



CERTIFICATE

By Authority Of
THE UNITED STATES OF AMERICA
Legally Binding Document

By the Authority Vested By Part 5 of the United States Code § 552(a) and Part 1 of the Code of Regulations § 51 the attached document has been duly INCORPORATED BY REFERENCE and shall be considered legally binding upon all citizens and residents of the United States of America. *HEED THIS NOTICE:* Criminal penalties may apply for noncompliance.



Document Name:

CFR Section(s):

Standards Body: Society of Cable Telecommunications Engineers



Official Incorporator:

THE EXECUTIVE DIRECTOR
OFFICE OF THE FEDERAL REGISTER
WASHINGTON, D.C.



***Society of Cable
Telecommunications
Engineers***

ENGINEERING COMMITTEE
Digital Video Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 65 2008

**SERVICE INFORMATION
DELIVERED OUT-OF-BAND FOR
DIGITAL CABLE TELEVISION**

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or nonmember of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members, whether used domestically or internationally. SCTE assumes no obligations or liability whatsoever to any party who may adopt the Standards. Such adopting party assumes all risks associated with adoption of these Standards or Recommended Practices, and accepts full responsibility for any damage and/or claims arising from the adoption of such Standards or Recommended Practices. Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. Patent holders who believe that they hold patents which are essential to the implementation of this standard have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at <http://www.scte.org>.

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc. 2008

140 Philips Road

Exton, PA 19341

SERVICE INFORMATION DELIVERED OUT-OF-BAND FOR DIGITAL CABLE TELEVISION

TABLE OF CONTENTS

1	PURPOSE, SCOPE AND ORGANIZATION.....	1
1.1	Purpose	1
1.2	Scope	1
1.3	Organization	3
2	REFERENCES	3
3	DEFINITIONS.....	5
3.1	Compliance Notation	5
3.2	Definition of Terms	5
3.3	Acronyms and Abbreviations	6
3.4	Section and Data Structure Syntax Notation	7
3.4.1	Field Sizes	7
4	TABLE STRUCTURE.....	9
4.1	Table ID Ranges and Values	9
4.2	Extensibility	10
4.3	Reserved Fields	11
4.4	Private Table Section Syntax	11
4.4.1	Protocol Version	11
4.4.2	Format Identifier	11
4.4.3	Private Message Body	11
4.4.4	CRC	11
5	TABLE SECTION FORMATS.....	12
5.1	Network Information Table	12
5.1.1	Carrier Definition Subtable (CDS)	13
5.1.2	Modulation Mode Subtable (MMS)	15
5.1.3	Descriptors Count	17
5.2	Network Text Table	17
5.3	Short-form Virtual Channel Table Section	20
5.3.1	Defined Channels Map	21
5.3.2	Virtual Channel Map	22
5.3.3	Inverse Channel Map	26
5.4	System Time Table Section	27

5.5	Master Guide Table (MGT)	28
5.5.1	Restrictions on PID Values	32
5.5.2	Restrictions on Order of Occurrence of Table References	32
5.6	Long-form Virtual Channel Table	33
5.7	Rating Region Table (RRT)	39
5.8	Aggregate Event Information Tables (AEIT)	42
5.9	Aggregate Extended Text Tables (AETT)	45
6	DESCRIPTORS.....	48
6.1	Descriptor Usage	48
6.2	Stuffing Descriptor	48
6.3	AC-3 Audio Descriptor	49
6.4	Caption Service Descriptor	49
6.5	Content Advisory Descriptor	50
6.6	Revision Detection Descriptor	52
6.7	Two Part Channel Number Descriptor	52
6.8	Channel Properties Descriptor	53
6.9	Extended Channel Name Descriptor	55
6.10	Time Shifted Service Descriptor	55
6.11	Component Name Descriptor	56
6.12	Daylight Savings Time Descriptor	57
6.13	User Private Descriptors	57
7	TEXT STRING CODING.....	59
7.1	Multilingual Text String (MTS) Format	59
7.1.1	Mode Byte Definition	62
7.1.2	Format Effectors	62
7.1.3	Default Attributes	63
7.1.4	Mode Zero	63
7.1.5	Supported Characters	64
7.2	Multiple String Structure (MSS)	64
ANNEX A OPERATIONAL PROFILES FOR CABLE SERVICE INFORMATION DELIVERY		
A.1	Operational Profiles	67
A.1.1	Profile 1 – Baseline	67
A.1.2	Profile 2 – Revision Detection	67
A.1.3	Profile 3 – Parental Advisory	67
A.1.4	Profile 4 – Standard Electronic Program Guide Data	67
A.1.5	Profile 5 – Combination	68
A.1.6	Profile 6 – PSIP Only	68

A.2 Profile Definition Tables	68
A.3 Operational Considerations for the use of profiles (Informative)	70
ANNEX B IMPLEMENTATION RECOMMENDATIONS	71
B.1 Implications for Retail Digital Cable-Ready Devices	71
B.2 Channel Number Handling	71
B.3 Processing of Dynamic Changes to Service Information	71
B.4 AEITs May Include Event Information for Inaccessible Channels	71
B.5 Splice Flag Processing	72
ANNEX C SERVICE INFORMATION OVERVIEW AND GUIDE	73
C.1 Table Hierarchy	73
C.2 SI_base PID	77
C.3 Representation of Time	86
ANNEX D PACKET RATES	90
D.1 Maximum cycle times	90
D.2 Maximum Transmission Rates	90
D.3 MINIMUM Transmission Rates	90
ANNEX E DAYLIGHT SAVINGS TIME CONTROL	91
ANNEX F STANDARD HUFFMAN TABLES FOR TEXT COMPRESSION	93
F.1 Character Set Definition	93
F.2 Standard Compression Type 1 Encode/Decode Tables	96
F.3 Standard Compression Type 2 Huffman Encode/Decode Tables	103

List of Figures

Figure 1.1 A Framework for the Extended Channel Service Information Stream	2
Figure C.1 Hierarchy of Table Sections -- Profiles 1 and 2	73
Figure C.2 Hierarchy of Table Sections -- Profile 3	74
Figure C.3 Hierarchy of Table Sections -- Profile 4	75
Figure C.4 Hierarchy of Table Sections -- Profile 5	76
Figure C.5 Hierarchy of Table Sections -- Profile 6	77
Figure C.6 An instance of a Rating Region Table	82
Figure C.7 Example AEIT-0	84
Figure C.8 AEIT data structure	85
Figure C.9 Structure of AETT	85

List of Tables

Table 3.1 Field Sizes Example	8
Table 4.1 Table ID Ranges and Values for Out-of-Band Transport	9
Table 4.2 Network private table section format	10
Table 5.1 Network Information Table section format	13
Table 5.2 Network Information Table Subtype	13
Table 5.3 CDS record format	14
Table 5.4 Spacing Unit	15
Table 5.5 Frequency Unit	15
Table 5.6 MMS record format	15
Table 5.7 Transmission System	16
Table 5.8 Inner Coding Mode	16
Table 5.9 Modulation Format	17
Table 5.10 Network Text Table section format	18
Table 5.11 Network Text Table Subtype	18
Table 5.12 Source Name Subtable format	19
Table 5.13 Short-form Virtual Channel Table section format	20
Table 5.14 S-VCT Table Subtypes	21
Table 5.15 DCM structure format	21
Table 5.16 VCM structure format	22
Table 5.17 Virtual channel record format	23
Table 5.18 Path Select	24
Table 5.19 Transport Type	24
Table 5.20 Channel Type	24
Table 5.21 Video Standard	26
Table 5.22 ICM structure format	27
Table 5.23 System Time Table section format	28
Table 5.24 Master Guide Table section format	30
Table 5.25 MGT Table Types	31
Table 5.26 Long-form Virtual Channel Table section format	34
Table 5.27 Major and Minor Channel Number Field Coding	36
Table 5.28 Modulation Modes	37
Table 5.29 Path Select	38
Table 5.30 Service Types	39
Table 5.31 Rating Region Table section format	40
Table 5.32 Rating Regions	41
Table 5.33 Aggregate Event Information Table format	44
Table 5.34 ETM_present	45
Table 5.35 Aggregate Extended Text Table format	46
Table 5.36 ETM ID	47
Table 6.1 Descriptor Usage	48
Table 6.2 Caption Service Descriptor format	49
Table 6.3 Content Advisory Descriptor format	51
Table 6.4 Revision Detection Descriptor format	52
Table 6.5 Two-part Channel Number Descriptor format	53
Table 6.6 Channel Properties Descriptor format	54

Table 6.7 Extended Channel Name Descriptor format	55
Table 6.8 Time Shifted Service Descriptor format	56
Table 6.9 Component Name Descriptor format	56
Table 6.10 Daylight Savings Time Descriptor format	57
Table 7.1 Text String Coding Format in Tables	59
Table 7.2 Text String Coding Format in Descriptors	59
Table 7.3 Mode Byte Encoding	61
Table 7.4 Multilingual text string format	61
Table 7.5 Format Effector Function Codes	63
Table 7.6 Encodings of Columns 8 and 9 of Mode Zero Latin Character Set	63
Table 7.7 Multiple String Structure	65
Table 7.8 Compression Types	65
Table 7.9 Modes	66
Table A.1 Usage of Table Sections in Various Profiles	69
Table A.2 Usage of Descriptors in Various Profiles	70
Table C.1 Example Master Guide Table content	81
Table C.2 Example Revised Master Guide Table content	81
Table C.3 Receiver Behavior with hidden and hide_guide attributes	86
Table D.1 Maximum cycle time for the STT, MGT, S-VCT, L-VCT and RRT	90
Table D.2 Maximum rate for each packet stream	90
Table D.3 Minimum rate for each packet stream	90
Table E.1 Basic Use of Daylight Savings Fields Through the Year	92
Table F.1 Characters with Special Definitions	93
Table F.2 Decode Table Format	94
Table F.3 Decode Tree Format	95
Table F.4 English-language Program Title Encode Table	96
Table F.5 English-language Program Title Decode Table	100
Table F.6 English-language Program Description Encode Table	103
Table F.7 English-language Program Description Decode Table	108

SERVICE INFORMATION DELIVERED OUT-OF-BAND FOR DIGITAL CABLE TELEVISION

1 PURPOSE, SCOPE AND ORGANIZATION

1.1 Purpose

This document defines a standard for Service Information (SI) delivered out-of-band on cable. This standard is designed to support “navigation devices” on cable. The current specification defines the syntax and semantics for a standard set of tables providing the data necessary for such a device to discover and access digital and analog services offered on cable.

1.2 Scope

This specification defines SI tables delivered via an out-of-band path to support service selection and navigation by digital cable set-top boxes and other “digital cable-ready” devices. The SI tables defined in this standard are formatted in accordance with the Program Specific Information (PSI) data structures defined in MPEG-2 Systems [1].

The formal definition of “digital cable-ready” has a scope broader than that of the current standard. The formal definition includes requirements related to navigation and service selection, demodulation and decoding, video format decoding, Emergency Alert handling, and other aspects. The current specification supports, primarily, the navigation and service selection function for services delivered in the clear, as well as those subject to conditional access.

This specification does not address the Electronic Program Guide application itself or any user interface which might deal with the presentation and application of the Service Information.

1.3 Digital Cable Ready Device

A digital cable-ready device can take the form of a cable set-top box, a computer, a television, or a convergence of these. Devices such as digital video recorders may also be cable-ready. A digital cable-ready device capable of processing access controlled digital services supports an interface to a conditional access module. As used here, the term “Host” refers to the capability to support an interface to a standard Point-of-Deployment (POD)¹ security module.

SI data delivered out-of-band is transported in accordance with the Extended Channel Interface defined in SCTE 28. To obtain access to a POD Extended Channel Interface, the digital cable-ready device must act as a Host to a POD security module. The Extended Channel

¹ The Point-of-Deployment module is also known as a CableCARD™ device.

interface is designed to present the needed SI data to the Host. This data can be used by the Host for channel navigation, construction of electronic program guides and other associated functions.

Figure 1.1 is a high-level block diagram illustrating the POD module to Host interface via the Extended Channel interface. The Host is responsible for providing a standard receiver/QPSK demodulator function for the POD module. The choice of transport format of bits coming across from the receiver/QPSK demodulator to the POD module is by mutual agreement between the POD and the cable head-end equipment. The transport format of data traveling between the Host and POD module on the Extended Channel interface conforms to standards defined in [10].

The POD module may perform various transport, filtering, and error checking/correction functions on the out-of-band data stream as depicted by the box labeled “Transport Processing, Filtering, and Routing.” As described in [12], the Host may request from the POD module to open one or several “flows” in which to receive PSI sections taken from the cable out-of-band data stream. Each flow is associated with a PID value, in accordance with MPEG-2 Transport Stream concepts.

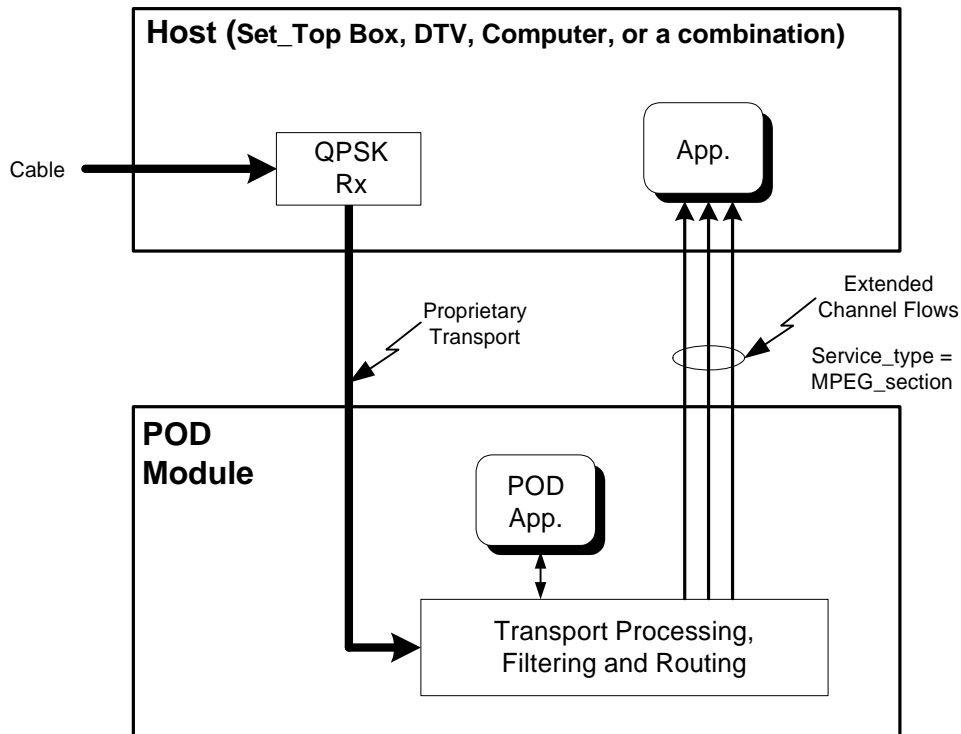


Figure 1.1 A Framework for the Extended Channel Service Information Stream

Data flowing to the Host from the POD module that is associated with Service_type=MPEG_section is required to be in the form of MPEG PSI data structures. However, data delivered into the POD from cable out-of-band may or may not be organized in a Transport Stream compliant with ISO/IEC 13818-1. In other words, PID values associated with MPEG-2 tables on the Extended Channel interface *may or may not* correspond to MPEG-2 Transport Stream packet header PID values from the cable out-of-band.

