NBR 15606-2.

11.4 Resource management between applications

Resource management between applications shall be in accordance to ABNT NBR 15606-2.

12 Transmission of application information

12.1 AIT descriptors and constant values

AIT descriptors and constant values shall be in accordance to ARIB STD-B23 and Table 32.

Table 32 — AIT Descriptors and constant values

Where used	Form	Value	Transmitted from	Scope	Descriptors
Data contents descriptor	Descriptor tag	0xC7	EIT	See 12.1 a 23.8, Annex B and C, ARIB STD-B-23	See ARIB STD-B23
Data coding descriptor		0xFD			
Carousel ID descriptor		0x13			
Association tag descriptor		0x14	PMT		
Extension tag descriptor		0x15			
Label descriptor	Descriptor tag	0x70	DII moduleinfo	See ARIB STD-B23 (object and data carousel)	See GEM - Middleware
Caching priority descriptor		0x71			
Content type descriptor		0x72	BIOP objectinfo		
Reserved for future descriptor OC by MHP		0x73-0x7F	OC		
Application information table (AIT)	Table ID no PID AIT	0x74		See ABNT NBR 15606-3, Section 12	
Application descriptor	Descriptor tag	0x00	AIT	See sectiom 12 and Annex C	
Application name descriptor		0x01			
Transport protocol descriptor		0x02			
Ginga-J application descriptor		0x03			
Ginga-J application location descriptor		0x04			
External application authorisation descriptor		0x05			
Ginga-NCL application descriptor		0x06			
Ginga-NCL application location descriptor		0x07			
NCL-HTML (reserved by MHP)		0x08 - 0x09			
NCL-HTML (reserved by MHP)		0x0A			
Application icons descriptor		0x0B			
Pre-fetch descriptor		0x0C			
DII location descriptor		0x0D			
Reserved for future uso by MHP		0x0E - 0x010			
IP signalling descriptor		0x11			
Reserved for future uso by MHP		0x12-0x5E			
Private data specifier descriptor		0x5F			
Reserved for future uso by MHP		0x60-07F			
User definition		0x80-0xFE			

Table 32 (continuation)

Where used	Form	Value	Transmitted from	Scope	Descriptors
Ginga coding system	Data coding scheme (data_component_id)	A0	PMT	See 12.2 a 12.18 and ARIB STD-B-23	See ARIB STD-B23
AIT transmission system		A3			
Data carousel transmission system	Transmission format (transmission_format)	1	Area of data component descriptor	See 12.2 a 12.18 and ARIB STD-B-23	
Object carousel transmission system		10			
Carrossel de objetos GINGA	Protocol ID (protocol_id)	0x0001	AIT	See 12.2 a 12.18 and ARIB STD-B-23	
Ginga dados carousel GINGA		0x0004			
Ginga-J application type	Application type (application_type)	0x0001	AIT	See 12.2 a 12.18 and ARIB STD-B-23	

12.2 Application execution Ginga

To realize the execution of the Ginga application, it is necessary to specify the application na transmit the additional aplilcation information to control the application.

This chapter describes the transmission protocolo f the application information to be used in this standard according to GEM 1.0:2005, Section 10.

Additional information according to the ARIB STD-B23 are the following:

— values of identification regarding Ginga and AIT to identify the storage of component data from additional_ ginga_j_info (), from the additional identifying information in the data component descriptor for the ES which transmits the application Ginga in PMT;

— storage of ginga_j_info () from the additional information within the data descriptor contents to be stored in the area of descriptors of an event program using applications Ginga in EIT;

— storage of ait_identifier_info () from the additional information within the data component descriptor for the ES which transmits the AIT in PMT;

— field additional_ ginga_j_info and ginga_j_info () , the carousel of objects shall be identified with the value'10'.

In addition to the information above, the Application Information Table (Table of Information Application) specified in 12.16.1, shall be transmitted in an ES that comprises the program in the form of private sections. Used to AIT, the information of application is stored (see 12.17.1) on the structure of the groups of descriptors stored in AIT. For additional information related to PMT and EIT tables , see Annex C.

12.3 Common application signal

Common application signal shall conform to the requirements specified in GEM1.0:2005, Subsection 10.1.1.

The information applications are transmitted in private sections from Table 46 (see 12.16.1), as an ES that comprises the program. Thus, the additional information required for each application is transmitted.

In addition, the values of identification of component data are determined so as to indicate the existence of transmission of AIT, as well as the application GINGA, and the structure of the selection of area the data component descriptor (see 12.7 for details).

The following descriptors shall be stored in the AIT as common information, without taking into account the format of the application:

— **transport protocol descriptor:** all applications shall be in the scope of at least one transport protocol descriptor. This descriptor can be stored both in a common information descriptor loop and an application information descriptor loop;

— **application descriptor:** a single application descriptor shall be stored in an application information descriptor loop for each application;

— **application name descriptor:** a single application shall be stored in an application information descriptor loop for each application.

12.4 Additional application signal necessary for Ginga

The requirements specified in the GEM 1.0:2005, Subsection 10.1.2, shall be observed. The required information, such as the application Ginga, should be transmitted via AIT.

NOTE For the purposes of this standard, Ginga-J and DVB-J are equivalent.

The descriptor enlace for the application of information AIT Ginga, the following descriptors shall be stored for at least one for each application.

— Ginga-J application descriptor;

— Ginga-J application location descriptor.

or

— Ginga-NCL application descriptor;

— Ginga-NCL application location descriptor.

12.5 Additional provision in PSI/SI

As for application information, application information Table 32 (AIT) is transmitted in private section mode as an ES that comprises the program. Additional provisions of application information transmission shall be the following:

— definition of the identification values corresponding to Ginga-J and AIT to identify the data component storage of additional_ginga_j_info() into additional identifying information in the data component descriptor for ES that transmits Ginga-J of PMT;

— storage of Ginga-J info() into additional information inside the data contents descriptor to be stored within the descriptor area of a program event that uses the Ginga-J application in EIT;

— storage ait_identifier_info() into additional information inside the data component descriptor for ES that transmits AIT of PMT;

— differentiation from other transmission systems by assigning 0x0004 as the protocol_id that corresponds to the data carousel transmission system or 0x0001 as protocol_id that corresponds to the object carousel. As for the details of the selector_byte, see 12.17.6;

— assignment of an object carousel transmission system to ‘10’ in the field of additional_ginga_j_info() and ginga_j_info() to identify the contents transmission system on the PAT/PMT level;

— In the case of transmission_format ='10 '(=object carousel transmission system), association_tag descriptor (value of tag: 0x14), the deffered_Association_tag descriptor (value of tag: 0x15), or Carousel_id descriptor (value: 0x13), specified in ISO / IEC 13818-6 shall be stored in PMT as needed.

12.6 Data component identification

A data_component_id is assigned to the Ginga application. Meanwhile, a value of the data component ID is assigned to AIT transmission and an elementary stream to be transmitted in private section mode is added to PMT.

12.7 Data component descriptor and data contents descriptor

12.7.1 Indirect reference

From the data carousel that transmits the Ginga application, indirect referencing with a data component descriptor relevant to the Ginga coding system and the data contents descriptor shall be made by the component tag (component_tag.)

12.7.2 Data component descriptor in Ginga application - Data coding system

When the data coding identification is made by the Ginga coding system, the additional_ginga_j_info() structure as shown in Table 33 is described within the area of additional identification information in the data component descriptor. Additional information that is not transmitted in AIT is stored here (see ABNT NBR 15603-2:2007, Subsection 8.3.20).

Table 33 – Additional_ginga_j_info()

Syntax	Number of bits	Mnemonic
additional_ginga_j_info() {		
transmission_format	2	bslbf
application_identifier_flag	1	bslbf
document_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
}		
}		

Description of additional_ginga_j_info( ) shall be following:

— **transmission_format** (transmission format): 2 bits area specifies the transmission system of the Ginga application (see Table 34);

Table 34 – Transmission format

Value	Description
00	Carrousel of data and message of events (except service only for data storage)
00	Carrousel of data (data service only for storage)
10	Carrousel of Objects
11	Reserved for the future

application_identifier_flag (application identifier flag): flag of 1 bit indicates whether the application identifier is included within the selector area (see Table 35);

Table 35 – Application identifier flag

default_version_flag	Descrição
0	Do not use the default value for the version number
1	Using default value for the version number

— **document_resolution** (document resolution): Ginga-J application resolution (corresponding to the resolution characteristics) and aspect ratio (corresponding to the display-aspect-ratio characteristics) are indicated. Choose one of the values at Table 36;

Tabela 36 – Document resolution

Valor	Descrição
0	Applications Ginga with multiple sizes and resolutions
1	1920 x 1080 (16:9)
10	1280 x 720 (16:9)
11	960 x 540 (16:9)
100	720 x 480 (16:9)
101	720 x 480 (4: 3)
110	160x120 (4:3)
111	160x90 (16:9)
1000	320x240 (4:3
1001	320x180 (16:9)
1010	352x288 (4:3)
1010 - 1111	Reserved for future

— **independent_flag** (independent listening and viewing audio and video availability flag): indicates whether the data-broadcasting program is assumed to be listened to and viewed independently; 0 = Impossible e 1 = Possible;

— **application_identifier()** (application identifier): a value to uniquely identify the application. See Table 37 for the details.