Independent of the fact that out-of-band data may reach the POD module via a proprietary method, the data structures delivered across the Extended Channel shall be formatted as MPEG-2 table sections. Like table sections carried in an MPEG-2 Transport Stream, each is associated with a PID value.

1.4 Organization

The sections of this document are organized as follows:

- **Section 1** — Provides this general introduction.
- **Section 2** — Lists applicable documents.
- **Section 3** — Provides a list of acronyms and abbreviations used in this document.
- **Section 4** — Describes the basic structure of sections.
- **Section 5** — Describes formats of sections carried in the Base PID.²
- **Section 6** — Explains descriptors applicable to the tables defined in this standard.
- **Section 7** — Describes multilingual character string coding.
- **Annex A** — Defines profiles of choice for cable operator compliance with this standard.
- **Annex B** — Discusses recommendations for receiver implementations.
- **Annex C** — Provides an overview of tables defined in this Service Information standard.
- **Annex D** — Specifies packet rates for delivery of SI data
- **Annex E** — Defines the daylight savings time control fields in the System Time Table.
- **Annex F** — Defines the standard Huffman tables used for text compression.

2 REFERENCES

The following documents are applicable to this Service Information standard:

1. ITU-T Rec. H.222.0 | ISO/IEC 13818-1:2007, Information Technology — Generic coding of moving pictures and associated audio information — Part 1: systems.
2. ITU-T Rec. H.262 | ISO/IEC 13818-2:2000, Information Technology — Generic coding of moving pictures and associated audio information — Part 2: Video.
3. ATSC Standard A/52 (1995), Digital Audio Compression (AC-3).
4. ATSC Standard A/53 (1995), ATSC Digital Television Standard.

² The Base PID is the PID associated with the “base” Service Information tables. In this protocol, the base_PID is fixed at 0x1FFC. Refer to Table 4.1.

5. deleted during 2007 update
6. ANSI/SCTE 07 2006, Digital Transmission Standard for Cable Television
7. ATSC Standard A/65C (2006), Program and System Information Protocol for Terrestrial Broadcast and Cable, Rev C with Amendment No. 1.
8. deleted during 2007 update
9. deleted during 2007 update
10. ANSI/SCTE 28 2007, Host-POD Interface Standard.
11. ANSI/SCTE 18 2007, Emergency Alert Messaging for Cable.
12. deleted during 2007 update
13. ISO 639, Code for the Representation of Names of Languages, 1988.
14. deleted during 2007 update
15. ISO/IEC 10646-1:1993, Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.
16. deleted during 2007 update
17. deleted during 2007 update
18. deleted during 2007 update
19. deleted during 2007 update
20. ISO/CD 13522-2:1993, Information Technology — Coded representation of multimedia and hypermedia information objects — Part 1: Base notation.
21. ISO/IEC 8859, Information Processing — 8-bit Single-Octet Coded Character Sets, Parts 1 through 10.
22. ITU-T Rec. J.83: 1997, Annex B, Digital multi-programme systems for television, sound and data services for cable distribution.
23. deleted during 2007 update
24. CEA-708-C Digital Television (DTV) Closed Captioning, Consumer Electronics Association, 2006.
25. CEA-766-B, U.S. and Canadian Rating Region Tables (RRT) and Content Advisory Descriptors for Transport of Content Advisory Information Using ATSC Program and System Information Protocol (PSIP), Consumer Electronics Association, 2006.
26. CEA-608-C, Line 21 Data Services, Consumer Electronics Association, 2005.

3 DEFINITIONS

3.1 Compliance Notation

As used in this document, “*shall*” denotes a mandatory provision of the standard. “*Should*” denotes a provision that is recommended but not mandatory. “*May*” denotes a feature whose presence does not preclude compliance, that may or may not be present as optional for the implementers.

3.2 Definition of Terms

The following terms are used throughout this document:

Conditional Access: The control and security of subscriber access to cable or broadcast services and events in the form of video, data and voice communications.

Host: A device capable of supporting a POD module by implementing the interface protocol defined in SCTE 28 [10]. SCTE 28 defines the Extended Channel data path through which the SI tables defined in this standard are passed.

navigation: The process of selection and movement among analog and digital services offered on the cable network. The service information tables defined in this protocol assist in the navigation process by providing physical service locations, channel names and numbers for user reference. Those tables supporting electronic program guides also assist the navigation process.

program element: A generic term for one of the elementary streams or other data streams that may be included in a program.

program: A collection of program elements. Program elements may be elementary streams. Program elements need not have any defined time base. Those that do have a common time base are intended for synchronized presentation. The term *program* is also used in the context of a “television program” such as a scheduled daily news broadcast. The distinction between the two usages should be understood by context.

region: as used in this document, a region is a geographical area consisting of one or more countries.

section or table section: A data structure comprising a portion of an *ISO/IEC 13818-1*-defined table, such as the Program Association Table (PAT), Conditional Access Table (CAT), or Program Map Table (PMT). The term conforms to MPEG terminology. All sections begin with the `table_ID` and end with the `CRC_32` field. Sections are carried in Transport Stream packets in which the starting point within a packet payload is indicated by the `pointer_field` mechanism defined in the *ISO/IEC 13818-1 Systems* document. The Network Information Table, for example, defines portions of several types of tables.

service: *ISO/IEC 13818-1* uses the term *program* to refer to a collection of program elements with no regard to time. In this Service Information standard, the term *service* is used in this same context to denote a collection of elementary components. Usage of the term *service* clarifies certain discussions that also involve the notion of the term *program* in its traditional meaning — for example, in the statement, “A video service carries a series of programs.” In a

broader sense, *service* is also intended for multimedia services of video, voice and data, as these services become prevalent.

stream: An ordered series of bytes. The usual context for the term *stream* involves specification of a particular PID (such as the “Program Map PID stream”), in which case the term indicates a series of bytes extracted from the packet multiplex from packets with the indicated PID value.

3.3 *Acronyms and Abbreviations*

The following acronyms and abbreviations are used within this specification:

AEIT	Aggregate Event Information Table
AETT	Aggregate Extended Text Table
ATSC	Advanced Television Standards Committee
BMP	Basic Multilingual Plane
bslbf	bit serial, leftmost bit first
CAT	Conditional Access Table
CC	Closed Caption
CDS	Carrier Definition Subtable
CRC	Cyclic Redundancy Check
DCM	Defined Channels Map
DTV	Digital Television
ECM	Entitlement Control Message
EMM	Entitlement Management Message
ETSI	European Telecommunications Standards Institute
GPS	Global Positioning System
ICM	Inverse Channel Map
ITU	International Telecommunications Union
L-VCT	Long-form Virtual Channel Table
LSB	Least Significant Bit
MGT	Master Guide Table
MMS	Modulation Mode Subtable
MPEG	Moving Picture Experts Group
MPAA	Motion Picture Association of America
MSB	Most Significant Bit
MSS	Multiple String Structure
MTS	Multi-lingual Text String
NTSC	National Television Standards Committee
NVOD	Near Video On Demand
OOB	Out-of-band
PAT	Program Association Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Packet Identifier
PMT	Program Map Table
POD	Point of Deployment
PSIP	Program and System Information Protocol

PTC	Physical Transmission Channel
PTS	Presentation Time Stamp
rpchof	remainder polynomial coefficients, highest order first
RRT	Rating Region Table
S-VCT	Short-form Virtual Channel Table
SCTE	Society of Cable Telecommunications Engineers
SI	Service Information
SNS	Source Name Subtable
TS	Transport Stream
UTC	Coordinated Universal Time ³
uimsbf	unsigned integer, most significant bit first
VCM	Virtual Channel Map

3.4 Section and Data Structure Syntax Notation

This document contains symbolic references to syntactic elements. These references are typographically distinguished by the use of a different font (e.g., *restricted*), may contain the underscore character (e.g., *sequence_end_code*) and may consist of character strings that are not English words (e.g., *dynrng*).

The formats of sections and data structures in this document are described using a C-like notational method employed in *ISO/IEC 13818-1*. Extensions to this method are described in the following sections.

3.4.1 Field Sizes

Each data structure is described in a table format wherein the size in bits of each variable within that section is listed in a column labeled “Bits.” The column adjacent to the Bits column is labeled “Bytes” and indicates the size of the item in bytes. For convenience, several bits within a particular byte or multi-byte variable may be aggregated for the count. An example follows:

³ Since unanimous agreement could not be achieved by the ITU on using either the English word order, CUT, or the French word order, TUC, a compromise to use neither was reached.

Table 3.1 Field Sizes Example

	Bits	Bytes	Format
foo_section(){			
section_syntax_indicator	1	1	
...			
if (section_syntax_indicator) {			
table_extension	16	(2)	uimsbf
reserved	2	(1)	bslbf
version_number	5		uimsbf
current_next_indicator	1		bslbf {next, current}
...			
}			
...			

In the byte count column, items that are conditional (because they are within a loop or conditional statement) are in parentheses. Nested parentheses are used if the loops or conditions are nested.

4 TABLE STRUCTURE

This section describes details of the structure of MPEG-2 tables defined in this standard.

Tables and table sections defined in this Service Information standard are structured in the same manner used for carrying *ISO/IEC 13818-1* -defined PSI tables. The MPEG-defined 32-bit CRC is required.

4.1 Table ID Ranges and Values

Table 4.1 defines table_ID ranges and values for tables defined in MPEG and in this standard.

Table 4.1 Table ID Ranges and Values for Out-of-Band Transport

Table ID Value (hex)	Tables	PID	Ref.
	ISO/IEC 13818-1 Sections:		
0x00	Program Association Table (PAT)	0	Ref. [1]
0x01	Conditional Access Table (CAT)	1	Ref. [1]
0x02	TS Program Map Table (PMT)	per PAT	Ref. [1]
0x03-0x3F	[ISO Reserved]		
	User Private Sections:		
0x40-0x7F	[User Private for other systems]		
0x80-0xBF	[SCTE User Private]		
	Other Standards:		
0xC0-0xC1	[Used in other standards]		
	Service Information Tables:		
0xC2	Network Information Table (NIT)	0x1FFC	Sec. 5.1
0xC3	Network Text Table (NTT)	0x1FFC	Sec. 5.2
0xC4	Short-form Virtual Channel Table (S-VCT)	0x1FFC	Sec. 5.3
0xC5	System Time Table (STT)	0x1FFC	Sec. 5.4
0xC6	[Used in other standards]	-	-
0xC7	Master Guide Table (MGT)	0x1FFC	Sec. 5.5
0xC8	Reserved	-	-
0xC9	Long-form Virtual Channel Table (L-VCT)	0x1FFC	Sec. 5.6
0xCA	Rating Region Table (RRT)	0x1FFC	Sec. 5.7
0xCB-0xD5	[Used in ATSC]	-	-
0xD6	Aggregate Event Information Table (AEIT)	per MGT	Sec. 5.8
0xD7	Aggregate Extended Text Table (AETT)	per MGT	Sec. 5.9
0xD8	Cable Emergency Alert Message	0x1FFC	Ref. [11]
0xD9-0xFE	[Reserved for future use or by other standards]	-	-

Table sections defined in this Service Information standard, and any created as user extensions to it are considered “private” with respect to *ISO/IEC 13818-1*. Table section types 0x80 through 0xBF are user-defined (outside the scope of this Service Information standard).

The maximum total length of any table section defined in this standard is 1024 bytes, except for the MGT, L-VCT, AEIT and AETT, each of which has a maximum total length of

4096 bytes. This total includes table_ID, CRC, and all fields contained within the specific table section.

4.2 Extensibility

This Service Information standard defines a number of tables and table sections. The Service Information standard is designed to be extensible via the following mechanisms:

1. **Reserved Fields:** Fields in this Service Information standard marked reserved are reserved for use either when revising this standard, or when another standard is issued that builds upon this one. See Section 4.4 below.
2. **Standard Table Types:** As indicated in Table 4.1, table_ID values in the range 0xCE through 0xFE are reserved for use either when revising this Service Information standard, or when another standard is issued that builds upon this one.⁴
3. **User Private Table Types:** As indicated in Table 4.1, table_id values in the range 0x80 through 0xBF are reserved for “user private” use. The format of user private tables carried in the Network PID shall conform to the syntax described in Table 4.2.
4. **User Private Descriptors:** Privately defined descriptors may be placed at designated locations throughout the table sections described in this Service Information standard. Ownership of one or more user private descriptors is indicated by the presence of an MPEG registration_descriptor() preceding the descriptor(s).

Table 4.2 Network private table section format

	Bits	Bytes	Format
network_private_table section(){			
private_table_ID	8	1	uimsbf (0x80 <= table_ID <= 0xBF)
section_syntax_indicator	1	2	bslbf
zero	1		bslbf
reserved	2		bslbf
section_length	12		uimsbf
if (section_syntax_indicator==1) {			
table_extension	16	(2)	uimsbf
reserved	2	(1)	bslbf
version_number	5		uimsbf
current_next_indicator	1		bslbf {next, current}
section_number	8	(1)	uimsbf
last_section_number	8	(1)	uimsbf
}			
zero	3	1	bslbf
protocol_version	5		see Section 4.4.1
format_identifier	32	4	uimsbf
private_message_body()	N*8	N	
CRC_32	32	4	rpchof
}			

⁴ Note: Assignment of table_ID values in the 0xCE to 0xFE range requires coordination between ATSC and SCTE.

4.3 *Reserved Fields*

reserved — Fields in this Service Information standard marked “reserved” shall not be assigned by the user, but shall be available for future use. Hosts are expected to disregard reserved fields for which no definition exists that is known to that unit. Fields marked “reserved” shall be set to “1” until such time as they are defined and supported.

zero — Indicates the bit or bit field shall be “0”.

4.4 *Private Table Section Syntax*

Table 4.2 defines the syntax for user private table sections. The MPEG-defined CRC is required. Refer to *ISO/IEC 13818-1* for definition of MPEG-standard fields.

private_table_ID — The value of `table_ID` in private table sections shall be in the range 0x80 through 0xBF.

4.4.1 **Protocol Version**

protocol_version — A 5-bit unsigned integer field whose function is to allow, in the future, any defined table type to carry parameters that may be structured fundamentally differently from those defined in the current protocol. At present, all defined table section types in this protocol are defined for `protocol_version` zero only. Nonzero values of `protocol_version` may only be processed by Receivers designed to accommodate the later versions as they become standardized.

4.4.2 **Format Identifier**

format_identifier — A 32-bit unsigned integer value which unambiguously identifies the entity defining this `network_private_table_section()` syntax. Values for `format_identifiers` shall be obtained from SCTE.

4.4.3 **Private Message Body**

private_message_body() — A data structure defined by the private entity identified by `format_identifier`.

4.4.4 **CRC**

CRC_32 — The 32-bit CRC value defined in [1] for PSI sections. The MPEG-2 CRC shall be checked in the POD, and only messages that pass the CRC check shall be forwarded to the Host. The Host shall not check the CRC.

5 TABLE SECTION FORMATS

The following sections define the formats of table sections as they are delivered across an Extended Channel Interface.

5.1 *Network Information Table*

Sections of the Network Information Table shall be associated on the POD-Host interface with PID value 0x1FFC, the *SI_base* PID. This table delivers sections of non-textual tables applicable system-wide. The table types included are the Carrier Definition Subtable (CDS) and the Modulation Mode Subtable (MMS).

Table 5.1 shows the format of the Network Information Table section.

table_ID — The *table_ID* of the Network Information Table section shall be 0xC2.

first_index — An 8-bit unsigned integer number in the range one to 255 that indicates the index of the first record to be defined in this table section. If more than one record is provided, the additional records define successive table entries following *first_index*. The value zero is illegal and shall not be specified.

number_of_records — An 8-bit unsigned integer number that specifies the number of records being defined in this table section. The maximum is limited by the maximum allowed length of the table section.

transmission_medium — This 4-bit field shall be set to zero (0x0).

Table 5.1 Network Information Table section format

	Bits	Bytes	Format
network_info_table_section(){			
table_ID	8	1	uimsbf value 0xC2
zero	2	2	bslbf
reserved	2		bslbf
section_length	12		uimsbf
zero	3	1	bslbf
protocol_version	5		Sec. 4.4.1
first_index	8	1	uimsbf range 1-255
number_of_records	8	1	uimsbf
transmission_medium	4	1	uimsbf
table_subtype	4		uimsbf see Table 5.2
for (i=0; i<number_of_records; i++) {			
if (table_subtype==CDS) {			
CDS_record()		((5))	
}			
if (table_subtype==MMS) {			
MMS_record()		((6))	
}			
descriptors_count	8	(1)	uimsbf range 0-255
for (i=0; i<descriptors_count; i++) {			
descriptor()	*	((*))	optional
}			
}			
for (i=0; i<N; i++) {			
descriptor()	*	(*)	optional
}			
CRC_32	32	4	rpchof
}			

table_subtype — A 4-bit value that defines the type of table delivered in the table section. One instance of a Network Information Table section can define entries within at most one type of table. The `table_subtype` parameter is defined in Table 5.2.

Table 5.2 Network Information Table Subtype

table_subtype	meaning
0	invalid
1	CDS — Carrier Definition Subtable
2	MMS — Modulation Mode Subtable
3-15	Reserved

The receiver shall discard a Network Information Table section with `table_subtype` indicating an unknown or unsupported `table_subtype`.

5.1.1 Carrier Definition Subtable (CDS)

Table 5.3 defines the structure of the `CDS_record()`. Each CDS defines a set of carrier frequencies. A full frequency plan table shall be constructed from one or more `CDS_record()` structures, each defining a starting frequency, a number of carriers, and a frequency spacing for carriers in this group.

The specified carrier represents the nominal center of the spectral band for all modulation methods, including analog. Carrier frequencies in the table thus represent the data carrier frequency for digital transmissions modulated using QAM or PSK.⁵

Each CDS_record represents a definition of N carriers. The first_index parameter reflects the index in a flat space between 1 and 255, representing the first carrier in the CDS_record. Starting from the first CDS_record defining carriers C₁, C₂, C₃, ..., C_N, where N = number_of_carriers, the carrier index for C_I is equal to first_index + I - 1. If the table section includes more than one CDS_record(), the carrier index of the second CDS_record would be first_index plus the number of carriers defined in the first CDS_record(), namely, first_index + number_of_carriers. References to the Carrier Definition Subtable, such as the CDS_reference in the virtual_channel() of Table 5.17, are to the carrier index (a carrier defined within a CDS_record()), between 1 and N, where N is normally much smaller than 255. These references are *not* to the index of a CDS_record() itself, which is sequenced from first_index and is not reset to 1 until it exceeds 255.

Note that the carriers, as defined by one or more CDS_record(s), may or may not end up sorted in the order of increasing carrier frequency. Certain frequency plans may be specified by overlapping two or more CDS_record(s), each of which defines equally-spaced carriers.

Note also that carriers may be defined that are currently not in use. To facilitate the compressed delivery format, defined carriers may not reflect reality. An example: carriers at 1, 2, 4, 5, 7, 8 MHz could be defined as eight carriers at 1MHz spacing (3 MHz and 6 MHz do not really exist, or are not currently in use).

Table 5.3 CDS record format

	Bits	Bytes	Format
CDS_record(){			
number_of_carriers	8	1	uimsbf
spacing_unit	1	2	bslbf see Table 5.4
zero	1		bslbf
frequency_spacing	14		uimsbf range 1-16,383 units of 10 or 125kHz
frequency_unit	1	2	bslbf see Table 5.5
first_carrier_frequency	15		uimsbf range 0-32,767 units of 10 or 125kHz
}			

number_of_carriers — An unsigned integer in the range 1 to 255 that represents the number of carriers whose frequency is being defined by this CDS_record().

spacing_unit — A 1-bit field identifying the units for the frequency_spacing field. Table 5.4 defines the coding for spacing_unit.

⁵ Note that transmission systems using VSB modulation transmit spectra are not symmetrical about the carrier or pilot tone. Acquisition of a VSB-modulated signal involves computation of the pilot tone (or in analog VSB, the picture carrier) location relative to the center of the band. For example, for the ATSC Digital Television Standard (Ref. [4]), where the channel bandwidth is 6 MHz, the pilot tone is located 310 kHz above the lower edge of the channel, or 2.690 MHz below the specified center of the band. Similarly, for analog NTSC, the picture carrier is 1.25 MHz above the lower edge of the channel, or 1.75 MHz below the specified center of the band.

Table 5.4 Spacing Unit

spacing_unit	meaning
0	10 kHz spacing
1	125 kHz spacing

frequency_spacing — A 14-bit unsigned integer number in the range one to 16,383 that defines the frequency spacing in units of either 10 kHz or 125 kHz, depending upon the value of the spacing_unit parameter. If spacing_unit is zero, indicating 10 kHz, then a value of one indicates 10 kHz spacing; two indicates 20 kHz, and so on. If the number_of_carriers field is one, the frequency_spacing field is ignored. The maximum frequency spacing that can be represented is $(2^{14} - 1) * 125 \text{ kHz} = 2047.875 \text{ MHz}$. The minimum frequency spacing is 10 kHz.

frequency_unit — A 1-bit field identifying the units for the first_carrier_frequency field. Table 5.5 defines the coding for frequency_unit.

Table 5.5 Frequency Unit

Frequency_unit	meaning
0	10 kHz units
1	125 kHz units

first_carrier_frequency — A 15-bit unsigned integer number in the range 0 to 32,767 that defines the starting carrier frequency for the carriers defined in this group, in units of either 10 kHz or 125 kHz, depending on the value of frequency_unit. If only one carrier is defined for the group, the first_carrier_frequency represents its frequency. When the frequency_unit indicates 125 kHz, the first_carrier_frequency can be interpreted as a fractional frequency (1/8 MHz) in the least-significant 3 bits, and an integer number of megahertz in the upper 12 bits. The range of frequencies that can be represented is 0 to $(2^{15} - 1) * 125 \text{ kHz} = 4095.875 \text{ MHz}$.

5.1.2 Modulation Mode Subtable (MMS)

Table 5.6 defines the structure of the MMS_record().

Table 5.6 MMS record format

	Bits	Bytes	Format
MMS_record(){			
transmission_system	4	1	uimbsf see Table 5.7
inner_coding_mode	4		uimbsf see Table 5.8
split_bitstream_mode	1	1	bslbf {no, yes}
zero	2		bslbf
modulation_format	5		uimbsf see Table 5.9
zero	4	4	bslbf
symbol_rate	28		uimbsf units: symbols per sec.
}			

transmission_system — A 4-bit field that identifies the transmission standard employed for the waveform defined by this MMS record. Table 5.7 defines the coding for transmission_system.

Table 5.7 Transmission System

transmission_system	meaning
0	unknown — The transmission system is not known.
1	Reserved (ETSI)
2	ITU-T annex B — The transmission system conforms to the ITU North American standard specified in Annex B of ITU Rec. J.83 [22].
3	Defined for use in other systems
4	ATSC — The transmission system conforms to the ATSC Digital Television Standard [4].
5-15	Reserved (satellite)

inner_coding_mode — A 4-bit field that indicates the coding mode for the inner code associated with the waveform described in this MMS record. The following values are currently defined: 5/11, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, and 7/8. Coding of the inner_coding_mode field is shown in Table 5.8.

modulation_format — A 5-bit field that defines the basic modulation format for the carrier. Table 5.9 defines the parameter.

Table 5.8 Inner Coding Mode

inner_coding_mode	meaning
0	rate 5/11 coding
1	rate 1/2 coding
2	Reserved
3	rate 3/5 coding
4	Reserved
5	rate 2/3 coding
6	Reserved
7	rate 3/4 coding
8	rate 4/5 coding
9	rate 5/6 coding
10	Reserved
11	rate 7/8 coding
12-14	Reserved
15	none — indicates that the waveform does not use concatenated coding

Table 5.9 Modulation Format

modulation_format	meaning
0	unknown — The modulation format is unknown.
1	QPSK — The modulation format is QPSK (Quadrature Phase Shift Keying).
2	BPSK — The modulation format is BPSK (Binary Phase Shift Keying).
3	OQPSK — The modulation format is offset QPSK.
4	VS8 — The modulation format is 8-level VSB (Vestigial Sideband).
5	VS16 — The modulation format is 16-level VSB.
6	QAM 16 — Modulation format 16-level Quadrature Amplitude Modulation (QAM).
7	QAM 32 — 32-level QAM
8	QAM 64 — 64-level QAM
9	QAM 80 — 80-level QAM
10	QAM 96 — 96-level QAM
11	QAM 112 — 112-level QAM
12	QAM 128 — 128-level QAM
13	QAM 160 — 160-level QAM
14	QAM 192 — 192-level QAM
15	QAM 224 — 224-level QAM
16	QAM 256 — 256-level QAM
17	QAM 320 — 320-level QAM
18	QAM 384 — 384-level QAM
19	QAM 448 — 448-level QAM
20	QAM 512 — 512-level QAM
21	QAM 640 — 640-level QAM
22	QAM 768 — 768-level QAM
23	QAM 896 — 896-level QAM
24	QAM 1024 — 1024-level QAM
25-31	Reserved

symbol_rate — A 28-bit unsigned integer field that indicates the symbol rate in symbols per second associated with the waveform described in this MMS record.

5.1.3 Descriptors Count

descriptors_count — An 8-bit unsigned integer value in the range 0 to 255 representing the number of descriptor blocks to follow.

descriptor() — The table section may include at its end one or more structures of the form tag, length, data. The number of descriptors present is determined indirectly by processing the **section_length** field. Descriptors are defined in Section 6.

5.2 Network Text Table

Sections of the Network Text Table shall be associated on an Extended Channel Interface with PID value 0x1FFC, the **SI_base** PID. This table delivers sections of textual tables applicable system-wide. Each instance of Network Text Table is associated with a language, as such the textual information may be provided multi-lingually. The Network Text Table delivers the Source Name Subtable (SNS).

Table 5.10 shows the format of the Network Text Table.

The Network Text Table carries Multilingual Text Strings, formatted as defined in Section 7.2. Text strings included in the Network Text Table shall not include format effectors (defined in Section 7.1.2). If format effectors are present in a text block, the Host is expected to disregard them.

Table 5.10 Network Text Table section format

	Bits	Bytes	Format
network_text_table_section(){			
table_ID	8	1	uimsbf value 0xC3
zero	2	2	bslbf
reserved	2		bslbf
section_length	12		uimsbf
zero	3	1	
protocol_version	5		see Sec. 4.4.1
ISO_639_language_code	24	3	per ISO 639.2/B
transmission_medium	4	1	uimsbf
table_subtype	4		uimsbf see Table 5.11
if (table_subtype==SNS) {			
source_name_subtable()	*	(*)	
}			
for (i=0; i<N; i++) {			
descriptor()	*	(*)	optional
}			
CRC_32	32	4	rpchof
}			

table_ID — The table_ID of the Network Text Table section shall be 0xC3.

ISO_639_language_code — A 3-byte language code per ISO 639.2/B defining the language associated with the text carried in this Network Text Table. The ISO_639_language_code field contains a three-character code as specified by ISO 639.2/B. Each character is coded into 8 bits according to ISO 8859-1 (ISO Latin-1) and inserted, in order, into the 24-bit field. The value 0xFFFFFFFF shall be used in case the text is available in one language only. The value 0xFFFFFFFF shall represent a “wild card” match when filtering by language.

transmission_medium — This 4-bit field shall be set to zero (0x0).

table_subtype — A 4-bit value that defines the type of table delivered in the table section. One instance of a Network Text Table section can define entries within at most one type of table. The table_subtype parameter is defined in Table 5.11.

Table 5.11 Network Text Table Subtype

table_subtype	meaning
0	invalid
1-5	Reserved
6	SNS — Source Name Subtable
7-15	Reserved

A Host shall discard a Network Text Table section with `table_subtype` indicating an unknown or unsupported value.

The SNS can provide a textual name associated with each service defined in the Short-form Virtual Channel Table, by reference to its `source_ID`. The format of the `source_name__subtable()` is given in Table 5.12.

number_of_SNS_records — An unsigned 8-bit integer number in the range 1 to 255 that specifies the number of records being defined in this table section.

application_type — A Boolean flag, when set, indicates that the name string being defined is for an application of the given `application_ID`. When the flag is clear, the name string being defined is for a source of the given `source_ID`. Support for application-type virtual channels is optional. Hosts not supporting application-type virtual channels may disregard name strings associated with these VC. Support for application-type virtual channels is beyond the scope of this standard.

application_ID — A 16-bit unsigned integer value identifying the application associated with the name string that follows. This field may be disregarded by Hosts not supporting application-type virtual channels.

source_ID — A 16-bit unsigned integer value identifying the programming source associated with the source name to follow.

name_length — An unsigned 8-bit integer number in the range 1 to 255 that defines the number of bytes in the `source_name()` that follows.

source_name() — A Multilingual Text String defining the name of the source or application, formatted according to the rules defined in Section 7.1.