Table 37 — Coding of application identifier

Data structure	Bit rate	Bit string
application_identifier() {		
organization_id	32	bslbf
application_id	16	bslbf
}		

— **organization_id** (organization ID): 32 bits field which shows the system that created the application. The ID stores the internationally unique number that has been assigned;

— **application_id** (application ID): 16 bits field which stores the number that is uniquely assigned in the system to identify the application. If the application described by the descriptor is an additional service to a television program, it is used to specify the application that actually associates with this television program or radio;

— **download_id** (download ID): 32 bits field which serves as a label that uniquely identifies the carousel. This shows the carousel that shall be mounted by default;

— **ondemand_retrieval_flag** (flag of availability of reception of audio and video on demand): one-bit area which indicates, for the application reception transmitted by the said ES, whether the application acquisition from the carousel in each case of the audience operation is assumed. The receivability is regulated by the operation of each media entity, 0 = Not available in 1 = Available;

— **file_storable_flag** (file storable flag): indicates whether file storage of the corresponding data broadcasting program is possible. For example, file storage is difficult if the information is updated during the program. The storability is regulated by the operation of each media entity, 0 = File not for storage 1 = File for storage;

— **event_section_flag** (event section transmission section flag): 1 bit field which indicates whether the event message is distributed by this component; 0 = the event message is not distributed and 1 = the event message is distributed;

— **carousel_id** (carousel ID): 32 bits field is the identification value that uniquely specifies the object carousel. This identification value is specified by the carousel_id descriptor (carousel identifier descriptor) that is stored in PMT.

12.7.3 Data contents decriptor in Ginga application - Data content system

If the data coding identification is made by the Ginga coding system, the ginga_j_info() structure shown in Table 38 shall be described in the selector area of the data contents descriptor in EIT. This enables advanced notification of the Ginga application to be scheduled for use by the program event unit.

Information concerning Ginga application and control signals is stored in AIT. It is not assumed that the application is controlled by the program event unit. Accordingly, there is no mechnaism in AIT that comprehends the schedule by which the Ginga application will be used in advance for each program unit (see ABNT NBR 15603-2:2007, Subsection 8.3.28).

Table 38 — Ginga_j_info()

Data structure	Bit rate	Bit string
ginga_j_info(){		
transmission_format	2	bslbf
reserved_future_use	1	bslbf
document_resolution	4	bslbf
default_version_flag	1	bslbf
independent_flag	1	bslbf
application_identifier_flag	1	bslbf
content_id_flag	1	bslbf
associated_application_flag	1	bslbf
reserved_future_use	3	bslbf
update_flag	1	bslbf
ISO_639_language_code	24	bslbf
if (application_identifier_flag == 1) {		
application_identifier()	bslbf	
}		
if (content_id_flag==1) {		
content_id	32	uimsbf
content_version	16	uimsbf
}		
if (default_version_flag==0) {		
application_profiles_length	8	uimsbf
for (i=0; I<N; i++) {		
application_profile	16	uimsbf
profile_major_version	8	uimsbf
Profile_minor_version	8	uimsbf
profile _micro_version	8	uimsbf
}		
}		
if (transmission_format == ‘00’) {		
ginga_carousel_info()		
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
reserved_future_use	6	bslbf
} else if (transmission_format == ‘01’) {		
ginga_stored_carousel_info()		
} else if (transmission_format == ‘10’) {		
ginga_object_carousel_info()		
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
reserved_future_use	6	bslbf

Description of ginga_j_info( ) shall be following:

— **transmission_format** (transmission format): 2 bit area specifies Ginga application transmission system (see Table 39);

Table 39 – Format transmission

Value	Description
00	Data carousel and post events (except service only for data storage)
00	Data carousel (data service only for storage)
10	Object carousel
11	Reserved for the future

— **document**_resolution (document resolution): resolution of the Ginga-J application and display-aspect-ratio are indicated in Table 40;

Table 40 – Document resolution

Value	Description
0	Applications ginga with multiple sizes and resolutions
1	1920 x 1080 (16:9)
10	1280 x 720 (16:9)
11	960 x 540 (16:9)
100	720 x 480 (16:9)
101	720 x 480 (4: 3)
110	160x120 (4:3)
111	160x90 (16:9)
1000	320x240 (4:3)
1001	320x180 (16:9)
1010	352x288 (4:3)
1010 - 1111	Reserved for future

— **default_version_flag** (default version use flag): 1 bit flag which indicates that the default value specified by the operation is used as a profile for execution of the Ginga-J application that shall be transmitted by the corresponding ES. It shall be in accordance with Table 41

Table 41 – default_version_flag

Value	Description
0	Do not use the default value for the *version number*
1	Use the default value for the *version number*

— **independent_flag** (independent listening and viewing availability flag): indicates whether the data-broadcasting program is assumed to be listened to and viewed independently; 0 = Impossible and 1 = Possible;

— **application_identifier_flag** (application identifier flag): 1 bit flag which indicates whether the application identifier is included within the selector area; 0 = Not Included in 1 =Included;

— **content_id_flag** (contents ID flag): 1 bit flag which indicates whether the contents ID and the contents version are included in the descriptor; 0 = Not Included in 1 =Included;

— **associated_application_flag** (associated application flag): 1 bit flag which indicates that contents associate with the television program or radio program when the application described by this descriptor is an additional data service to the television program or radio program. For an application that is not an additional service, the value shall always be 0;

— **update_flag** (update flag): indicates whether there will be differential distribution to this application in the future; 0 = No differential distribution and 1 = There differential distribution;

— **ISO_639_language_code** (language code): language code used for Ginga application;

— **application_identifier( )** (application identifier): value to uniquely identify the application. See Table 42 for details;

Table 42 — Structure of application identifier

Data structure	Bit rate	Bit string
application_identifier() {		
organization_id	32	bslbf
application_id	16	bslbf
}		

— **organization_id** (organization ID): 32 bits field which indicates the system that prepared the application. This ID stores the internationally unique number that has been assigned;

— **application_id** (application ID): 16 bits field which stores the number that identifies the application. The number is uniquely assigned in the system. When the application described by this descriptor is an additional service to the television program or radio program, it is used to specify the application that actually associates with the television program or radio program;

— **content_id** (contents ID): 32 bits field which is a label to identify the data-broadcasting program and is assigned uniquely in the broadcasting company. In case of data storage, if the content_id has the same value as the one of the previous data-broadcasting program, the data can be overwritten;

— **content_version** (contents version): 16 bits field which indicates the version number among the data-broadcasting program that has an identical contents ID;

— **application_profiles_length** (application profile length specification): indicates the length of the field that specifies the receiver profile with which the application is executable;

— **profile_major_version** (profile major number): 8 bits field which indicates the major number among the version numbers of receiver profiles that shall correspond at least to the relevant Ginga application execution;

— **profile_minor_version** (profile minor number): 8 bits field which indicates the minor number among the version numbers of receiver profiles that shall correspond at least to the relevant Ginga application execution;

— **profile_micro_version** (profile micro number): 8 bits field which indicates the micro number among the version numbers of the receiver profiles that shall correspond at least to the relevant Ginga application execution;

— **ginga_carousel_info( ):** data structure specified in accordance to ARIB STD-B24:2007, Anexo C2.

— **ondemand_retrieval_flag:** (on demand reception for listening and viewing availability flag): 1 bit area which indicates, for the application reception transmitted by the said ES, whether the application acquisition from the carousel in each case of the audience operation is assumed. The receivability is regulated by the operation of each media entity; 0=Not available and 1 = Available

— **file_storable_flag** (file storable flag): indicates whether file storage of the corresponding data-broadcasting program is possible. For example, file storage is considered difficult if the information is updated during the program. The storability is regulated by the operation of each media entity;

— **ginga_stored_carousel_info():** data structure specified in ARIB STD-B24:2007, Annex C2;

— **ginga_object_carousel_info():** data structure specified in 12.7.4.2.

12.7.4 Data component descriptor for AIT transmission

12.7.4.1 Ait identifier info

When the data coding ID is AIT transmission, the ait_identifier_info() structure shown in the Table 43 shall be described within the selector area of the data component descriptor in PMT (see ABNT NBR 15603-2:2007, Subsection 7.2.3).

Table 43 — Ait_identifier_info()

Data structure	Bit rate	Bit string
ait_identifier_info(){		
for (i=0; i<N; i++) {		
application_type	16	uimsbf
reserved_future_use	3	bslbf
AIT_version_number	5	uimsbf
}		
}		

— **application_type** (application type): indicates the value of the application type to be transmitted in AIT. As for DVB, DVB-J is assigned 0x0001. In Ginga-J, the value shall be also 0x0001;

— **AIT_version_number** (AIT version number): current version_number is stored.

12.7.4.2 Selector area of data contents descriptor at object carousel transmission

When information for the object carousel reception control is inserted within the selector area of the data contents descriptor, the information shown in Table 44 shall be added at the position in the selector area specified for each data component.