Table 5.12 Source Name Subtable format

	Bits	Bytes	Format
source_name_subtable(){			
number_of_SNS_records	8	1	uimsbf range 1-255
for (i=0; i<number_of_SNS_records; i++) {			
application_type	1	(1)	bslbf {false, true}
zero	7		bslbf
if (application_type) {			
application_ID	16	((2))	uimsbf
} else {			
source_ID	16	((2))	uimsbf
}			
name_length	8	(1)	size of source_name() (L)
source_name()	L*8	(L)	multilingual text
SNS_descriptors_count	8	(1)	uimsbf range 0-255
for (i=0; i<SNS_descriptors_count; i++) {			
descriptor()	*	((*))	
}			
}			
}			

SNS_descriptors_count — An unsigned 8-bit integer number, in the range 0 to 255, that defines the number of descriptors to follow.

descriptor() — The table section may include, at its end, one or more structures of the form tag, length, data. The number of descriptors present is determined indirectly by processing the section_length field. Descriptors are defined in Section 6.

5.3 Short-form Virtual Channel Table Section

The Short-form Virtual Channel Table sections deliver portions of the Virtual Channel Map (VCM), the Defined Channels Map (DCM) and the Inverse Channel Map (ICM). Sections of the Short-form Virtual Channel Table shall be associated on an Extended Channel Interface with PID value 0x1FFC, the SI_base PID.

Table 5.13 shows the syntax of the Short-form Virtual Channel Table section.

Table 5.13 Short-form Virtual Channel Table section format

	Bits	Bytes	Format
shortform_virtual_channel_table_section(){			
table_ID	8	1	uimsbf value 0xC4
zero	2	2	bslbf
reserved	2		bslbf
section_length	12		uimsbf
zero	3	1	bslbf
protocol_version	5		see Sec. 4.4.1
transmission_medium	4	1	uimsbf
table_subtype	4		uimsbf see Table 5.14
VCT_ID	16	2	uimsbf
if (table_subtype==DCM) {			
DCM_structure()	*	(*)	
}			
if (table_subtype== VCM) {			
VCM_structure()	*	(*)	
}			
if (table_subtype== ICM) {			
ICM_structure()	*	(*)	
}			
for (i=0; i<N; i++) {			
descriptor()	*	(*)	optional
}			
CRC_32	32	4	rpchof
}			

table_ID — The table_ID of the Short-form Virtual Channel Table shall be 0xC4.

transmission_medium — This 4-bit field shall be set to zero (0x0).

table_subtype — A 4-bit field that indicates the map type being delivered in this S-VCT section. Three map types are currently defined, the Virtual Channel Map (VCM), the Defined Channels Map (DCM), and the Inverse Channel Map (ICM). Table 5.14 defines table_subtype.

Table 5.14 S-VCT Table Subtypes

table_subtype	meaning
0	VCM — Virtual Channel Map
1	DCM — Defined Channels Map
2	ICM — Inverse Channel Map
3-15	Reserved

An S-VCT section received with `table_subtype` indicating an unknown or unsupported map type shall be discarded.

VCT_ID — A 16-bit unsigned integer value, in the range 0x0000 to 0xFFFF, indicating the VCT to which the channel definitions in this table section apply. This 16-bit field may be used by the POD module for filtering purposes. Only one version of the S-VCT, corresponding to one value of `VCT_ID`, shall be delivered to the Host across the Extended Channel interface at a given time.

5.3.1 Defined Channels Map

Table 5.15 shows the format of the `DCM_structure()`.

Table 5.15 DCM structure format

	Bits	Bytes	Format
DCM_structure(){			
zero	4	2	bslbf
first_virtual_channel	12		uimsbf range 0-4095
zero	1	1	bslbf
DCM_data_length	7		uimsbf range 1-127
for (i=0; i<DCM_data_length; i++) {			
range_defined	1	(1)	bslbf {no, yes}
channels_count	7		uimsbf range 1-127
}			
}			

first_virtual_channel — An unsigned 12-bit integer reflecting the first virtual channel whose existence is being provided by this table section, for the map identified by the `VCT_ID` field. The range is 0 to 4095.

DCM_data_length — A 7-bit unsigned integer number, in the range 1 to 127, that defines the number of DCM data fields to follow in the table section.

The DCM data bytes taken as a whole define which virtual channels, starting at the channel number defined by `first_virtual_channel`, are defined and which are not. Each `DCM_data_field` defines two pieces of data: a flag indicating whether this block of channels is defined or not, and the number of channels in the block. The bytes are interpreted in an accumulative way, with a pointer into the Short-form Virtual Channel Table which is initialized to `first_virtual_channel`. As each byte is processed, the pointer is incremented by the number of channels indicated by the `channels_count` field.

For example, if channels 2-90, 200-210, 400-410, 600-610, 800-810, and 999 were defined, and `first_virtual_channel` was zero, the DCM data sequence (in decimal) would be the

following, where underlined numbers have the `range_defined` bit set: 2, 89, 109, 11, 127, 62, 11, 127, 62, 11, 127, 62, 11, 127, 61, 1.

range_defined — A Boolean flag that indicates, when true, that the number of channels given by `channels_count` is defined in the VCT, starting at the current pointer value. When the flag is clear, the number of channels equal to `channels_count` are currently not defined starting at the current pointer value.

channels_count — An unsigned 7-bit integer number, in the range one to 127, that indicates the number of defined (or undefined) channels in a group.

5.3.2 Virtual Channel Map

Table 5.16 shows the format of the `VCM_structure()`.

Table 5.16 VCM structure format

	Bits	Bytes	Format
<code>VCM_structure(){</code>			
zero	2	1	bslbf
descriptors_included	1		bslbf {no, yes}
zero	5		bslbf
splice	1	1	bslbf {no, yes}
zero	7		bslbf
activation_time	32	4	uimsbf
number_of_VC_records	8	1	
for (i=0; i<number_of_VC_records; i++) {			
virtual_channel()	*	(*)	
}			
}			

descriptors_included — A Boolean flag that indicates, when set, that one or more record-level descriptors are present in the table section. Record-level descriptors are those defined in Table 5.17 following the “if (`descriptors_included`)” statement. When the flag is clear, the record-level descriptor block is absent. The `descriptors_included` flag is not applicable to the section level descriptors shown at the bottom of Table 5.13.

The activation time indicates the time at which the data delivered in the table section will be valid.

splice — A Boolean flag that indicates, when set, that the Host should arm video processing hardware to execute the application of the data delivered in the `VCM_structure()` at the next MPEG-2 video splice point if the virtual channel changes described in the table section apply to a currently acquired channel, and the `activation_time` is reached. If the activation is immediate or specified as a time that has since passed, the data should be applied immediately. When the `splice` flag is clear, the virtual channel change is made directly, without arming video hardware for a splice.

activation_time — A 32-bit unsigned integer field providing the absolute second the virtual channel data carried in the table section will be valid, defined as the number of seconds since 0000 Hours UTC, January 6th, 1980. If the `GPS.UTC_offset` delivered in the System Time Table is zero,

activation_time includes the correction for leap seconds. Otherwise, activation_time can be converted to UTC by subtracting the GPS_UTC_offset. If the activation_time is in the past, the data in the table section shall be considered valid immediately. An activation_time value of zero shall be used to indicate immediate activation.

A Host may enter a virtual channel record whose activation times are in the future into a queue. Such a queue may be called a *pending virtual channel* queue. Hosts are not required to implement a pending virtual channel queue, and may choose to discard any data that is not currently applicable.

number_of_VC_records — An 8-bit unsigned integer number, in the range 1 to 255, that identifies the number of virtual_channel() records to follow in the table section. The number of records included is further limited by the allowed maximum table section length.

virtual_channel() — Table 5.17 defines the virtual_channel() record structure.

Table 5.17 Virtual channel record format

	Bits	Bytes	Format
virtual_channel(){			
zero	4	2	bslbf
virtual_channel_number	12		uimsbf range 0-4095
application_virtual_channel	1	1	bslbf {no, yes}
zero	1		bslbf
path_select	1		bslbf see Table 5.18
transport_type	1		bslbf see Table 5.19
channel_type	4		uimsbf see Table 5.20
if (application_virtual_channel) {			
application_ID	16	(2)	
} else {			
source_ID	16	(2)	
}			
if (transport_type==MPEG_2) {			
CDS_reference	8	((1))	uimsbf range 1-255
program_number	16	((2))	
MMS_reference	8	((1))	uimsbf range 1-255
} else { /* non-MPEG-2 */			
CDS_reference	8	((1))	uimsbf range 0-255
scrambled	1	((1))	bslbf {no, yes}
zero	3		bslbf
video_standard	4		uimsbf see Table 5.21
zero	16	((2))	bslbf
}			
if (descriptors_included) {			
descriptors_count	8	(1)	uimsbf
for (i=0; i<descriptors_count; i++) {			
descriptor()	*	((*))	
}			
}			
}			

virtual_channel_number — An unsigned 12-bit integer, in the range zero to 4095, reflecting the virtual channel whose definition is being provided by this virtual channel record, for the map identified by the `VCT_ID` field.

application_virtual_channel — A binary flag that, when set, indicates this virtual channel defines an access point represented by the `application_ID`. When the flag is clear, the channel is not an application access point, and this virtual channel defines an access point represented by the `source_ID`. Support for application-type virtual channels is optional. Hosts not supporting application-type virtual channels may disregard all data associated with them. Support for application-type virtual channels is beyond the scope of this standard.

path_select — A 1-bit field that associates the virtual channel with a transmission path. For the cable transmission medium, `path_select` identifies which physical cable carries the Transport Stream associated with this virtual channel. Table 5.18 defines `path_select`.

Table 5.18 Path Select

<code>path_select</code>	meaning
0	path 1
1	path 2

transport_type — A 1-bit field identifying the type of transport carried on this carrier as either being an MPEG-2 transport (value zero), or not (value one). Table 5.19 defines the coding.

Table 5.19 Transport Type

<code>transport_type</code>	meaning
0	MPEG-2 transport
1	non-MPEG-2 transport

channel_type — A 4-bit field defining the channel type. Table 5.20 defines `channel_type`.

Table 5.20 Channel Type

<code>channel_type</code>	meaning
0	normal — Indicates that the record is a regular virtual channel record. For non-MPEG-2 channels, the <code>waveform_type</code> shall be defined as “normal.”
1	hidden — Indicates that the record identifies a virtual channel that may not be accessed by the user by direct entry of the channel number (hidden). Hidden channels are skipped when the user is channel surfing, and appear as if undefined if accessed by direct channel entry. Programs constructed for use by specific applications (such as NVD theaters) utilize hidden virtual channels. If a <code>channel_properties_descriptor()</code> is present and the <code>hide_guide</code> bit is 0, the channel may be considered to be <i>inactive</i> . Inactive channels may appear in EPG displays.
2-15	reserved — Hosts are expected to treat virtual channel records of unknown <code>channel_type</code> the same as non-existent (undefined) channels.

application_ID — A 16-bit unsigned integer number, in the range 0x0001 to 0xFFFF, that identifies the application associated with the virtual channel, on a system-wide basis. One particular

program guide application, for example, may look for a program carrying data in its native transmission format by searching through the Short-form Virtual Channel Table for a match on its assigned `application_ID`. In some cases, one application may be able to process streams associated with more than one application ID. The application ID may be used to distinguish content as well as format, for the benefit of processing within the application. The value zero for `application_ID` shall not be assigned; if specified in a Virtual Channel record, the value zero indicates “unknown” or “inapplicable” for the `application_ID/source_ID` field.

Support for application-type virtual channels is optional. Hosts not supporting application-type virtual channels may disregard all data associated with them. Support for application-type virtual channels is beyond the scope of this standard.

source_ID — A 16-bit unsigned integer number, in the range 0x0000 to 0xFFFF, that identifies the programming source associated with the virtual channel, on a system-wide basis. In this context, a *source* is one specific source of video, text, data, or audio programming. For the purposes of referencing virtual channels to the program guide database, each such program source is associated with a unique value of `source_ID`. The `source_ID` itself may appear in an EPG database, where it tags entries to specific services. The value zero for `source_ID`, if used, shall indicate the channel is not associated with a source ID.

program_number — A 16-bit unsigned integer number that associates the virtual channel number being defined with services defined in the Program Association and TS Program Map Table sections. Access to elementary streams defined in each virtual channel record involves first acquiring the Transport Stream on the carrier associated with the virtual channel, then referencing the Program Association section in PID 0 to find the PID associated with the TS Program Map Table section for this `program_number`. PIDs for each elementary stream are then found by acquisition of the TS Program Map Table section.

A `program_number` with value 0x0000 (invalid as a regular program number) is reserved to indicate that the Host is expected to discard the corresponding virtual channel record from the queue of pending virtual channel changes. Records are identified in the pending queue by their `activation_time`, `VCT_ID`, and `virtual_channel_number`. If no pending virtual channel change is found in the Host’s queue, no action should be taken for this virtual channel (i.e. the record is expected to be discarded).

For inactive channels (those not currently present in the Transport Stream), `program_number` shall be set to zero. This number shall **not** be interpreted as pointing to a Program Map Table entry.

descriptors_count — An 8-bit unsigned integer value, in the range 0 to 255, that defines the number of descriptors to follow.

CDS_reference — An unsigned 8-bit integer number, in the range 0 to 255, that identifies the frequency associated with this virtual channel. Values 1 to 255 of `CDS_reference` are used as indices into the Carrier Definition Subtable to find a frequency to tune to acquire the virtual channel. The value zero is reserved to indicate that the referenced service is carried on *all* digital multiplexes in this VCM. The `CDS_reference` field shall be disregarded for inactive channels.

MMS_reference — An 8-bit unsigned integer value, in the range 0 to 255, that references an entry in the Modulation Mode Subtable (MMS). The value zero is illegal and shall not be specified.

For digital waveforms, the `MMS_reference` associates the carrier with a digital modulation mode. For Host implementations that support only one set of modulation parameters, in systems in which one modulation method is used for all carriers, storage and processing of the `MMS_reference` is unnecessary. The `MMS_reference` field shall be disregarded for inactive channels.

video_standard — A 4-bit field that indicates the video standard associated with this non-Standard virtual channel. Table 5.21 defines `video_standard`.