Table 44 —Ginga_object_carousel_info() structure

Data structure	Bit rate	Bit string
ginga_object_carousel_info(){		
num_of_carousels	8	uimsbf
for(i=0; i< num_of_carousels; i++) {		
association_tag	16	uimsbf
event_section_flag	1	bslbf
reserved_future_use	3	bslbf
Component_size_flag	1	bslbf
default_transaction_id_flag	1	bslbf
default_timeout_DSI_flag	1	bslbf
default_leak_rate_flag	1	bslbf
if (component_size_flag == '1') {		
component_size	32	uimsbf
}		
if (default_transaction_id_flag == '1') {		
transaction_id	32	uimsbf
}		
if (default_timeout_DSI_flag == '1') {		
timeout_value_DSI	32	uimsbf
}		
if (default_leak_rate_flag == '1') {		
leak_rate	22	uimsbf
reserved	2	bslbf
}		
}		
}		

The ginga_object_carousel_info() structure shall be following:

— **num_of_carousels** (number of carousels): 8 bits field which indicates the number of object carousels included in a round of the loop;

— **association_tag** (association tag): 16 bits field which specifies the component stream, to which the DSI message with the ServiceGatewayInfo of the object carousel stored, by the association tag that is assigned by the association_tag descriptor of PMT;

— **event_section_flag:** with this component, the event message distribution is indicated;

— **component_size_flag** (component size flag): 1 bit field which indicates whether the component size is coded in the data structure. When the value of the component_size field is not defined yet, it is not coded (0: not coded; 1: coded);

— **default_transaction_id_flag:** 1 bit field which indicates whether the transaction ID is coded in the data structure. When the acquisition of DII for optional transaction ID is specified, the transaction ID is not coded (0: not coded; 1: coded);

— **default_timeoutDSI_flag:** 1 bit field which indicates whether the DSI timeout value is coded in the data structure. When the default value defined by the operation is used as the value of DSI timeout, it is not coded (0: not coded; 1: coded);

— **default_leak_rate_flag:** 1 bit field which indicates whether the leak rate is coded in the data structure. When the default value defined by the operation is used as the leak rate value, it is not coded (0: not coded; 1: coded);

— **component_size** (component size): 32 bits field which indicates the total data size (unit: byte) to be transmitted by the said object carousel;

— **transaction_id** (transaction ID): transaction ID value to be transmitted by the component. Non-coded transaction ID shows the necessity of DSI acquisition that has optional transaction ID;

— **time_out_value_DSI** (DSI timeout value): 32 bits field which indicates the recommended timeout value (unit: millisecond) for the whole DSI section reception of the relevant carousel. When the value is 0xFFFFFFFF, this means there is no recommendable timeout value;

— **leak_rate** (leak rate): 22 bits field which indicates the leak rate of the transport buffer of receiver. The unit is 50 byte/s.

12.8 Locator in application description

The locator on description of application shall be in accordance with the GEM 1.0:2005, Subsection 11.3. The locator descriptor shall be in accordance with the ARIB STD-B23: 2004, Section 14.

12.9 Application description

The application description shall be conform to GEM 1.0:2005, Subsection 10.4. The transmission system is described in Subsection 12.3. The section table structure of the AIT to be transmitted in the private section transmission mode shall be compliant with subseção 12.16. The string field in AIT can be encoded by an 8 unit character code as well as UTF-8. For operation, one of them shall be applied and mixing both coding shall be avoided. On the other hand, the strings acquired by the getName() method such as org.dvb.application API can be automatically converted to strings that are usable on Java.

Applications types shall be modified, including the application type 0x0008 as a GINGA reserved and 0x0009 as a Ginga-NCL application, as shown on Table 45.

Table 45 — Application type

Application type	Description
0x00	Reserved
0x0001	DVB-J or Ginga-J
0x0002	DVB-HTML
0x0006	ACAP-J
0x0007	ARIB - BML
0x0008	Reserved GINGA
0x0009	Ginga-NCL
0x000A…0x7FFF	Subject to registration with DVB

12.10 Transmission and monitoring of application description

The transmission and monitoring of application description shall be conform to GEM 1.0:2005, Subsection 10.4.1. As for the process of transmission, the section table that is compliant with the AIT structure is transmitted as ES that comprises the program via private section transmission mode.

For the operation, the value 0x0001, 0x0008 and 0x0009 are assigned as the application_type of Ginga-J, Ginga and Ginga-NCL, and 0x0001 is assigned as the protocol_id value to signify the object carousel transmission system.

The selector_byte in the transport protocol descriptor for the data carousel transmission system shall be the same syntax as the case of the “data carousel transport protocol” (protocol_id = 0x0004). For details of the structure, see Table 56.

12.11 Visibility of application description

The visibility of application description shall be conform to GEM 1.0:2005, Subsection 10.4.2.

12.12 Details of application description

Application descriptions are transmitted based on the transmission system described in 12.3. The application descriptors specified in 12.17 shall be used for storage of application descriptions.

12.13 Application handling from previously selected service

The application handling from previously selected service shall be conform to GEM 1.0:2005, Subsection 10.4.4.

12.14 Ginga-J specific application description

The Ginga-J specific application description shall bem conform to GEM 1.0:2005, Subsection 10.5.

12.15 Details of Ginga application description

Ginga application descriptions are transmitted based on the transmission system described in 12.3.

The AIT structure as well as the descriptor structure in AIT shall conform to 12.6. DVB-J in MHP shall be interpreted as Ginga-J in this Standard. The application descriptors specified in 12.18 shall be used for storage of application descriptions of the Ginga application.

12.16 Application information coding system

12.16.1 Application information

In AIT , all information relevant to the application and requirements for start-up status are stored. It is also possible to instruct the receiver to change the start-up status from the broadcasting station with the data in the AIT.

All AIT sections that have the same Applicaton_Type on the identical PID comprise a sub-table. The descriptor tag value in the AIT shall be unique in the AIT.

The data structure of the application information is shown in Table 46.

Table 46 — Application information table (AIT)

Data structure	Bit rate	Bit string
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
reserved_future_use	4	bslbf
application_descriptors_loop_length	12	uimsbf
for (j=0; j<M; ;j++) {		
descriptor ()		
}		
}		
CRC_32	32	rpchof
}		

The Application information section shall be following

— **table_id** (table ID): 8 bits field, 0x74 is stored to indicate that this is AIT table;

— **section_syntax_indicator** (section syntax indicator): the section syntax indication is always “1” in the 1 bit field;

— **section_length** (section length): 12 bits field. The first 2 bits shall always be “00.” This specifies the number of bits from the section length field to the last section including CRC32. The value shall be lower than 1021 (0x3FD in hexadecimal);

— **application_type** (application type): 16 bits field which indicates the value of application type that is being

transmitted by AIT. In DVB, 0x0001 is assigned to DVB-J application, the aplication types value is shown above in Table 45;

— **version_number** (version number): 5 bits field which is the version number of the sub-table. It is necessary to add one to the version number when a change is made in the information inside the sub-table. When the value reaches “31,” the next value will be back to “0.”;

— **current_next_indicator** (current next indication): this 1 bit indication is always “1”;

— **section_number** (section number): 8 bits field which shows the section number. The section number of the first section in the sub-table is 0x00. Each addition of a section that has the identical Table ID and application type adds “1” to the section number;

— **last_section_number** (last section number): 8 bits field which specifies the number of the last section in the sub-table to which the sections belong;

— **common_descriptors_length** (common descriptors loop length): 12 bits field which specifies the byte length of the subsequent common descriptors area. The descriptors inside descriptors area are applicable to all applications in the AIT sub-table;

— **application_control_code** (application control code): 8 bits field which specifies the control code that controls application status. This field is dependent on the value of application type. For detais see 12.16.5 ;

— **application_loop_length** (application information loop length): 12 bits field which specifies the whole loop byte length where subsequent application information is stored;

— **application_identifier ()** (application identifier): see Table 47 ;

— **application_descriptors_loop_length** (application information descriptors loop length): 12 bits field which specifies the byte length of the subsequent descriptors area. These descriptors in the descriptors area are applicable only to the designated application.

12.16.2 Application ID – Identificação de codificação da aplicação

An application is uniquely identified by the application identifier shown in Table 47.This identifier is comprised of a 6 bytes (48 bits) structure.

Table 47 — Application identifier

Data structure	Bit rate	Bit string
application_identifier () {		
organization_id	32	bslbf
application_id	16	bslbf
}		bslbf

The application identifier description shall be the following:

— **organization_id** (organization ID): 32 bits field which indicates the organization that created the application. This ID stores the number uniquely assigned in the world;

— **application_id** (application ID): 16 bits field stores which the number that identifies the application and is uniquely assigned in the organization ID. Application ID is divided into two ranges. One is the unsigned application range and the other is the signed application range. This division is made for security purposes. (see Annex B). The range of the value is respectively shown in Table 48.

Table 48 — Application ID value

Appication ID value	Application type
0x0000...0x3fff	Range for unsigned application
0x40000...0x7fff	Range for signed application
0x8000...0xfffd	Reserved by DVB
0xfffe	Wil card value (indicates all signed applications of the same organization ID)
0xffff	Wil card value (indicates all applications of the same organization ID)

In the application ID, the values 0xffff and 0xfffe are for wild card value. These are not used to specify the application, but, for example, are used as a descriptor for external application authorization. 0xffff corresponds to all applications that have the same organization.ID (ornanization_id). 0xfffe corresponds to all signed applications that have the same organization ID.

Sometimes the same application identifier is used between applications of different types, as in the case of the execution of the same function by different application types, for example.

NOTE Only one type appears in the collection of AIT sub-tables of the same application type in one service.

12.16.3 Effect on life cicle

The basic outline of the effect on life cycle is presented for the occasion when the service is changed over or the applications that have the same application identifier are started up, as follows:

— at service changeover, if service_bound_flag in the active application in the previous service is “0,” and the application identifier exists in the newly selected service’s AIT, then the application works continuously, unless there is any resource constraint;

— at service changeover, if service_bound_flag in the active application in the previous service is “0,” and if only the application is on the list of external application authorization descriptors, then the application works continuously, even if the application is not a part of the newly selected service, unless there is any resource constraint;

— as for an application that has one application identifier, only one instance is activated. Even if another application has the same application identifier, it cannot be activated if one application with the same application identifier has already existed;

— this affects the behavior of application start API or automatic start application. If service_bound_flag “1” is set for the application, the application will end (KILL) at each service selection.