Table 5.21 Video Standard

video_standard	meaning
0	NTSC — The video standard is NTSC
1	PAL 625 — The video standard is 625-line PAL
2	PAL 525 — The video standard is 525-line PAL
3	SECAM — The video standard is SECAM
4	MAC — The video standard is MAC
5-15	Reserved

descriptor() — The table section may include, at its end, one or more structures of the form `tag, length, data`. The number of descriptors present is determined indirectly by processing the `section_length` field. Descriptors are defined in Section 6.

5.3.3 Inverse Channel Map

The Inverse Channel Map, once reconstructed in the Host from a sequence of Virtual Channel records that belong to the ICM, consists of a list of `source_ID/virtual_channel_number` pairs, ordered by `source_ID`. The Host may use this table to quickly find the virtual channel carrying the program given by a particular value of `source_ID` (by binary search), if such a virtual channel exists. One Inverse Channel Map can be defined per Virtual Channel Map. The ICM may be constructed from the VCM, or linear searches may be done to resolve `source_ID` references. Transmission of the ICM is therefore optional.

Virtual channels that provide access points for applications (i.e., with the `application_virtual_channel` flag set to “yes”) are not included in the ICM.

Table 5.22 describes the format of the `ICM_structure()`.

Table 5.22 ICM structure format

	Bits	Bytes	Format
ICM_structure(){			
zero	4	2	bslbf
first_map_index	12		uimsbf range 0-4095
zero	1	1	bslbf
record_count	7		uimsbf range 1-127
for (i=0; i<record_count; i++) {			
source_ID	16	(2)	uimsbf
zero	4	(2)	bslbf
virtual_channel_number	12		uimsbf range 0-4095
}			
}			

first_map_index — A 12-bit unsigned integer, in the range 0 to 4095, that represents the index into the Inverse Channel Map where data carried in this ICM_structure() should be stored.

record_count — A 7-bit unsigned integer value, in the range 1 to 127, that represents the total number of source_ID/ virtual_channel pairs defined in this table section.

source_ID — A 16-bit unsigned integer number, in the range 0x0000 to 0xFFFF, that identifies the source associated with the virtual channel, on a system-wide basis. In this context, a “source” is one specific source of video, text, data, or audio programming. For the purposes of referencing virtual channels to the program guide database, each such source is associated with a unique value of source_ID. .

virtual_channel_number — A 12-bit unsigned integer value, in the range 0 to 4095, that represents the virtual channel, in the Short-form Virtual Channel Table section (ref. Table 5.13) given by VCT_ID, associated with the given source_ID through the virtual_channel() record (ref. Table 5.17). A virtual_channel_number of zero indicates that the program given by source_ID is currently not carried in this Short-form Virtual Channel Table. Such placeholders are useful in the case where the existence of a certain program within a VCM may come and go.

5.4 System Time Table Section

The System Time Table is used to synchronize Hosts with accurate calendar time. The System Time Table shall be associated on an Extended Channel Interface with PID value 0x1FFC, the SI_base PID. Rate of transmission is typically once per minute, at second 00 of each minute.

The processing of the System Time Table in the Host is time-critical. Delays between reception and processing of the table section increase the inaccuracy of timed events. Processing delays should be kept below 200 milliseconds.

Table 5.23 shows the format of the System Time Table section.

Table 5.23 System Time Table section format

	Bits	Bytes	Format
<code>system_time_table_section(){</code>			
table_ID	8	1	uimsbf value 0xC5
zero	2	2	bslbf
reserved	2		bslbf
section_length	12		uimsbf
zero	3	1	
protocol_version	5		see Sec. 4.4.1
zero	8	1	bslbf
system_time	32	4	uimsbf
GPS.UTC_offset	8	1	uimsbf seconds
for (i=0; i<N; i++) {			
descriptor()	*	(*)	optional
}			
CRC_32	32	4	rpchof
<code>}</code>			

table_ID — The `table_ID` of the System Time Table shall be 0xC5.

system_time — A 32-bit unsigned integer quantity representing the current system time, as the number of GPS seconds since 0000 Hours UTC, January 6th, 1980. The `system_time` value may or may not include the correction factor for leap seconds, depending upon the value of `GPS.UTC_offset`, as described below.

GPS.UTC_offset — An 8-bit value that serves dual roles. When set to zero, the field indicates that the `system_time` field carries UTC time directly. When `GPS.UTC_offset` is not equal to zero, it is interpreted as an 8-bit unsigned integer that defines the current offset in whole seconds between GPS and UTC time standards. To convert GPS time to UTC, the `GPS.UTC_offset` is subtracted from GPS time. Whenever the International Bureau of Weights and Measures decides that the current offset is too far in error, an additional leap second may be added (or subtracted), and the `GPS.UTC_offset` will reflect the change.

descriptor() — The table section may include at its end one or more structures of the form tag, length, data. The number of descriptors present is determined indirectly by processing the `section_length` field. Descriptors are defined in Section 6.

5.5 Master Guide Table (MGT)

The Master Guide Table is used to indicate the location, size, and version of tables it references. The MGT shall be associated on an Extended Channel Interface with PID value 0x1FFC, the `SI_base` PID. The MGT syntax is shown in Table 5.24. Syntax and semantics are identical to [7], except that additional table types are added to refer to all tables defined in this protocol.

table_ID — The `table_ID` of the Master Guide Table section shall be 0xC7.

section_syntax_indicator — This 1-bit field shall be set to ‘1’. It denotes that the section follows the generic section syntax beyond the section length field.

private_indicator — This 1-bit field shall be set to ‘1’.

section_length — 12-bit field specifying the number of remaining bytes in this section immediately following the `section_length` field up to the end of the section. The value of the `section_length` shall be no larger than 4,093.

map_ID — This 16-bit field may be used by the POD module for filtering purposes. The Host is expected to ignore `map_ID`. Only one version of the MGT, corresponding to one value of `map_ID` shall be delivered to the Host across an Extended Channel Interface at a given time. Consequently, the Host may disregard `map_ID` and may process the MGT `version_number` field as an indication that the MGT version has changed.

Note: The `map_ID` may be considered to be an identifier for this instance of the Master Guide Table. In some applications, the POD module may receive multiple Master Guide Table sections corresponding to distinct channel maps. In this case, the POD module is responsible for accepting one MGT and discarding the others using information provided by means outside the scope of this standard.

Table 5.24 Master Guide Table section format

	Bits	Bytes	Format
master_guide_table_section () {			
table_ID	8	1	0xC7
section_syntax_indicator	1	2	'1'
private_indicator	1		'1'
reserved	2		'11'
section_length	12		uimsbf
map_ID	16	2	uimsbf
reserved	2	1	'11'
version_number	5		uimsbf
current_next_indicator	1		'1'
section_number	8	1	0x00
last_section_number	8	1	0x00
protocol_version	8	1	uimsbf
tables_defined	16	2	uimsbf
for (i=0;i<tables_defined;i++) {			
table_type	16	2	uimsbf
reserved	3	2	'111'
table_type_PID	13		uimsbf
reserved	3	1	'111'
table_type_version_number	5		uimsbf
number_bytes	32	4	uimsbf
reserved	4	2	'1111'
table_type_descriptors_length	12		uimsbf
for (k=0;k<N;k++)			
descriptor()	var		
}			
reserved	4	2	'1111'
descriptors_length	12		uimsbf
for (I = 0;I< N;I++)			
descriptor()	var		
CRC_32	32	4	rpchof
}			

version_number — This 5-bit field is the version number of MGT. The version number shall be incremented by 1 modulo 32 when any field in the `table_types` defined in the loop below or the MGT itself changes.

current_next_indicator — This 1-bit indicator is always set to '1' for the MGT section; the MGT sent is always currently applicable.

section_number — The value of this 8-bit field shall always be 0x00 (this table is only one section long).

last_section_number — The value of this 8-bit field shall always be 0x00.

protocol_version — An 8-bit unsigned integer field whose function shall be to allow, in the future, this table type to carry parameters that may be structured differently than those defined in the current protocol. At present, the only valid value for `protocol_version` is zero. Non-zero values of `protocol_version` may only be processed by Hosts designed to accommodate the later versions as they become standardized.

tables_defined — This 16-bit unsigned integer in the range 0 to 65,535 represents the number of tables in the following loop.

table_type — This 16-bit unsigned integer specifies the type of table, based on Table 5.25.

Table 5.25 MGT Table Types

table_type	Meaning
0x0000-0x0001	[Assigned by ATSC]
0x0002	Long-form Virtual Channel Table with <code>current_next_indicator=1</code>
0x0003	Long-form Virtual Channel Table with <code>current_next_indicator=0</code>
0x0004	[Assigned by ATSC]
0x0005-0x000F	[Reserved]
0x0010	Short-form Virtual Channel Table—VCM Subtype
0x0011	Short-form Virtual Channel Table—DCM Subtype
0x0012	Short-form Virtual Channel Table—ICM Subtype
0x0013-0x01F	[Reserved]
0x0020	Network Information Table—CDS Table Subtype
0x0021	Network Information Table—MMS Table Subtype
0x0021-0x02F	[Reserved]
0x0030	Network Text Table—SNS Subtype
0x0031-0x00FF	[Reserved]
0x0100-0x017F	[Assigned by ATSC]
0x0180-0x01FF	[Reserved]
0x0200-0x027F	[Assigned by ATSC]
0x028F-0x0300	[Reserved]
0x0301-0x03FF	Rating Region Table with rating_region 1-255
0x0400-0x0FFF	[User private]
0x1000-0x10FF	Aggregate Event Information Table with MGT_tag 0 to 255
0x1100-0x11FF	Aggregate Extended Text Table with MGT_tag 0 to 255
0x1200-0xFFFF	[Reserved]

For table types formatted with the MPEG short-form syntax, the `revision_detection_descriptor()` shall be used to indicate the section number and version. For example, `table_type 0x0020` indicates the Network Information Table, CDS table subtype. One MGT reference to CDS would cover all sections of the delivered CDS.

MGT table types 0x1000 through 0x10FF reference AEIT instances with `MGT_tag` values 0x00 through 0xFF, respectively. Table types 0x1100 through 0x11FF reference AETT instances with `MGT_tag` values 0x00 through 0xFF, respectively. A `table_type` value of 0x1023 in the MGT, for example, refers to the instance of the AEIT with `MGT_tag` value 0x23.

Note that the choice of value of the `MGT_tag` is independent of the timeslot number. For example, the `MGT_tag` value used to deliver AEIT-0 may be zero or any other value up to 255.

table_type_PID — This 13-bit field specifies the PID for the `table_type` described in the loop.

table_type_version_number— This 5-bit field reflects the version number of the `table_type` described in the loop. The value of this field shall be the same as the `version_number` entered in the corresponding fields of tables and table instances. The version number for the next L-VCT (`current_next_indicator = 0`) shall be one unit more (modulo 32) than the version number for the current L-VCT (`current_next_indicator = 1`).

number_bytes — This 32-bit unsigned integer field indicates the total number of bytes used for the `table_type` described in the loop. There may be more than one instance of the indicated `table_type`.

table_type_descriptors_length — Total length of the descriptors for the `table_type` described in the loop (in bytes).

descriptors_length — Total length of the MGT descriptor list that follows (in bytes).

descriptor() — The table section may include, at its end, one or more structures of the form tag, length, data. Descriptors are defined in Section 6.

CRC_32 — This is a 32-bit field that contains the CRC value to ensure a zero output from the registers in the decoder defined in Annex A of ISO/IEC 13818-1 “MPEG-2 Systems” after processing the entire Master Guide Table section.

5.5.1 Restrictions on PID Values

Certain restrictions apply to the PID values specified in the MGT. These restrictions are necessary to ensure the Host can collect EPG data using a minimum number of concurrent flows on the Extended Channel.

- All AEIT and AETT table sections with common `MGT_tag` values shall share a common PID.
- AEIT-0, AETT-0, AEIT-1 and AETT-1 instances shall share a common PID value.⁶
- AEIT-2, AETT-2, AEIT-3 and AETT-3 instances shall be associated with a second separate PID value.
- EPG data describing events farther into the future may be associated with one or more PID values; the second PID value may be used for all or some of the AEIT/AETT-4 through AEIT/AETT-N instances ($N < 256$).

5.5.2 Restrictions on Order of Occurrence of Table References

For all table references except AEIT and AETT, the order of appearance in the MGT of various table references is not specified or restricted. For AEIT and AETT references, the following restriction applies:

⁶ Please refer to Sec. 5.8 on page 45 for definition of the AEIT-*n* and AETT-*n* notation convention used in this document.

- The order of appearance of AEIT/AETT references in the MGT shall correspond to increasing time slot assignments.

Note: this rule allows a Host to know, before processing the AEIT/AETT data which table instances correspond to near-term data and which correspond to data farther into the future. This information is useful if the Host has insufficient RAM to hold all data transmitted.

5.6 Long-form Virtual Channel Table

The Long-form Virtual Channel Table is carried in MPEG-2 table sections with table ID 0xC9, and conforms to the syntax and semantics of the MPEG-2 Private Section as described in Section 2.4.4.10 and 2.4.4.11 of ISO/IEC 13818-1. The sections of the Long-form Virtual Channel Table shall be associated on an Extended Channel Interface with PID value 0x1FFC, the SI_base PID.

The bit stream syntax for the Long-form Virtual Channel Table is shown in Table 5.26.

table_id — An 8-bit unsigned integer number that indicates the type of table section being defined here. For the `longform_virtual_channel_table_section`, the `table_id` shall be 0xC9.

section_syntax_indicator — The `section_syntax_indicator` is a one-bit field which shall be set to ‘1’ for the `longform_virtual_channel_table_section()`.

private_indicator — This 1-bit field shall be set to ‘1’.

section_length — This is a twelve bit field that specifies the number of bytes of the section, starting immediately following the `section_length` field, and including the CRC. The value in this field shall not exceed 4093.

map_ID — A 16-bit identifier for this Long-form Virtual Channel Table. In some applications, the POD module may receive multiple Long-form Virtual Channel Table sections corresponding to distinct channel maps. In this case, the POD may use the `map_ID` to distinguish them, using information provided outside the scope of this standard. In every case, the Host will receive just one L-VCT across the POD to Host interface, and the `map_ID` parameter may be ignored.

version_number — This 5 bit field is the version number of the Long-form Virtual Channel Table. For the current L-VCT (`current_next_indicator` = 1), the version number shall be incremented by 1 whenever the value of the current L-VCT changes. Upon reaching the value 31, it wraps around to 0. For the next L-VCT (`current_next_indicator` = 0), the version number shall be one unit more than that of the current L-VCT (also in modulo 32 arithmetic). In any case, the value of the `version_number` shall be identical to that of the corresponding entries in the MGT.

current_next_indicator — A one-bit indicator, which when set to ‘1’ indicates that the Long-form Virtual Channel Table sent is currently applicable. When the bit is set to ‘0’, it indicates that the table sent is not yet applicable and shall be the next table to become valid.

section_number — This 8 bit field gives the number of this section. The `section_number` of the first section in the Long-form Virtual Channel Table shall be 0x00. It shall be incremented by one with each additional section in the Long-form Virtual Channel Table.

last_section_number — This 8 bit field specifies the number of the last section (that is, the section with the highest `section_number`) of the complete Long-form Virtual Channel Table.