12.16.4 Application ID authentication

The application ID authentication shall be conform to GEM 1.0:2005, Subsection 12.5. The application ID is authenticated with organization ID (organization_id) stored in the subject field of certificate.

12.16.5 Application life cycle control

For Application Life Cycle Control , the signaling mechanisms will be provided from the broadcasting station to control the life cycle for the standard type applications.

12.16.6 Entering and leaving the application domain application

Entering and leaving the application domain application is the domain defined as a collection of services that has applications listed in AIT. This means the applications are the ones listed in the application information loops of AIT or the ones listed in external application authorization descriptors. The services of which applications are not listed as in the way mentioned above are regarded as outside the application domain.

12.16.7 Ginga application dynamic control

The Ginga-J application synamic control should be in accordance to GEM 1.0:2005, Subsection 10.4.3 and extends ETSI TS 101 812:2003, Subsection 10.6.2.

GINGA defines the “application_control_code” value 0x07 as UNBOUND applications (see Table 49).

Table 49 — Ginga application control code values

Code	Identifier	Semantics
0x00		Reserved for future use
0x01	AUTOSTART	Applications with AUTOSTART control code are started automatically when the receiver changes to that service
0x02	PRESENT	Applications with PRESENT control code are not started automatically, but will be added to the receiver's list of available applications. The user can then choose to start this application by choosing it from that list
0x03	DESTROY	Applications with DESTROY will be automatically killed by the receiver
0x04	KILL	Applications with KILL will be automatically killed by the receiver. The difference from DESTROY control code is that an application with a KILL control code can be granted the option of keep running if the application chooses to
0x05		Reserved for future use
0x06	REMOTE	This identifies a remote application that is only launchable after service selection
0x07	UNBOUND	Applications with UNBOUND control code are similar to PRESENT applications. The difference from PRESENT is that the receiver will ask the user if the application shall be stored for postponed execution
0x08…0xFF		Reserved for future use

12.17 AIT Descriptors – Descriptor for tranmission of information of applications

12.17.1 Common descriptor

The descriptors to be commonly used in AIT regardless of the type of application are described from 12.17.2 to 12.17.11.

12.17.2 Application descriptor

One application descriptor is stored in the AIT application information descriptor loop for each application (see Table 50).

Table 50 — Application descriptor structure

Data structure	Bit rate	Bit string
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	
}		
}		

The *application_descriptor* structure shall be following:

— **descriptor_tag** (descriptor tag): 8 bits field, 0x00 is stored to indicate that this is the said descriptor;

— **application_profiles_length** (application profile information length): 8 bits field which indicates the whole byte length of application profile information that is included in the subsequent loop;

— **application_profile** (application profile): 16 bits field. The application profile that can execute the application is stored. If this profile is mounted in the receiver, this means the application is executable. The details of profile are defined for each application type;

— **version.major** (major version): 8 bits field which indicates the major version of the above mentioned profile;

— **version.minor** (minor version): 8 bits field which indicates the minor version of the above mentioned profile;

— **version.micro** (micro version): 8 bits field which indicates the micro version of the above mentioned profile.

The four field described above comprise the minimum profile for execution of this application. Ginga terminal starts this application if any of the following theoretical formulas is applicable and any profile that shows “true” exists in the application profile information:

(application profile $\in$ collection of profiles mounted in the terminal)

AND {(application major version < major version of the terminal for the profile)

OR [(application major version - major version of the terminal for the profile)

AND ({application minor version < minor version of the terminal for the profile }

OR {[application minor version = minor version of the terminal for the profile]

AND [application micro version ≤ micro version of the terminal for the profile]})] }

Detailed platform profile definitions in Ginga-J and Ginga NCL application shall be according to ABNT NBR 15606-1.

Profile coding in Ginga-J application shall be according GEM 1.0.

Profile coding in Ginga-NCL application shall be according to ABNT NBR 15606-2.

— **service_bound_flag** (service bound flag): 1 bit field which indicates if the application is effective only in the present service. If the field is “1,” the application is relevant only to the present service. When the service is changed to another service, end processing of the said application will be started;

— **visibility** (visibility): 2 bits field which indicates if the application is visible to end users when it is activated. Status definitions to the visibility value are shown in Table 51;

— **application_priority** (application priority): 8 bits field which indicates relative priority between the applications notified in the service;

— **transport_protocol_label** (transport protocol label): 8 bits field which stores the value for unique identification of the transport protocol that transports the application. The value corresponds to transport_protocol_label field value of transport protocol descriptor.

Table 51 — Visibility

Visible value	Description
0	This application is visible neither to other aplications via application enumeration API nor to users via navigator except for error information of log-out and the like
1	This application is not visible to users but visible from other applications via application enumeration API
10	Reserved for future
11	This application is visible to both users and other applications

12.17.3 Application name descriptor

One application name descriptor is stored for each application in the AIT application information descriptor loop. The application name is used for differentiation and gives information to users (see Table 52).

Table 52 — Structure of application name descriptor

Data structure	Bit rate	Bit string
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
}		
}		
}		

The application name descriptor structure shall be following:

— **descriptor_tag** (descriptor tag): 8 bits field, 0x01 is stored to indicate this is the said descriptor;

— **ISO_639_language_code** (language code): 24 bits field which identifies the language of the application name descriptor. The language code is indicated by the three-alphabetical -letter code as regulated in ISO 639-2. Each letter is coded in 8 bits according to ISO 8859-1 and inserted according to the order into the 24 bits field;

— **application_name_length** (application name description length): 8 bits field which indicates the byte length of the subsequent application name description;

— **application_name_char** (application name description): 8 bits field which specifies the letter description of the application name. This letter description is used as information for users.

12.17.4 Application icons information descriptor

The application icon information descriptor is stored by one at most for each application (see Table 53). The descriptor indicates the information about the icon relevant to the application. The contents format of the icon is coded by PNG and uses the system provided in ABNT NBR 15606-1.

Table 53 — Structure of application icon information descriptor

Data structure	Bit rate	Bit string
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
}		
}		

The application icon information descriptor structure shall be following:

— **descriptor_tag** (descriptor tag): 8 bits field which stores 0x0B that indicates the said descriptor;

— **icon_locator_length** (icon locator length): 8 bits field which indicates the byte length of subsequent icon locator;

— **icon_locator_byte** (icon locator): 8bits field which stores the locator to static image file as an icon. This locator shall be put before the icon file name and shall conform to GEM 1.0:2005, Subsection 10.4.3. As for Ginga-J application, the details shall conform to the provisions in the case where application_type is 0x0008 and a the relative path from the base directory defined by Ginga-J application is stored;

— **icon_flags** (icon flag): 16 bits field which stores the flag that indicates the size and the aspect ratio of the usable icon. The details conform to the provisions to GEM 1.0:2005, Subsection 10.4.3, and the coding for each case is as shown in Table 54. The value is stored after OR (logical ad) by unit.

Table 54 — Icon flag bits

Icon flag bits	Icon size and aspect relation
0000 0000 0000 0001	32 x 32 for square pixel display
0000 0000 0000 0010	32 x 32 for broadcast pixels on 4:3 display
0000 0000 0000 0100	24 x 32 for broadcast pixels on 16:9 display
0000 0000 0000 1000	64 x 64 for square pixel display
0000 0000 0001 0000	64 x 64 for broadcast pixels on 4:3 display
0000 0000 0010 0000	48 x 64 for broadcast pixels on 16:9 display
0000 0000 0100 0000	128 x 128 for square pixel display
0000 0000 1000 0000	128 x 128 for broadcast pixels on 4:3 display
0000 0001 0000 0000	96 x 128 for broadcast pixels on 16:9 display
xxxx xxx0 0000 0000	reserved_future_use

The file name of icon is coded according to the above described icon flag value. The file name coding method shall conform to GEM 1.0:2005, Subsection 10.4.3.

12.17.5 External application authorization descriptor

The external application authorization descriptor can be stored in one or more in the first common descriptor loop of AIT as needed. In each descriptor, information relevant to external applications are stored. An external application is the ones that can operate continuously with the applications listed in the AIT sub-table but cannot be activated from the said service.

This external authorization is applicable to the application that has application_identifier () in application_type specified by AIT that includes this descriptor (see Table 55).

Table 55 — Structure of external application authorization descriptor

Data structure	Bit rate	Bit string
external_application_authorisation_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0; i<N ; i++) {		
application_identifier ()		
application_priority	8	uimsbf
}		
}		

The external application authorization descriptor structure shall be the following:

— **descriptor_tag** (descriptor tag): 8 bits field where 0x05 is stored to indicate this is the said descriptor;

— **application_identifier()** (application identifier): 48 bits field which indicates the application with which external reference is available. For details of the field structure, see 12.16.1;

— **application_priority** (application priority): 8 bits field which indicates the priority of the present application on the assumption of the said service context. For details of the priority, see 12.17.2.

12.17.6 Transport protocol descriptor

The transport protocol descriptor indicates the transport protocol ID relevant to the service component and stores information about the protocol. This descriptor is stored in the first common descriptor loop or application information descriptor loop. When it is stored in the common descriptor loop, it is applicable to the whole sub-tables of AIT. The transport protocol descriptor in the application information descriptor loop describes the additional transport protocol to be used specifically in the application (seeTable 56).