Table 5.26 Long-form Virtual Channel Table section format

Syntax	Bits	Bytes	Format
longform_virtual_channel_table_section () {			
table_id	8	1	0xC9
section_syntax_indicator	1	2	'1'
private_indicator	1		'1'
reserved	2		'11'
section_length	12		uimsbf
map_ID	16	2	uimsbf
reserved	2	1	'11'
version_number	5		uimsbf
current_next_indicator	1		bslbf
section_number	8	1	uimsbf
last_section_number	8	1	uimsbf
protocol_version	8	1	uimsbf
num_channels_in_section	8	1	uimsbf
for(i=0; i<num_channels_in_section;i++) {			
short_name	7*16	(14)	unicode™BMP
reserved	4	(3)	'1111'
major_channel_number	10		uimsbf
minor_channel_number	10		uimsbf
modulation mode	8	(1)	uimsbf
carrier_frequency	32	(4)	uimsbf
channel_TSID	16	(2)	uimsbf
program_number	16	(2)	uimsbf
reserved	2	(2)	'11'
access_controlled	1		bslbf
hidden	1		bslbf
path_select	1		bslbf
out_of_band	1		bslbf
hide_guide	1		bslbf
reserved	3		'111'
service_type	6		uimsbf
source_id	16	(2)	uimsbf
reserved	6	(2)	'111111'
descriptors_length	10		uimsbf
for (i=0;i<N;i++) {			
descriptors()			
}			
}			
reserved	6	2	'111111'
additional_descriptors_length	10		uimsbf
for(j=0; j<N;j++) {			
additional_descriptors()		var	
}			
CRC_32	32	4	rpchof
}			

protocol_version — An 8-bit unsigned integer field whose function is to allow, in the future, this table type to carry parameters that may be structured differently than those defined in the current

protocol. At present, the only valid value for `protocol_version` is zero. Non-zero values of `protocol_version` may only be processed by Hosts designed to accommodate the later versions as they become standardized.

num_channels_in_section— This 8 bit field specifies the number of virtual channels in the L-VCT section. The number is limited by the section length.

short_name— The name of the virtual channel, represented as a sequence of one to seven 16-bit character codes coded in accordance with the Basic Multilingual Plane (BMP) of Unicode™, as specified in ISO 10646-1. If the name of the virtual channel is shorter than seven Unicode™ characters, one or more instances of the null character value 0x0000 shall be used to pad the string to its fixed 14-byte length.

major_channel_number, minor_channel_number — These two 10-bit fields represent either a two-part or a one-part virtual channel number associated with the virtual channel being defined in this iteration of the “for” loop. One-part numbers range from 0 to 16,383. Two-part numbers consist of a major and a minor number part; the range of each is 0 to 999. The one- or two-part number acts as the user’s reference number for the virtual channel. Some channels may be represented with a one-part number while others in the VCT are represented with two-part numbers.

The six MSBs of the `major_channel_number` field, when all 1, indicate that a one-part number is being specified. The value of the one-part number is given, in C syntax, by:

```
one_part_number = (major_channel_number & 0x00F) << 10 + minor_channel_number
```

When the six MSBs of the `major_channel_number` field are not all 1, and the 10-bit `major_channel_number` field is less than 1000, two fields specify a two-part channel number. The value of the two-part number is given by `major_channel_number` and `minor_channel_number`.

Table 5.27 summarizes the coding of the `major_channel_number` and `minor_channel_number` fields.

Table 5.27 Major and Minor Channel Number Field Coding

	20-bit major/minor field (10-bit major + 10-bit minor)		User Channel Number
	Major Number (10 bits)	Minor Number (10 bits)	Two-part user channel number
Two-part channel numbers (1000 major numbers, each with 1000 minor numbers)	000d	000d	0-0
	000d	001d	0-1

	000d	999d	0-999
	001d	000d	1-0

	999d	999d	999-999
[Reserved]	000d to 999d	1000d-1023d	N/A
	1000-1007d	All values	N/A
One-part channel numbers (16,383 linear space numbers)	6-bit flag (set = 111111b)	One-Part Number (14 bits)	One-part user channel number
	set	0d	0
	set	1d	1
	set
	set	16383d	16383

modulation_mode — An 8-bit unsigned integer number that indicates the modulation mode for the transmitted carrier associated with this virtual channel. Values of `modulation_mode` are defined by this standard in Table 5.28. For digital signals, the standard values for modulation mode (values below 0x80) indicate transport framing structure, channel coding, interleaving, channel modulation, forward error correction, symbol rate, and other transmission-related parameters, by means of a reference to an appropriate standard. Values of `modulation_mode` 0x80 and above are outside the scope of SCTE. These may be used to specify non-standard modulation modes in private systems. A value of 0x80 for `modulation_mode` indicates that modulation parameters are specified in a private descriptor. The `modulation_mode` field shall be disregarded for inactive channels.

carrier_frequency— A 32-bit unsigned integer that represents the carrier frequency associated with the analog or digital transmission associated with this virtual channel, in Hz. For QAM-modulated signals, the given `carrier_frequency` represents the location of the digitally modulated carrier; for VSB-modulated signals, the given `carrier_frequency` represents the location of the pilot tone; for analog signals, it represents the frequency of the picture carrier. The `carrier_frequency` field shall be disregarded for inactive channels.

Table 5.28 Modulation Modes

modulation_mode	meaning
0x00	[Reserved]
0x01	analog — The virtual channel is modulated using standard analog methods for analog television.
0x02	SCTE_mode_1 — The virtual channel has a symbol rate of 5.057 Msps, transmitted in accordance with <i>Digital Transmission Standard for Cable Television</i> , Ref. [6] (Mode 1). Typically, mode 1 will be used for 64-QAM.
0x03	SCTE_mode_2 — The virtual channel has a symbol rate of 5.361 Msps, transmitted in accordance with <i>Digital Transmission Standard for Cable Television</i> , Ref. [6] (Mode 2). Typically, mode 2 will be used for 256-QAM.
0x04	ATSC (8 VSB) — The virtual channel uses the 8-VSB modulation method conforming to the <i>ATSC Digital Television Standard</i> , Ref [4].
0x05	ATSC (16 VSB) — The virtual channel uses the 16-VSB modulation method conforming to the <i>ATSC Digital Television Standard</i> , Ref [4].
0x06-0x7F	[Reserved for future use]
0x80	Modulation parameters are defined by a private descriptor
0x81-0xFF	[User Private]

channel_TSID— A 16-bit unsigned integer field, in the range 0x0000 to 0xFFFF, that represents the MPEG-2 Transport Stream ID associated with the Transport Stream carrying the MPEG-2 program referenced by this virtual channel. For inactive channels, `channel_TSID` represents the ID of the Transport Stream that will carry the service when it becomes active. The Host may use the `channel_TSID` to verify that a TS acquired at the referenced carrier frequency is actually the desired multiplex. Analog signals may have a TSID provided that it is different from any DTV Transport Stream identifier; that is, it shall be truly unique if present.⁷ A value of 0xFFFF for `channel_TSID` shall be specified for analog channels that do not have a valid TSID.

program_number — A 16-bit unsigned integer number that associates the virtual channel being defined here with the MPEG-2 Program Association and TS Program Map tables. For virtual channels representing analog services, a value of 0xFFFF shall be specified for `program_number`. For inactive channels (those not currently present in the Transport Stream), `program_number` shall be set to zero. This number shall **not** be interpreted as pointing to a Program Map Table entry.

access_controlled — A 1-bit Boolean flag, when set, indicates that events associated with this virtual channel may be access controlled. When the flag is set to 0, event access is not restricted.

hidden — A 1-bit Boolean flag that indicates, when set, that the virtual channel is not accessed by the user by direct entry of the virtual channel number. Hidden virtual channels are skipped when the user is channel surfing, and appear as if undefined, if accessed by direct channel entry.

⁷ A method to include such a unique 16-bit “Transmission Signal ID” in the NTSC VBI is specified in CEA-608-C [26].

Typical applications for hidden channels are test signals and NVOD services. Whether a hidden channel and its event may appear in EPG displays depends on the state of the `hide_guide` bit.

path_select — A 1-bit field that associates the virtual channel with a transmission path. Two paths are available as defined in Table 5.29 below. For the cable transmission medium, `path_select` identifies which of two physical input cables carries the Transport Stream associated with this virtual channel.

Table 5.29 Path Select

path_select	Meaning
0	path 1
1	path 2

out_of_band — A Boolean flag that indicates, when set, that the virtual channel defined in this iteration of the “for” loop is carried on the cable on the Extended Channel interface carrying the tables defined in this protocol. When clear, the virtual channel is carried within a standard tuned multiplex at that frequency.

Note: A virtual channel carried on the out-of-band channel may be acquired by opening a flow between Host and POD to capture the PAT on PID 0. Processing the PAT will determine the PID associated with that service’s PMT. Then, a flow can be opened to capture and process the PMT to determine the PIDs associated with elementary stream components of the service. Finally, a flow associated with the service’s PID can be opened to capture service-related data.

hide_guide — A Boolean flag that indicates, when set to 0 for a hidden channel, that the virtual channel and its events may appear in EPG displays. This bit shall be ignored for channels which do not have the `hidden` bit set, so that non-hidden channels and their events may always be included in EPG displays regardless of the state of the `hide_guide` bit. Typical applications for hidden channels with the `hide_guide` bit set to 1 are test signals and services accessible through application-level pointers.

An *inactive channel* is defined as a channel that has program guide data available, but the channel is not currently on the air. Inactive channels are represented as hidden channels with the `hide_guide` bit set to 0. The Transport Stream shall not carry a Program Map Table representing an inactive channel.

service_type— A 6-bit enumerated type field that identifies the type of service carried in this virtual channel, based on Table 5.30.

Table 5.30 Service Types

service_type	Meaning
0x00	[Reserved]
0x01	analog_television — The virtual channel carries analog television programming
0x02	ATSC_digital_television — The virtual channel carries television programming (audio, video and data) conforming to the ATSC Digital Television Standard
0x03	ATSC_audio_only — The virtual channel conforms to the ATSC Digital Television Standard, and has one or more standard audio and data components but no video.
0x04	ATSC_data_broadcast_service — Conforming to the ATSC data broadcast standard under development by T3/S13.
0x05-0x3F	[Reserved for future ATSC use]

source_id — A 16-bit unsigned integer number that identifies the programming source associated with the virtual channel. In this context, a *source* is one specific source of video, text, data, or audio programming. Source ID value zero is reserved to indicate that the programming source is not identified. Source ID values in the range 0x0001 to 0x0FFF shall be unique within the Transport Stream that carries the VCT, while values 0x1000 to 0xFFFF shall be unique at the regional level. Values for *source_ids* 0x1000 and above shall be issued and administered by a Registration Authority designated by the ATSC.

descriptors_length — Total length (in bytes) of the descriptors for this virtual channel that follows.

additional_descriptors_length — Total length (in bytes) of the VCT descriptor list that follows.

CRC_32 — This is a 32-bit field that contains the CRC value that ensures a zero output from the registers in the decoder defined in Annex A of ISO/IEC 13818-1 “MPEG-2 Systems” after processing the entire Long-form Virtual Channel Table section.

For inactive channels, the *short_name*, *major_channel_number*, and *minor_channel_number* fields reflect the name and channel number of the inactive channel, and may be used in construction of the program guide. The *source_ID* for inactive channels is used, as it is for active channels, to link the virtual channel to the program guide data. The *service_type* field and attribute flags reflect the characteristics of the channel that will be valid when it is active.

5.7 Rating Region Table (RRT)

The Rating Region Table carries rating information for multiple geographical regions. The RRT shall be associated on an Extended Channel Interface with PID value 0x1FFC, the *SI_base* PID.

Transmission of the RRT is required whenever any Transport Stream carries a service that includes a *content_advisory_descriptor()* in one of its Program Map Tables, or if a *content_advisory_descriptor()* appears in any transmitted AEIT. An instance of the RRT for each region referenced in any *content_advisory_descriptor()* shall be transmitted.

Each RRT instance, identified by *rating_region* (the eight least significant bits of *table_id_extension*), conveys the rating system information for one specific region. The size of each RRT instance shall not be more than 1,024 bytes (including section header and trailer), and it shall be carried by only one MPEG-2 private section.

Table 5.31 describes the Rating Region Table.

table_ID — The table_ID of the Rating Region Table (RRT) shall be 0xCA.

section_syntax_indicator — This 1-bit field shall be set to ‘1’. It denotes that the section follows the generic section syntax beyond the section length field.

private_indicator — This 1-bit field shall be set to ‘1’.

Table 5.31 Rating Region Table section format

	Bits	Bytes	Format
rating_region_table_section () {			
table_ID	8	1	0xCA
section_syntax_indicator	1	2	‘1’
private_indicator	1		‘1’
reserved	2		‘11’
section_length	12		uimsbf
table_ID_extension{			
reserved	8	1	0xFF
rating_region	8	1	uimsbf
}			
reserved	2	1	‘11’
version_number	5		uimsbf
current_next_indicator	1		‘1’
section_number	8	1	uimsbf
last_section_number	8	1	uimsbf
protocol_version	8	1	uimsbf
rating_region_name_length	8	1	uimsbf
rating_region_name_text()	var		
dimensions_defined	8	1	uimsbf
for(i=0; i<dimensions_defined;i++) {			
dimension_name_length	8	1	uimsbf
dimension_name_text()	var		
reserved	3	1	‘111’
graduated_scale	1		bslbf
values_defined	4		uimsbf
for (j=0;j<values_defined;j++) {			
abbrev_rating_value_length	8	1	uimsbf
abbrev_rating_value_text()	var		
rating_value_length	8	1	uimsbf
rating_value_text()	var		
}			
}			
reserved	6	2	‘111111’
descriptors_length	10		uimsbf
for (i=0;i<N;i++) {			
descriptors()	var		
}			
CRC_32	32	4	rpchof
}			

section_length — 12-bit field specifying the number of remaining bytes in this section immediately following the `section_length` field up to the end of the section. The value of the `section_length` shall be no larger than 1,021.

rating_region — An 8-bit unsigned integer number that defines the rating region to be associated with the text in this `rating_region_table_section()`. The value of this field is the identifier of this rating region, and thus this field may be used by the other tables (e.g. MGT) for referring to a specific rating region table. Values of `rating_region` are defined in Table 5.32.

Table 5.32 Rating Regions

rating_region	Rating Region Name
0x00	Forbidden
0x01	US (50 states + possessions)
0x02-0xFF	[Reserved]

version_number — This 5-bit field is the version number of the Rating Region Table identified by combination of the fields `table_ID` and `table_ID_extension`. The version number shall be incremented by 1 modulo 32 when any field in this instance of the Rating Region Table changes. The value of this field shall be the same as that of the corresponding entry in MGT.

current_next_indicator — This 1-bit indicator is always set to ‘1’.

section_number — The value of this 8-bit field shall always be 0x00.

last_section_number — The value of this 8-bit field shall always be 0x00.

protocol_version — The value of this 8-bit field shall always be 0x00.

rating_region_name_length — An 8-bit unsigned integer number that defines the total length (in bytes) of the `rating_region_name_text()` field to follow.

rating_region_name_text() — A data structure containing a Multiple String Structure which represents the rating region name, e.g. “U.S. (50 states + possessions)”, associated with the value given by `rating_region`. The `rating_region_name_text()` shall be formatted according to the Multiple String Structure (see Section 7.2). The display string for the rating region name shall be limited to 32 characters or less.

dimensions_defined — This 8-bit field (1-255) specifies the number of dimensions defined in this `rating_region_table_section()`.

dimension_name_length — An 8-bit unsigned integer number that defines the total length in bytes of the `dimension_name_text()` field to follow.

dimension_name_text() — A data structure containing a Multiple String Structure which represents the dimension name being described in the loop. One dimension in the U.S. rating region, for example, is used to describe the MPAA list. The dimension name for such a case may be defined as “MPAA”. The `dimension_name_text()` shall be formatted according to the Multiple String Structure (see Section 7.2). The dimension name display string shall be limited to 20 characters or less.

graduated_scale — This 1-bit flag indicates whether or not the rating values in this dimension represent a graduated scale, i.e., higher rating values represent increasing levels of rated content within the dimension. Value 1 means yes, while value 0 means no.

values_defined — This 4-bit field (1-15) specifies the number of values defined for this particular dimension.

abbrev_rating_value_length — An 8-bit unsigned integer number that defines the total length (in bytes) of the `abbrev_rating_value_text()` field to follow.

abbrev_rating_value_text() — A data structure containing a Multiple String Structure which represents the abbreviated name for one particular rating value. The abbreviated name for rating value 0 shall be set to a null string, i.e., “”. The `abbrev_rating_value_text()` shall be formatted according to the Multiple String Structure (see Section 7.2). The abbreviated value display string shall be limited to 8 characters or less.

rating_value_length — An 8-bit unsigned integer number that defines the total length (in bytes) of the `rating_value_text()` field to follow.

rating_value_text() — A data structure containing a Multiple String Structure which represents the full name for one particular rating value. The full name for rating value 0 shall be set to a null string, i.e., “”. The `rating_value_text()` shall be formatted according to the Multiple String Structure (see Section 7.2). The rating value display string shall be limited to 150 characters or less.

descriptors_length — Length (in bytes) of all of the descriptors that follow this field.

CRC_32 — This is a 32-bit field that contains the CRC value that ensures a zero output from the registers in the decoder defined in Annex A of ISO/IEC 13818-1 “MPEG-2 Systems” after processing the entire Rating Region Table section.

5.8 Aggregate Event Information Tables (AEIT)

The Aggregate Event Information Table delivers event title and schedule information that may be used to support an Electronic Program Guide application. The transmission format allows instances of table sections for different time periods to be associated with common PID values. Reducing the total number of PID values in use over an Extended Channel Interface is important, because the POD module can typically support only a small number of concurrent data flows (each associated with one PID value).

Each AEIT instance describes event data for one three-hour time period. The start time for any AEIT is constrained to be one of the following eight UTC times: 00:00 (midnight), 03:00, 06:00, 09:00, 12:00 (noon), 15:00, 18:00, and 21:00.

The notation AEIT-*n* refers to the AEIT corresponding to timeslot *n*. Value 0 for *n* indicates the current timeslot, value 1 the next timeslot, etc. The same notational methods apply to AETT.

Except for AEIT-0, each AEIT instance shall include event data only for those events actually starting within the covered time period.⁸ AEIT-0 shall also include event data for all

⁸ Although AEIT is similar in structure to the EIT in ATSC A/65, its properties differ from EIT in this regard.

events starting in a prior timeslot but continuing into the current timeslot. In addition, if the VCT entry for a particular source ID includes a `time_shifted_service_descriptor()`, AEIT-0 shall describe event data for active events on any channels referenced through the `time_shifted_service_descriptor()`.

ETMs for events described in AEIT-0 shall be provided in AETT-0 on the PID associated with AEIT-0 until they are no longer referenced by AEIT-0.

Table 5.33 defines the syntax of the Aggregate Event Information Table.

table_ID — The `table_ID` of the Aggregate Event Information Table shall be 0xD6.

section_syntax_indicator — This 1-bit field shall be set to ‘1’. It denotes that the section follows the generic section syntax beyond the section length field.

private_indicator — This 1-bit field shall be set to ‘1’.

section_length — 12-bit field specifying the number of remaining bytes in this section immediately following the `section_length` field up to the end of the section, including the `CRC_32` field. The value of this field shall not exceed 4,093.

AEIT_subtype — This 8-bit field identifies the subtype of the AEIT. In the current protocol, only table subtype value 0x00 is defined. Host devices shall discard instances of the `aggregate_event_information_table_section()` in which an unknown `AEIT_subtype` is specified (currently, any value other than zero).

MGT_tag — An 8-bit field that ties this AEIT instance to the corresponding `table_type` in the MGT and to an AETT instance with the same value. The `MGT_tag` value for an AEIT instance for a given timeslot shall be one higher (modulo 256) than the instance for the preceding time period.

version_number — This 5-bit field is the version number of the AEIT instance. An instance is identified by the `MGT_tag`. The version number shall be incremented by 1 modulo 32 when any field in the AEIT instance changes. The value of this field shall be identical to that of the corresponding entry in the MGT.

current_next_indicator — This 1-bit indicator is always set to ‘1’ for AEIT sections; the AEIT sent is always currently applicable.

section_number — This 8-bit field gives the number of this section.

last_section_number — This 8-bit field specifies the number of the last section.

num_sources_in_section — This 8-bit field gives the number of iterations of the “for” loop describing program schedule data.

source_ID — This 16-bit field specifies the `source_ID` of the virtual channel carrying the events described in this section.

Table 5.33 Aggregate Event Information Table format

Syntax	Bits	Bytes	Format
<code>aggregate_event_information_table_section () {</code>			
table_ID	8	1	0xD6
section_syntax_indicator	1	2	'1'
private_indicator	1		'1'
reserved	2		'11'
section_length	12		uimsbf
AEIT_subtype	8	1	uimsbf
MGT_tag	8	1	uimsbf
reserved	2		'11'
version_number	5		uimsbf
current_next_indicator	1		'1'
section_number	8	1	uimsbf
last_section_number	8	1	uimsbf
if (AEIT_subtype == 0) {			
num_sources_in_section	8	1	uimsbf
for (j = 0; j < num_sources_in_section; j++) {			
source_ID	16	(2)	uimsbf
num_events	8	(1)	uimsbf
for (j = 0; j < num_events; j++) {			
reserved	2	((2))	'11'
event_ID	14		uimsbf
start_time	32	((4))	uimsbf
reserved	2	((3))	'11'
ETM_present	2		bslbf
duration	20		uimsbf
title_length	8	((1))	uimsbf
title_text()	var		
reserved	4	((2))	'1111'
descriptors_length	12		
for (i=0; i < N; i++) {			
descriptor()			
}			
}			
}			
}			
else			
reserved	n*8	n	
CRC_32	32	4	rpchof
}			

num_events — Indicates the number of events to follow associated with the program source identified by `source_ID`. Value 0 indicates no events are defined for this source for the time period covered by the AEIT instance.

event_ID — This 14-bit field specifies the identification number of the event described. This number serves as a part of the event `ETM_ID` (identifier for event Extended Text Message). An assigned `event_ID` shall be unique at least within the scope of the instance of the AEIT in which it appears. Accordingly, as an example, the event associated with `event_ID 0x0123` in AEIT-m shall be considered to be an event distinct from `event_ID 0x0123` in AEIT-n, when m is not equal to n.

start_time — A 32-bit unsigned integer quantity representing the start time of this event as the number of seconds since 0000 Hours UTC, January 6th, 1980. If the `GPS.UTC_offset` delivered in the System Time Table is zero, `start_time` includes the correction for leap seconds. Otherwise, `start_time` can be converted to UTC by subtracting the `GPS.UTC_offset`.

ETM_present — This 2-bit field indicates the existence of an Extended Text Message (ETM) based on Table 5.34.

Table 5.34 ETM_present

ETM_present	Meaning
0x00	No ETM
0x01	ETM present on this out-of-band Extended Channel
0x02-0x03	[Reserved for future use]

duration — Duration of this event in seconds.

title_length — This field specifies the length (in bytes) of the `title_text()`. Value 0 means that no title exists for this event.

title_text() — The event title in the format of a Multiple String Structure. `title_text()` shall be formatted according to the Multiple String Structure (see Section 7.2).

descriptors_length — Total length (in bytes) of the event descriptor list that follows.

CRC_32 — This is a 32-bit field that contains the CRC value that ensures a zero output from the registers in the decoder defined in Annex A of ISO-13818-1 “MPEG-2 Systems” after processing the entire Aggregate Event Information Table section.

5.9 Aggregate Extended Text Tables (AETT)

The Aggregate Extended Text Table contains Extended Text Messages (ETM), which are used to provide detailed descriptions of events. An ETM is a multiple string data structure. Thus, it may represent a description in several different languages (each string corresponding to one language). If necessary, the description may be truncated to fit the allocated display space.

The transmission format of the AETT and its affiliated AEIT allows instances of AEIT/AETT table sections for different time slots to be associated with common PID values.

AETT-*n* shall be associated with the same PID value as AEIT-*n* for a given value of *n*.

The Aggregate Extended Text Table is carried in an MPEG-2 private section with `table_ID` 0xD7. An instance of the AETT includes one or more ETMs. Each description is distinguished by its unique 32-bit `ETM_ID`.

Table 5.35 defines the syntax of the Aggregate Extended Text Table.

Table 5.35 Aggregate Extended Text Table format

Syntax	Bits	Bytes	Format
<code>aggregate_extended_text_table_section () {</code>			
table_ID	8	1	0xD7
section_syntax_indicator	1	2	'1'
private_indicator	1		'1'
reserved	2		'11'
section_length	12		uimsbf
AETT_subtype	8	1	uimsbf
MGT_tag	8	1	uimsbf
reserved	2	1	'11'
version_number	5		uimsbf
current_next_indicator	1		'1'
section_number	8	1	uimsbf
last_section_number	8	1	uimsbf
if (AETT_subtype == 0) {			
num_blocks_in_section	8	1	uimsbf
for (j = 0; j < num_blocks_in_section; j++) {			
ETM_ID	32	(4)	uimsbf
reserved	4	(2)	'1111'
extended_text_length	12		uimsbf
extended_text_message()	var		
}			
}			
else			
reserved	n*8	n	
CRC_32	32	4	rpchof
}			

table_ID — The `table_ID` of the Aggregate Extended Text Table shall be 0xD7.

section_syntax_indicator — This 1-bit field shall be set to '1'. It denotes that the section follows the generic section syntax beyond the section length field.

private_indicator — This 1-bit field shall be set to '1'.

section_length — 12-bit field specifying the number of remaining bytes in the section immediately following the `section_length` field up to the end of the section. The value of the `section_length` shall be no larger than 4093.

AETT_subtype — This 8-bit field identifies the subtype of the AETT. In the current protocol, only table subtype value 0x00 is defined. Host devices shall discard instances of the `aggregate_extended_text_table_section()` in which an unknown `AETT_subtype` is specified (currently, any value other than zero).

MGT_tag — An 8-bit field that ties this AETT instance to the corresponding `table_type` in the MGT and to an AEIT instance with the same value. The `MGT_tag` value for an AETT instance for a given time period shall be one higher (modulo 256) than the instance for the preceding time period.

version_number — This 5-bit field is the version number of the AETT instance. An instance is uniquely identified by its `MGT_tag`. The version number shall be incremented by 1 modulo 32

when any field in the AETT instance changes. The value of this field shall be identical to that of the corresponding entry in the MGT.

current_next_indicator — This 1-bit indicator is always set to ‘1’ for AETT sections; the AETT sent is always currently applicable.

section_number — This 8-bit field gives the number of this section.

last_section_number — This 8-bit field specifies the number of the last section.

num_blocks_in_section — This 8-bit field gives the number of iterations of the “for” loop describing ETM data.

ETM_ID — Unique 32-bit identifier of this Extended Text Message. This identifier is assigned by the rule shown in Table 5.36.

Table 5.36 ETM ID

	MSB			LSB	
Bit	31	16	15	2	1 0
event ETM_ID	source_ID		event_ID		1 0

extended_text_length — A 12-bit unsigned integer number that represents the length, in bytes, of the extended_text_message() field directly following.

extended_text_message() — The extended text message in the format of a Multiple String Structure (see Section 7.2).

CRC_32 — This is a 32-bit field that contains the CRC value that ensures a zero output from the registers in the decoder defined in Annex A of ISO-13818-1 “MPEG-2 Systems” after processing the entire Transport Stream AETT section.

6 DESCRIPTORS

This section defines descriptors applicable for use with various table sections defined in this standard.

6.1 Descriptor Usage

Table 6.1 lists all descriptors, their tag numbers and associated table sections applicable to out-of-band SI transport. Asterisks mark the tables where the descriptors may appear. The range of descriptor tags defined or reserved by MPEG-2 includes those with tag values 0x3F or below, plus 0xFF.