The application described in this section shall be within the scope of at least one transport protocol descriptor.

Table 56 — Structure of transport protocol descriptor

Data structure	Bit rate	Bit string
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
}		
}		

The transport protocol descriptor structure shall be the following:

— **descriptor_tag** (descriptor tag): 8 bits field stores 0x02 that indicates this is the said present descriptor;

— **protocol_id** (protocol ID): 16 bits field which indicates the protocol that transports the application (see Table 57);

Table 57 — Protocol ID value

Value	Descriptor
0x0000	Resenved for future
0x0001	Object carousel Transport protocol
0x0002	IP via DVB Multiprotocol Ecapsutation defined at EN 301 192 and ETSI TR 101 202
0x0003	Reserved /Data Piping
0x0004	Data carousel Transport protocol
0x0005...FxFFFF	reserved_future_use

— **transport_protocol_label** (transport protocol label): 8 bits field which indicates the value that uniquely identify the transport protocol in the AIT section. For application descriptor, see the connection device (For example:elementary stream of carousel) that transports the application with these values;

— **selector_byte** (selector area): 8 bits field which stores additional information provided for each protocol ID. The data structure described in the area is specified separately for each protocol ID (see Table 58).

Table 58 — Selector_byte structure

Valor de Protocol_id	Selector byte structure
0x0000	Reserved for future
0x0001	See Table 54
0x0002	See 12.17.8
0x0003	Indefined / Data Piping
0x0004 (T.B.D.)	See Table 54
0x0005...FxFFFF	Reserved for future

12.17.7 Transport via OC (object carousel)

Data structures are shown in Table 59 for object carousel transport protocol (protocol_id=0x0001) and data carousel transport protocol (protocol_id=0x0004).

Table 59 — Structure of transport protocol descriptor selector area (in case of transport protocol of object carousel/data carousel transport protocol)

Data structure	Bit rate	Bit string
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

The description shall be the following:

— **remote_connection** (remote connection): if this 1 bit field is “1,” this shows that the actual service component is transmitted from another source than the one that transmits to AIT. A service like this is not performed automatically but is visible and possible to start via API In addition, REMOTE is stored to such application in application_control_code;

— **original_network_id** (original network ID): when remote_connection is “1,” the original network ID of the actual service transmission is stored;

— **transport_stream_id** (transport stream ID): when remote_connection is “1,” the transport stream ID of the actual service transmission is stored;

— **service_id** (service ID): when remote_connection is “1,” the service ID of the actual service transmission is stored;

— **component_tag** (component tag): indicates the main service component that transmits the application. In case of data carousel, the elementary stream that transmits the carousel automatically mounted at the application start is indicated. In case of object carousel, the elementary stream that transmits DSI is indicated.

12.17.8 Transport via IP

When the protocol ID is 0x0002 the selector bytes in the transport protocol descriptor shall be as shown in Table 60, To provide all the information necessary to obtain the Ginga applications and data components of the applications delivered by IP protocols.

Table 60 — Syntax of selector bytes for IP transport

Data structure	Bit rate	Bit string
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
}		

The description shall be the following:

— **remote_connection:** this and the associated 3 fields (original_network_id, transport_stream_id and service_id) have identical syntax and semantics to the fields with the same names under 12.17.8;

— **alignment_indicator:** 1 bit field which indicates the alignment that exists between the bytes of the datagram_section and the transport stream bytes;

— **URL_length:** 8 bits field which indicates the number of bytes in the URL;

— **URL_byte:** these bytes form a URL conforming to RFC 2396.

For URL using the "server" field including the host:port notation as defined in RFC 2396, only numeric IP addresses will be used for identifying IP transmissions carried in the broadcast channel as there is no Domain Name Service in the broadcast-only scenario to be used for resolving names. IP to MAC mapping will be done as described in RFC 1112.

NOTE This specification does not define or require any URL format to be supported in this descriptor. Hence it cannot be used in an inter-operable way.

12.17.9 IP signalling descriptor

The IP signalling descriptor is defined for use either in the "common" or in the "application" loop of the AIT. This descriptor indicates the identification of the organisation providing the IP multicast streams used by all applications (when present in the "common" loop) or by the particular signalled application (when present in the "application" loop). For the definition of the INT, see EN 301 192.

This descriptor and the INT with action_type 0x01 shall be used for applications relying on the presence of IP multicast streams on the broadcast link. The knowledge of the identification present in the descriptor enables to recover the appropriate IP notification table (INT) with action_type 0x01 that contains the correspondence between the multicast IP address and port and the stream localization (see Table 61).

Table 61 — Syntax of the IP signaling descriptor

Syntax	Bits	Mneumonics
ip_signalling_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
platform_id	24	uimsbf

The IP signaling descriptor description shall be the following:

— **descriptor_tag:** 8 bits field which with value 0x11 identifies this descriptor;

— **descriptor_length:** 8 bits field identifies the number of bytes following the length field;

— **platform_id:** 24 bits field containing a platform_id of the organisation providing IP/MAC streams on transport streams/services. Allocations of the value of platform_id are found in the ETSI TR 101 162.

12.17.10 Pre-fetch descriptor

Only one pre-fetch descriptor is stored in the application information descriptor loop in AIT as needed. This is defined to be used for object carousel (protocol_id=0x0001). Each descriptor is associated to one transport protocol descriptor via transport protocol label (see Table 62).

Ginga-J terminal may acquire modules denoted by the label in advance so as to speed up the starting time of the application. As for the labels, the descriptors of labels specified by the transmission of the carousel of objects are used in accordance with ISO / IEC 13818-6:1998, Annex B, and the ARIB STD-B23. The broadcast of this signaling is optional.

Table 62 — Structure of pre-fetch descriptor

Data structure	Bit rate	Bit string
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
}		
}		

The prefetch_descriptor structure shall be the following:

— **descriptor_tag**: 8 bits field where 0x0C is stored to show the said descriptor;

— **transport_protocol_label**: 8 bits field which stores the transport protocol label to specify the object carousel that transmits the modules referred in the above pre-fetch descriptor. For transport protocol label, see 12.17.6;

— **label_length :** This 8-bit field indicates the byte length of the label description to be included in the consequent loop;

— **label_char:** This is an 8-bit field. A module label is stored. This corresponds to the label description transmitted by the label descriptor that is stored in userInfo of moduleInfo of DII in the object carousel;

— **prefetch_priority:** This 8-bit field stores the values from 1 to 100. These values show the priority of pre-fetch. The higher value shows the higher priority.

12.17.11 DII location descriptor

For each application, only one DII location descriptor can be given as needed. This descriptor can be stored both in the common descriptor loop and application information descriptor loop. This is defined to be used for object carousel transmission system (protocol_id=0x0001). Each descriptor is associated with one transport protocol descriptor via transport protocol label.

The module group as a part of DSM-CC object carousel is notified by DownloadInfoIndication (DII). All DII messages that show the existing object carousel cannot be listed all in one at one location.

It is necessary to make all the relevant DII messages available in order to find the modules corresponding to labels for pre-fetch (see 12.17.10). DII location descriptor lists the locations of these DII.

In case the DII location descriptor is not included, only DII indicating modules that include ServiceGateway are found.

The loops of DII identification in the descriptor are located in the order of importance. Namely, DII that has labels with higher pre-fetch priority shall be located at the beginning of the loop. The receiver mounted with module-base pre-fetch mechanism checks DII in the order listed in DII location descriptor (see Table 63).

Table 63 — DII location descriptor structure

Data structure	Bit rate	Bit string
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
}		
}		

The DII location descriptor structure shall be the following:

— **descriptor_tag**: 8 bits field which stores 0x0D that indicates this is the said descriptor;

— **transport_protocol_label**: 8 bits field which stores the transport protocol label that specifies the object carousel that is transmitting the modulesreferredin pre-fetch descriptor. For details of transport protocol label, see 12.17.6;

— **DII_identification:** 15 bits field stores the value that specifies the DIImessage. This value corresponds to the bit 1-15 of the transaction ID in dsmMessageHeader () of DII message;

— **association_tag:** 16 bits field indicates the relationship with DII message that is broadcasted (for example: elementary stream).

12.18 Ginga application descriptors

12.18.1 Structure of Ginga application descriptor

One descriptor is stored in the AIT application information descriptor loop for each Ginga-J application. This indicates the parameter information for application starting (see Table 64).

Table 64 — Structure of Ginga application descriptor

Data structure	Bit rate	Bit string
ginga_j_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
}		
}		
}		

The Ginga-J application descriptor structure shall be the following:

— **descriptor_tag:** 8 bits field where 0x03 is stored to indicate the Ginga-J application descriptor; and 0x06 is stored to indicate the Ginga-NCL application descriptor;

— **parameter_length:** 8 bits field shows the byte length of the subsequent parameter description;

— **parameter_byte:** 8 bits field. The string to be given to the application as parameters is stored.

12.18.2 Ginga application location descriptor

The descriptor is stored in AIT application information descriptor loop by one for each Ginga-J application. This stores necessary path information in the application (see Table 65).

Table 65 — Structure of Ginga application location descriptor

Data structure	Bit rate	Bit string
ginga_j_application_location_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
base_directory_length	8	uimsbf
for (i=0; i<N ; i++) {		
base_directory_byte	8	uimsbf
}		
classpath_extension_length	8	uimsbf
for (i=0; i<N ; i++) {		
classpath_extension_byte	8	uimsbf
}		
for (i=0; i<N ; i++) {		
initial_class_byte	8	uimsbf
}		
}		

The Ginga-J application location descriptor structure shall be the following:

— **descriptor_tag:** 8 bits field where 0x04 is stored to indicate the Ginga-J application location descriptor; and 0x07 is stored to indicate the Ginga-NCL application location descriptor;

— **base_directory_length:** 8 bits field which indicates the byte length to be included in the subsequent loop. The stored value shall be 1 or higher;

— **base_directory_byte:** 8 bits field. The directory name from the file system root shall be stored by delimiter letter using “/” (0x2F). This directory name is used for the base directory in the relative path. If the root directory of file system is designated as a base directory, “/” shall be stored;

— **classpath_extension_length:** 8 bits field indicates the byte length of subsequent additional class path;

— **classpath_extension_byte:** 8 bits field. When a class path is designated by the directory other than base directory, the class path name is stored. The directory name from the file system root is stored by the delimiter letter using “/” (0x2F). If there are more than one paths, they shall be stored by enumerating with “;” (0x3B);

— **initial_class_byte:** 8 bit field. The object name on the file system of Xlet interface mounted class is stored.

13 Event message transmission specification

13.1 Event message

The event message transmission specification provides a means to send message information immediately or at a specified time for the application operated on a receiver unit from a broadcast station.

The event message transmission specification defined in this chapter is extended to deal various times appointing methods requested from the application based on the specification of stream descriptor and its DSM-CC section transmission specification specified in ISO/IEC13818-6.

13.2 Stream descriptors

13.2.1 DSM-CC stream descriptor

This section conforms to the requirements of ISO-13818-6.

Stream descriptors may be used to provide DSM-CC information that is correlated with a MPEG-2 transport stream or program stream. These descriptors are in the format of program and program element Descriptors as defined in ISO/IEC 13818-1. DSM-CC stream descriptors shall only be carried in a DSMCC_section (see ISO/IEC13818-6:1998, seção 9). This creates a unique identifier space (from that defined by ISO/IEC 13818-1) for descriptor tag values. The general format for all stream descriptors defined in this specification is shown in Table 66.

Table 66 — DSM-CC stream descriptor

Syntax	Number of bits	Mnemonic
dsmccStreamDescriptor () {		
descriptorTag	8	uimsbf
descriptorLength	8	uimsbf
descriptor()		
}		

The DSM-CC stream descriptor structure shall be the following:

— **descriptorTag:** 8 bits field which identifies each descriptor. The range of possible value descriptorTag are shown in Table 67;

— **descriptorLength:** 8 bit field which specifying the number of bytes of the descriptor immediately following descriptorLength field.

Table 67 — DescriptorTag field value

descriptorTag	Description
0x00	ISO/IEC13818-6 reserved
0x01	NPT reference descriptor
0x02	NPT endpoint descriptor
0x03	Stream mode sescriptor
0x04	Stream Event descriptor
0x05 - 0xFF	ISO/IEC13818-6 reserved

13.2.2 NPT reference descriptor

To enable the determination of the NPT time of an event, the NPT reference descriptor is defined. The format of the NPT reference descriptor is shown in Table 68.

Table 68 — NPT reference descriptor

Syntax	Number of bits	Mnemonic
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
}		

— **descriptorTag:** 8 bit field that identifies the type of stream descriptor. The value of the descriptorTag field for the NPT reference descriptor is shown in Table 62;

— **descriptorLength:** 8 bit field specifying the number of bytes of the descriptor immediately following descriptorLength field;

— **postDiscontinuityIndicator:** 1 bit field. A value of 0 indicates that the NPT reference descriptor is valid upon reception. A value of 1 indicates that the NPT reference descriptor will become valid at the next system time base discontinuity as defined in ISO/IEC 13818-1;

— **contented:** 7 bits field that identifies which of a nested set of content is being presented. The contentId field may be used to indicate a transition to a different NPT time base within an existing NPT time base. For example, this field may be changed when a commercial is presented within a television show, and then changed back when the television show is resumed;

— **STC_Reference:** 33 bits unsigned integer that indicates the STC value for which the NPT equals the value given in the NPT_Reference field. The value of the STC_Reference field is specified in units of the period of the system clock frequency, as defined in ISO/IEC 13818-1, divided by 300, yielding units of 90 kHz. The STC_Reference is derived from the system clock frequency as shown in equation below:

$$STC\_Refere\ ncek = (STCNPT(k)\ /\ 300)\ \%\ 233$$

where

STCNPT(k) is the value of the System Time Clock when the NPT equals the value of the NPT_Reference;

— **NPT_Reference:** signed 33 bit integer indicating the NPT value at the STC value given in the STC_Reference field;

— **scaleNumerator:** signed 16 bits integer used with the scaleDenominator, a 16 bit unsigned integer, to define the rate of change of the NPT in relation to the STC. A value of 1 for scaleNumerator with a value of 1 for scaleDenominator indicates that the NPT is changing at a rate equivalent to the STC, yielding the standard presentation rate. A value of 0 for scaleNumerator with a non-zero value for scaleDenominator indicates that the NPT is not changing in relation to the STC, yielding a constant value of the NPT. A value of 0 for scaleNumerator with a value of 0 for scaleDenominator indicates that the scaleNumerator and scaleDenominator fields are not provided in the NPT Reference Descriptor. A non-zero value for scaleNumerator with a value of 0 for scaleDenominator shall not be used (see Table 69).

Table 69 — ScaleNumerator

scaleNumerator	*scaledenominator*	Semantic
0	0	Indicates that scaleNumerator ans scaleDenominator are not used
0	Other than 0	NPT continues constant value irrelevant of STC
1	1	NPT and STC advance at the same rate
Other than 0	0	Such combination shall not be used

13.3 Stream mode descriptor

The stream mode descriptor contains information about the mode of the stream state machine, allowing clients to better synchronize their actions with stream state changes. The format of the stream mode descriptor is shown in Table 70.

Table 70 — Stream mode descriptor

Syntax	Number of bits	Mnemonic
StreamModeDescriptor () {		
descriptorTag	8	uimsbf
descriptorLength	8	uimsbf
streamMode	8	uimsbf
reserved	8	bslbf
}		

The StreamModeDescriptor () structure shall be the following:

— **descriptorTag:** 8 bits field that identifies the type of stream descriptor. The value of the descriptorTag field for the stream mode descriptor is shown in Table 65;

— **descriptorLength:** 8 bits field specifying the number of bytes of the descriptor immediately following descriptorLength field;

— **streamMode:** 8 bits field whose value indicates the current state of the stream state machine. The values for streamMode are shown in Table 71. The stream state machine states are defined ISO/IEC 13818-6:1998, Section 5.

Table 71 — StreamMode field values

streamMode	Descrição
0	Open
1	Pause
2	Transport
3	Transport Pause
4	Search Transport
5	Search Transport Pause
6	Pause Search Transport
7	End of Stream
8	Pre Search Transport
9	Pre Search Transport Pause
10 - 255	Reserved ISO/IEC13818-6

13.4 Stream event descriptor

The stream event descriptor contains information allowing the transmission of application-specific events as defined in ISO/IEC 13818-6:1998, Section 5, so that they may be synchronous with the stream. Note that the definition of event in this context is not the same as event as it relates to NPT. The format of the stream event descriptor is shown in Table 72.

Table 72 — Stream event descriptor

Syntax	Number of bits	Mnemonic
StreamEventDescriptor () {		
descriptorTag	8	uimsbf
descriptorLength	8	uimsbf
eventId	16	uimsbf
reserved	31	bslbf
eventNPT	33	uimsbf
for(i=0;i<N;i++){		
privateDataByte	8	uimsbf
}		
}		

The StreamEventDescriptor () structure shall be the following:

— **descriptorTag:** 8 bits field that identifies the type of stream descriptor. The value of the descriptorTag field for the stream event descriptor is shown in Table 72;

— **descriptorLength:** 8 bits field that specifies the number of bytes of the descriptor immediately following descriptorLength field;

— **eventide:** 8 bits field whose value is the type of the application specific event;

— **eventNPT:** All unsigned integer whose value is the value of the NPT when the event occurred, or the value of the NPT when the event will occur;

— **privateDataByte:** fields that allows inclusion of application specific data in the stream event descriptor.

13.5 General event descriptor

The general event descriptor (general_event_descriptor) is a descriptor to communicate information applied generally to event messages.

The data structure of general event descriptor is shown in Table 73.

Table 73 — General event descriptor

Syntax	Number of bits	Mnemonic
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
}		
}		

The General_event_descriptor () structure shall be the following:

— **event_msg_group_id:** 12 bits field that indicates the identifier to identify the message group to be received by the applicator. Details of operations are specified for each data coding identifier. When operating an event message having more than one message group identification at the same time, only general event descriptors having the same message group identifier shall be included in one DSM-CC section;

— **time_mode:** 8 bit field that indicates the method to designate the time when the event message is generated (see Table 74);

Table 74 — Time Mode

time_mode	Method for time designation	Semantic
0x00	None	Event message is generated immendiately after reception
0x01	*MJD_UTC_time*	Event Message is generated at the absolute time indicates by MJD and UTC time. Event message is also generates when playing back stream recorded contents referring playing back time
0x02	NPT	Event message is generated at the time specified with the NPT time data
0x03	*eventRelativeTime*	Event message is generated when the period specified in this field(in milliseconds) after the program start time
0x04	-	Reserved for future
0x05	*MJD_UTC_time*	Event Message is generated at the absolute time indicates by MJD and UTC time. When playing back stream recorded contents, event message is generated by referring to the on air time
0x06-0xFF	-	Reserved for future

— **event_msg_MJD_UTC_time:** 40 bits field that is coded in the case of time_mode = 0 x 01 or 0x05 and indicates the time when the event message is generation in JUTC and MJD (refer to ARIB STD-B10, Part 2, Appendix C). This field contains a copy of the lowest 16 bits of MJD and t followed by six 4 bits binary coded decimal (BCD) representations;

— **event_msg_NPT:** 33 bits field that is coded in the case of time_mode = 0 x 02 and indicates the time using Normal Play Time of DSM-CC (see 7.1), when the event message is genered;

— **event_msg_relativeTime:** 36 bits field that is coded in the case of time_mode = 0 x 03 and indicates that the event message is generated when the period specified in this field passes after the program start time. The value of this field is described in the order of hour (2 digits), minute (2 digits), second (2 digits), millisecond (3 digits) to form nine 4 bits binary coded decimal (BCD) representations;

— **event_msg_type:** an identifier indicating the event message type. Usage and semantics is specified in each data coding specification;

— **event_msg_id**: 16 bit field that contains the identifier to identify each event message. Usage and semantics is specified in each data coding specification;

— **private_data_byte:** 8 bit field that stores the information related to the event message required by the data coding specification specified in event_msg_type.

13.6 Syntax of DSM-CC section transmitting stream descriptor

The stream descriptor is transmitted in the DSM-CC section shown in Table 75.

Table 75 — DSM-CC section (transmission of stream descriptor)

Syntax	Number of bits	Mnemonic
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
}		
}		

The *DSM-CC_section* () structure shall be the following:

— **table_id :** 8 bits field that is set to 0x3D to indicate that the stream descriptor is stored in the payload of the DSM-CC section;

— **section_syntax_indicator:** 1 bit field that indicates that CRC32 exists at the end of the section when it is 1. When it is 0, it indicates that check sum exists. To transmit an event message, this field shall be set to 1;

— **private_indicator:** 1 bit field that stores the complement value of the section syntax indicator value;

— **dsmcc_section_length:** 12 bits field that indicates the byte length of the area from the position immediately following this field to the end of the section. This field value shall not exceed 4093;

— **data_event_id:** 4 bits field that is the identifier to identify the data event among the preceding and following data events that uses the event message to allow for the intended local content, despite of being preceded and/or followed by other local contents, to receive the event message. Adjacent local contents when transmitting are allocated with different identifiers;

— **event_msg_group_id:** 12 bits field that contains the identifier to identify the event messages to be received by the application. Detailed semantics is specified in each data coding specification;

— **version_number:** 5 bits field that is the version number of the sub-table. The version number is incremented by 1 when any piece of information in the sub-table has been changed. The available values are from 0 to 31.The value 0 is used to update the value 31;

— **current_next_indicator:** 1 bit indicator that indicates that the sub-table is the current sub-table when it is '1'. When it is '0', the sent sub-table is not yet applied and used as the next sub-table;

— **section_number:** 8 bits field indicates the section number;

— **last_section_number:** 8 bits field indicates the number of the last section (that is the section having the maximum section number) of the sub-table to which the concerned section belongs.

14 Broadcast filesystem and trigger transport

The broadcast filesystem and trigger transport shall be conform to GEM 1.0:2005, Annex B.

Annex A
(normative)

Video and audio PES

A.1 Data transmission format of MPEG-2 video PES coded

When using video PES of MPEG-2 video (see ISO/IEC 13818-2) to transmit data, the user data area (user data field) following the picture header of a video stream shall be used. The syntax of the User Data field is shown in Table A.1. A more detailed usage of this area depends on broadcasters.

Table A.1 — Syntax of user data field of video stream

Syntax	Number of bits	Mnemonic
User Data () {		
user_data_start_code	32	bslbf
while (nextbits() != 0x000001){		
user_data	8	uimsbf
}		
next_start_code()		
}		
NOTE Código de início de dados de usuários: 0x000001b2.		

A.2 Data transmission format of audio PES coded with MPEG-2 BC audio

When using audio PES of MPEG-2 BC Audio (see ISO/IEC 13818-3) to transmit data, the ancillary data area, which may contain data other than MPEG audio for each audio frame, shall be used. The syntax of the ancillary data area is shown in Table A.2. A more detailed usage of this area depends on broadcasters.

Table A.2 — Ancillary data area of audio stream

Syntax	Number of bits	Mnemonic
MPEG1_ancillary_data() {		
if(ext_bit_stream_present == 1){		
for(b=0; b<8*n_ad_bytes;b++)		
ancillary_bit	1	bslbf
}		
}		

A.3 Data transmission format of audio PES coded with MPEG-2 AAC audio

When using audio PES of MPEG-2 AAC Audio (see ISO/IEC 13818-3) to transmit data, the data_stream_element area, which may contain data other than audio data for each raw_data_block shall be used. The syntax of this area is shown in Table A.3. A more detailed usage of this area depends on broadcasters.

Table A.3 — Data stream element area of áudio stream (MPEG-2 AAC audio)

Syntax	Number of bits	Mnemonic
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
}		

Annex B
(normative)

PSI/SI information for data carousel trasmission and event message transmission

B.1 Data coding specification based on data carousel and event message scheme

In addition to the data coding specification applied to data transmission based on the data carousel and event message scheme, an additional syntax which depends on data transmission format to be inserted in data_component_descriptor in PMT and data_content_descriptor in EIT specified in ARIB SDT-B23 shall be defined.

This Standard is based on the following assumptions about transmission operation for data broadcasting services, as following:

— the DII and DDB belonging to one carousel are transmitted in one ES;

— one data broadcasting service may consist of two or more carousels. Event messages may be transmitted.

B.2 Content of *additional_data_component_info* loop of *data_component_descriptor*

To insert the information for reception control of the data carousel and the possible event messages in the loop which contains additional_data_component_info at the end of data_component_descriptor, the following data structure shall be placed in the loop, as specified by the data coding specification.

Table B.1 – *Additional_ginga_carousel_info*

Syntax	Number of bits	Mnemonic
additional_ginga_carousel_info() {		
data_event_id	4	uimsbf
event_section_flag	1	bslbf
reserved	3	bslbf
}		

The additional_ginga_carousel_info () service shall be the following:

— **data_event_id:** 4 bits identifier that recognizes the preceding and following data events using the data carousel and the possible event messages to avoid failing to receive the appropriate local content transmitted in the data carousel and the possible event messages. In the case of all bits of this field set to 1, it means that the DIIs having a data_event_id identifier delivered in this service and the event messages is valid;

— **event_section_flag:** 1 bit field indicates whether or not event messages are sent with this component, as following:

0: Event messages are not transmitted;

1: Event messages are transmitted.

— **reserved:** reserved.

B.3 Selector byte of data_contents_descriptor

B.3.1 Data structure

To insert the information for data carousel reception control in the selector byte of the data contents descriptor such as EIT, a data structure shall be placed in the selector_byte field, as specified by the concerned data coding specification.

B.3.2 Data structure for data carousel reception control for non-stored data services

For non-stored data services, insert the information in Table B.2.

Table B.2 – Non-stored data services

Syntax	Number of bits	Mnemonic
ginga_carousel_info () {		
num_of_carousels	8	uimsbf
for(i=0;i<num_of_carrousels;i++){		
component_tag	8	uimsbf
event_section_flag	1	bslbf
reserved_future_use	3	bslbf
component_size_flag	1	bslbf
default_transaction_id_flag	1	bslbf
default_timeout_DII_flag	1	bslbf
default_leak_rate_flag	1	bslbf
if(component_size_flag == ‘1’){		
component_size	32	uimsbf
}		
if(default_transaction_id_flag == ‘1’){		
timeout_value_DII	32	uimsbf
}		
if(default_leak_rate_flag == ‘1’){		
leak_rate	22	uimsbf
reserved	2	bslbf
}		
}		
}		

The ginga_carousel_info () service shall be the following:

— **num_of carousels:** 8 bits field that indicates the number of carousels included in the following loop;

— **component_tag:** 8 bits field that designates the component stream transmitting the carousels with the component tag given by the stream identifier descriptor in PMT;

— **event_section_flag:** field that indicates whether or not event messages are sent using this component;

— **component_size_flag:** 1 bit field that indicates whether or not the data structure contains the component size. When the component_size field value is not available, it shall be set to “0”.(0: not coded; 1: coded);

— **default_transaction_id_flag**: 1 bit field indicates whether or not the transaction identifier is coded in this syntax. To designate to gain DII of optional transaction identification, transaction identification shall not be coded (0: not coded; 1: coded);

— **default_timeout DII_flag:** 1 bit field indicates whether or not the DII timeout value is coded in this syntax. When default value specified in the operation as DII timeout value is used, it is not coded (0: not coded; 1: coded);

— **default_leak_rate_flag:** 1 bit field that indicates whether or not leak rate is coded in this syntax. When default value specified in the operation as leak rate value is used, it is not coded (0: not coded; 1: coded);

— **component_size:** 32 bits field that indicates the total size (in bytes) of the data transmitted in the carousels in this component;

— **transaction_id:** DII transaction identification value transmitted in this component. In the case of the transaction identification is not present, a DII with a transaction identification shall be gained;

— **time_out_value_DII:** 32 bit field that indicates the recommended timeout value (in milliseconds) to receive the whole section of the DII of this carousel. When the value is 0 x FFFFFFFF, it means that t no recommended timeout value exists;

— **leak_rate:** 22 bits field that indicates the leak rate of the transport buffer of the receiver unit in a unit of 50 byte/s.

B.3.3 Data structure for data carousel reception control for stored data services

For stored data services, insert the information in Table B.3.

Table B.3 — Stored data services

Syntax	Number of bits	Mnemonic
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
}		
}		

The ginga_stored_carousel_info () service shall be the following:

— **num_of carousels:** 8 bits field that indicates the number of carousels included in the following loop;

— **component_tag:** 8 bits field that designates the component stream transmitting the carousels with the component tag given by the stream identifier descriptor in PMT;

— **um_dataEvent_flag:** 1 bit field that indicates whether or not the data structure contains the number of data events. When the num_dataEvent field value is not available, it shall be set to 0 (0: not coded; 1: coded);

— **num_modules_flag:** 1 bit field that indicates whether or not the data structure contains the total number of the modules. When the num_modules field value is not available, it shall be set to 0 (0: not coded; 1: coded);

— **num_resources_flag:** 1 bit field that indicates whether or not the data structure contains the total number of the resources. When the num_resources field value is not available, it shall be set to 0 (0: not coded; 1: coded);

— **compressed_component_size_flag:** 1 bit field that indicates whether or not the data structure contains the compressed component size. When the compressed_component_size field value is not available, it shall be set to “0” (0: not coded; 1: coded);

— **component_size_flag:** 1 bit field that indicates whether or not the data structure contains the component size. When the component_size field value is not available, it shall be set to “0” (0: not coded; 1: coded);

— **default_transaction_id_flag**: 1 bit field that indicates that the transaction identifier is coded or not in this syntax. To designate to gain DII of optional transaction identification, transaction identification shall not be coded (0: not coded; 1: coded);

— **default_timeout DII_flag:** 1 bit field that indicates whether or not the DII timeout value is coded in this syntax. When the default value specified in the operation as DII timeout value is used, it is not coded (0: not coded; 1: coded);

— **default_leak_rate_flag:** 1 bit field that indicates whether or not the leak rate is coded in this syntax. When the default value specified in the operation as leak rate value is used, it is not coded (0: not coded; 1: coded);

— **num_dataEvent:** 32 bits field that indicates the total number of the data events in the concerned component;

— **num_modules:** 32 bits field that indicates the total number of the modules in the data events in the concerned component;

— **num_resources:** 32 bits field indicates the total number of the resources in the concerned component;

— **compressed_component_size:** 32 bits field that indicates the total size (in bytes) of the data in the data events in the data carousels in this component. The size of a compressed module is calculated based on the compressed state, not the extracted state;

— **component_size:** 32 bits field that indicates the total size (in bytes) of the data in the data events in the data carousels in this component. The size of a compressed module is calculated based on the extracted state, not the compressed state;

— **transaction_id:** DII transaction identification value transmitted in this component. In the case of the transaction identification is not present, a DII with a transaction identification shall be gained;

— **time_out_value_DII:** 32 bits field that indicates the recommended timeout value (in milliseconds) to receive the whole section of the DII in this carousel. When the value is 0xFFFFFFFF, it means that t no recommended timeout value exists;

— **leak_rate:** 22 bits field that indicates the leak rate of the transport buffer of the receiver unit in a unit of 50 byte/s.

Annex C
(informative)

Relation between PMT/EIT descriptor and AIT

The Figure C.1 shows the relation of AIT to the PMT data component descriptor and EIT data contents descriptor in Ginga.

The extension items for Ginga are as follows:

— new assignment of identification value to be used in Ginga (For example: data component ID, transport_id, application_id);

— for component ES that transmits Ginga-J application or for component ES that transmits AIT, selector areas are specified for data component descriptors that store additional information not to be transmitted via AIT;

— selector areas are specified for data contents descriptors in order to store Ginga application details on the basis of the program event;

— in a descriptor, application_identifier_flag is set instead of entry_point_flag in order to store information specifying an application that is an entry point on PMT/EIT.

The additional provisions are for selector areas of data component descriptors as well as data contents descriptors relevant to Ginga and AIT transmission. The rest are supposed to conform to the AIT transmission under MHP1.0.

For the data carousel that does not transmit Ginga application, it is possible to refer to the contents by specifying the locator from the Ginga application.

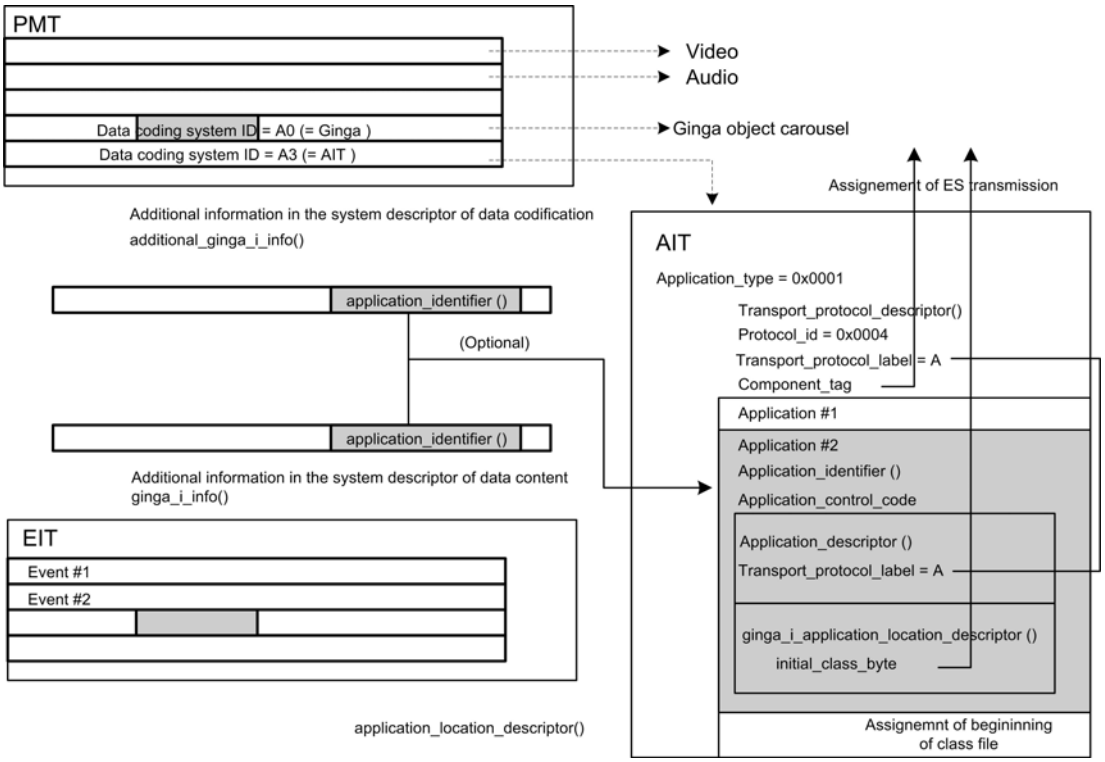

Figure C.1 — Relation of the AIT transmission and data component descriptor

Annex D
(informative)

Supplementary notes on PES transmission

A data_identifier shall be present at the start of the PES_packet_data_byte to identify the data type (see EN 301 192, ATSC DVS 161 and DAVIC 1.4:1998, part 9).

NOTE In Japan, the Ministry of Posts and Telecommunications (Notification Hei 10/260) stipulates the use of data_content_id in PMT.

In order to ensure conformity with DVB, ATSC and DAVIC, the a data_identifier field in this standard contains a copy of the value allocated to the user-defined area in those standards.

The reasons for employing The Independent PES specification as a standard PES transmission method are as follows:

— less size constraints permit more freedom;

— video and audio data are allowed to be separately produced before bing multiplexed for less effort;

— data is allowed to be shared in multiple pieces of video data and audio data for easier access.

Data carousel transmission is based on the U-N downloading (DSM-CC data carousel) stipulated in ISO/IEC 13818-6, to which the following have been added:

— in relation to the module data area, assuming its use for file transmissions, etc., the addition of usage stipulations in the description format conforming to DVB prEN301 192. In addition, the addition of expire, Activationtime, and CompressionType descriptors, which are absent from DVB prEN301 192.

— except for the download services assumed when the original ISO/IEC13818-6 [12] was developed, these stipulations enable the use of data carousel transmissions that are efficient and that have minimal reception processing loads in a wide variety of applications such as multimedia services.

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: Introductio*n

[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, RFC 1321:1992, *The MD5 Message-Digest Algorithm*

[16] MD2, RFC 1319:1992, *The MD2 Message-Digest Algorithm*

[17] TLS1.0, RFC 2246:1999, *The TLS Protocol Version 1*.0

[18] RFC 1590:1994, J.Postel, *Media Type Registration Procedure, RFC 1590, IS*I

[19] The Notification No. 260 of Ministry of Posts and Telecommunications in 1998 – JAPAN

[20] RFC 1877:1995, *PPP Internet Protocol Control Protocol Extensions for Name Server Addresses*

[21] RFC 1954:1996, *Transmission of Flow Labelled IPv4 on ATM Data Links Ipsilon Version 1.0*

[22] RFC 768:1980, *User Datagram Protocol*

[23] RFC 1334: 1992, *PPP Authentication Prot*ocols

[24] RFC 1332: 1992, *The PPP Internet Protocol Control Protocol (IPCP)*

[25] RFC 1034:1987, *Domain names - concepts and faci*lities

[26] RFC 1035:1987, *Domain names - implementation and specification*