Table 6.1 Descriptor Usage

Descriptor Name	Tag	Table Section								
		PMT	NIT	NTT	S-VCT	STT	MGT	L-VCT	RRT	AET
stuffing descriptor	0x80	*	*	*	*	*	*	*	*	*
AC-3 audio descriptor	0x81	*								*
Caption service descriptor	0x86	*								*
Content advisory descriptor	0x87	*								*
Revision detection descriptor	0x93		*	*	*					
Two part channel no. descriptor	0x94				*					
Channel properties descriptor	0x95				*					
Daylight savings time descriptor	0x96					*				
Extended channel name descr.	0xA0							*		
Time shifted service descriptor	0xA2							*		
Component name descriptor	0xA3	*								
User private descriptors	0xC0-0xFF		*	*	*	*	*	*	*	*

6.2 Stuffing Descriptor

For certain applications it is necessary to define a block of N bytes as a placeholder. The N bytes themselves are not to be processed or interpreted. The `stuffing_descriptor()` is specified for this purpose. The `stuffing_descriptor()` is simply a descriptor type for which the contents, as indicated by the `descriptor_length` field, are to be disregarded. The tag type for the stuffing descriptor is 0x80. The `stuffing_descriptor()` may appear where descriptors are allowed in any table defined in this standard.

6.3 AC-3 Audio Descriptor

The AC-3 audio descriptor, as defined in Ref. [3] and constrained in Annex B of Ref. [4], may be used in the PMT and/or in AEITs.

6.4 Caption Service Descriptor

The caption service descriptor provides closed captioning information, such as closed captioning type and language code for events with closed captioning service. This descriptor shall not appear on events with no closed captioning service.

The bit stream syntax for the Caption Service Descriptor is shown in Table 6.2.

Table 6.2 Caption Service Descriptor format

Syntax	Bits	Bytes	Format
caption_service_descriptor() {			
descriptor_tag	8	1	0x86
descriptor_length	8	1	uimsbf
reserved	3	1	'111'
number_of_services	5		uimsbf
for (i=0;i<number_of_services;i++) {			
language	8*3	(3)	uimsbf
cc_type	1	(1)	bslbf
reserved	1		'1'
if (cc_type==line21) {			
reserved	5		'11111'
line21_field	1		bslbf
}			
else			
caption_service_number	6		uimsbf
easy_reader	1	(2)	bslbf
wide_aspect_ratio	1		bslbf
reserved	14		'1111111111111111'
}			
}			

descriptor_tag — An 8-bit field that identifies the type of descriptor. For the `caption_service_descriptor()` the value is 0x86.

descriptor_length — An 8-bit count of the number of bytes following the `descriptor_length` itself.

number_of_services — An unsigned 5-bit integer in the range 1 to 16 that indicates the number of closed caption services present in the associated video service. Note that if the video service does not carry television closed captioning, the `caption_service_descriptor()` shall not be present either in the Program Map Table or in the Aggregate Event Information Table.

Each iteration of the “for” loop defines one closed caption service present as a sub-stream within the 9600 bit per second closed captioning stream. Each iteration provides the sub-stream’s language, attributes, and (for advanced captions) the associated Service Number

reference. Refer to Ref. [24] for a description of the use of the Service Number field within the syntax of the closed caption stream.

language — A 3-byte language code per ISO 639.2/B (Ref. [13]) defining the language associated with one closed caption service. The `ISO_639_language_code` field contains a three-character code as specified by ISO 639.2/B. Each character is coded into 8 bits according to ISO 8859-1 (ISO Latin-1) and inserted in order into the 24-bit field.

cc_type — A flag that indicates, when set, that an advanced television closed caption service is present in accordance with Ref. [24]. When the flag is clear, a line-21 closed caption service is present. For line 21 closed captions, the `line21_field` indicates whether the service is carried in the even or odd field.

line21_field — A flag that indicates, when set, that the line 21 closed caption service is associated with the field 2 of the NTSC waveform. When the flag is clear, the line-21 closed caption service is associated with field 1 of the NTSC waveform. The `line21_field` flag is defined only if the `cc_type` flag indicates line-21 closed caption service.

caption_service_number — A 6-bit unsigned integer value in the range zero to 63 that identifies the Service Number within the closed captioning stream that is associated with the language and attributes defined in this iteration of the “for” loop. See Ref. [24] for a description of the use of the Service Number. The `caption_service_number` field is defined only if the `cc_type` flag indicates closed captioning in accordance with Ref. [24].

easy_reader — A Boolean flag which indicates, when set, that the closed caption service contains text tailored to the needs of beginning readers. Refer to Ref. [24] for a description of “easy reader” television closed captioning services. When the flag is clear, the closed caption service is not so tailored.

wide_aspect_ratio — A Boolean flag which indicates, when set, that the closed caption service is formatted for displays with 16:9 aspect ratio. When the flag is clear, the closed caption service is formatted for 4:3 display, but may be optionally displayed centered within a 16:9 display.

6.5 Content Advisory Descriptor

The `content_advisory_descriptor()` is used to indicate, for a given event, ratings for any or all of the rating dimensions defined in the RRT (Rating Region Table). Ratings may be given for any or all of the defined regions, up to a maximum of 8 regions per event. An event without a `content_advisory_descriptor()` indicates that the rating value for any rating dimension defined in any rating region is zero. The absence of ratings for a specific dimension is completely equivalent to having a zero-valued rating for such a dimension. The absence of ratings for a specific region implies the absence of ratings for all of the dimensions in the region. The absence of a `content_advisory_descriptor()` for a specific event implies the absence of ratings for all of the regions for the event. The bit stream syntax for the `content_advisory_descriptor()` is shown in Table 6.3.

descriptor_tag — This 8-bit unsigned integer shall have the value 0x87, identifying this descriptor as `content_advisory_descriptor`.

Table 6.3 Content Advisory Descriptor format

Syntax	Bits	Bytes	Format
<code>content_advisory_descriptor() {</code>			
descriptor_tag	8	1	0x87
descriptor_length	8	1	uimsbf
reserved	2	1	'11'
rating_region_count	6		
for (i=0; i<rating_region_count; i++) {			
rating_region	8	1	uimsbf
rated_dimensions	8	1	uimsbf
for (j=0;j<rated_dimensions;j++) {			
rating_dimension_j	8	1	uimsbf
reserved	4	1	'1111'
rating_value	4		uimsbf
}			
rating_description_length	8	1	uimsbf
rating_description_text()	var		
}			
}			

descriptor_length — This 8-bit unsigned integer specifies the length (in bytes) immediately following this field up to the end of this descriptor.

rating_region_count — A 6-bit unsigned integer value in the range 1 to 8 that indicates the number of rating region specifications to follow.

rating_region — An unsigned 8-bit integer that specifies the rating region for which the data in the bytes to follow is defined. The `rating_region` associates ratings data given here with data defined in a Ratings Region Table tagged with the corresponding rating region.

rated_dimensions — An 8-bit unsigned integer field that specifies the number of rating dimensions for which content advisories are specified for this event. The value of this field shall not be greater than the value specified by the field `dimensions_defined` in the corresponding RRT section.

rating_dimension_j — An 8-bit unsigned integer field specifies the dimension index into the RRT instance for the region specified by the field `rating_region`. These dimension indices shall be listed in numerical order, i.e., the value of `rating_dimension_j+1` shall be greater than that of `rating_dimension_j`.

rating_value — A 4-bit field represents the rating value of the dimension specified by the field `rating_dimension_j` for the region given by `rating_region`.

rating_description_length — An 8-bit unsigned integer value in the range zero to 80 that represents the length of the `rating_description_text()` field to follow.

rating_description_text() — The rating description in the format of a Multiple String Structure (see Section 7.2). The `rating_description` display string shall be limited to 16 characters or less. The rating description text shall represent the program’s rating in an abbreviated form suitable for on-screen display. The rating description text collects multidimensional text information into a single small text string. If “xxx” and “yyy” are abbreviated forms for rating values in two dimensions, then “xxx-yyy” and “xxx (yyy)” are examples of possible strings represented in `rating_description_text()`.

The program source provider shall be the responsible party for insertion of correct content_advisory_descriptors in the Program Map Table (PMT). Also, the content_advisory_descriptors may be included in Aggregate Event Information Tables. If content_advisory_descriptors are available both in AEIT and PMT, the PMT should be used first, then the AEITs.

6.6 Revision Detection Descriptor

The revision_detection_descriptor() is used to indicate whether new information is contained in the table section in which it appears.

Table 6.4 describes the revision_detection_descriptor. This descriptor should be the first descriptor in the list to limit processing overhead.

Table 6.4 Revision Detection Descriptor format

	Bits	Bytes	Format
revision_detection_descriptor(){			
descriptor_tag	8	1	uimsbf value 0x93
descriptor_length	8	1	uimsbf
reserved	3	1	bslbf
table_version_number	5		uimsbf range 0–31
section_number	8	1	uimsbf range 0 –255
last_section_number	8	1	uimsbf range 0 –255
}			

descriptor_tag—An 8-bit unsigned integer number that identifies the descriptor as a revision_detection_descriptor(). The tag shall have the value 0x93.

descriptor_length—An 8-bit unsigned integer number that indicates the number of bytes to follow in the descriptor. At present, just three bytes are defined, but the length field shall be processed to allow new data to be added to the descriptor in the future.

table_version_number—this 5-bit unsigned integer in the range 0 to 31 identifies the version of the current table. This integer applies only to the table (or the section of it) currently transmitted. Other types of tables may have different version numbers. To indicate a change in a specific table, this integer is incremented by 1 modulo 32.

section_number—An 8-bit unsigned integer in the range 0 to 255 that identifies the current table section. Version numbers for all sections of a table must be the same. Note that section_number = 0 indicates the first section of a table.

last_section_number— An 8-bit unsigned integer in the range 0 to 255 that identifies the number of sections in a table. Note that if the last_section_number = 0, then there is only one section in this table.

6.7 Two Part Channel Number Descriptor

Table 6.5 describes the two_part_channel_number_descriptor(). This descriptor may appear in the virtual_channel() record, contained in the VCM_structure; within the Short-form Virtual Channel Table section. The descriptor may be used by compatible Hosts to associate a two-part user channel

number with any virtual channel. Some channels may have a `two_part_channel_number_descriptor()` while others do not.

Note: For the L-VCT, the 10-bit major/minor number fields can be coded to represent a one-part channel number. The one-part representation is not needed for the major/minor number fields in the `two_part_channel_number_descriptor()` in the S-VCT, because there is already a 12-bit one-part number on each channel in S-VCT. It would cause confusion to allow a second one-part number to be associated with a channel defined in S-VCT.

Table 6.5 Two-part Channel Number Descriptor format

	Bits	Bytes	Format
<code>two_part_channel_number_descriptor(){</code>			
descriptor_tag	8	1	uimsbf value 0x94
descriptor_length	8	1	uimsbf
reserved	6	2	bslbf
major_channel_number	10		uimsbf range 0-999
reserved	6	2	bslbf
minor_channel_number	10		uimsbf range 0-999
<code>}</code>			

descriptor_tag—An 8-bit unsigned integer number that identifies the descriptor as a `two_part_channel_number_descriptor()`. The tag shall have the value 0x94.

descriptor_length—An 8-bit unsigned integer number that indicates the number of bytes to follow in the descriptor. At present, just four bytes are defined, but the length field shall be processed to allow new data to be added to the descriptor in the future.

major_channel_number—A 10-bit unsigned integer in the range 0 to 999 that identifies the “major” channel number to be associated with the virtual channel.

minor_channel_number—A 10-bit unsigned integer in the range 0 to 999 that identifies the “minor” channel number to be associated with the virtual channel.

Hosts that support two-part channel numbering must support this descriptor. It is only mandatory for this descriptor to be sent in the instance where system support of two-part channel numbering is required. This means for `virtual_channel()` records where the Host does not receive the two-part channel number descriptor, that the Host is expected to use the `virtual_channel_number` described in the `virtual_channel()` record in Section 5.3.2.

6.8 Channel Properties Descriptor

The `channel_properties_descriptor()` is defined to allow both forms of VCTs (S-VCT and L-VCT) carrying the same properties. Table 6.6 describes the syntax for this descriptor. The descriptor may appear within a `virtual_channel()` record in the Short-form Virtual Channel Table.

Table 6.6 Channel Properties Descriptor format

	Bits	Bytes	Format
channel_properties_descriptor(){			
descriptor_tag	8	1	uimbsf value 0x95
descriptor_length	8	1	uimbsf
channel_TSID	16	2	uimbsf
reserved	6	1	'111111'
out_of_band_channel	1		uimbsf
access_controlled	1		uimbsf
hide_guide	1	1	bslbf
reserved	1		'1'
service_type	6		uimbsf
}			

descriptor_tag—An 8-bit unsigned integer number that identifies the descriptor as a `channel_properties_descriptor()`. The tag shall have the value 0x95.

descriptor_length—An 8-bit unsigned integer number that indicates the number of bytes to follow in the descriptor. At present, just four bytes are defined, but the length field shall be processed to allow new data to be added to the descriptor in the future.

channel_TSID — A 16-bit unsigned integer field in the range 0x0000 to 0xFFFF that represents the MPEG-2 Transport Stream ID associated with the Transport Stream carrying the MPEG-2 program referenced by this virtual channel. For inactive channels, `channel_TSID` represents the ID of the Transport Stream that will carry the service when it becomes active. The Host may use the `channel_TSID` to verify that a TS acquired at the referenced carrier frequency is actually the desired multiplex. Analog signals may have a TSID that is different from any MPEG-2 Transport Stream identifier, that is, it shall be truly unique if present. A value of 0xFFFF for `channel_TSID` shall be specified for situations where a valid TSID is not known (reserved as a wildcard capability).

out_of_band — A Boolean flag that indicates, when set, that the virtual channel associated with this descriptor is carried on the cable on the Extended Channel interface carrying the tables defined in this protocol. When clear, the virtual channel is carried within a standard tuned multiplex at that frequency.

access_controlled—A Boolean flag that indicates, when set, that events associated with this virtual channel may be access controlled. When the flag is zero, event access is not restricted.

hide_guide – A Boolean flag that indicates, when set to 0 for a channel of `channel_type` hidden, that the virtual channel and its events may appear in EPG displays. This bit shall be ignored for channels which are not the `hidden` type, so that non-hidden channels and their events may always be included in EPG displays regardless of the state of the `hide_guide` bit. Typical applications for hidden channels with the `hide_guide` bit set to 1 are test signals and services accessible through application-level pointers.

service_type— A 6-bit enumerated type field that identifies the type of service carried in this virtual channel. Service type is coded according to Table 5.30.

Hosts may use this descriptor to become aware of aspects of the channel. In the case where this descriptor is not received, the Host must tune the channel and self-discover these

aspects of the channel. For example, if this descriptor is not sent, and the channel is access controlled, the Host must determine when it can obtain access permission (the same as if that bit in the descriptor were set). Similar rules can be applied for service type and `channel_TSID`.

6.9 Extended Channel Name Descriptor

The extended channel name descriptor provides the long channel name for the virtual channel containing this descriptor.

The bit stream syntax for the extended channel name descriptor is shown in Table 6.7.

Table 6.7 Extended Channel Name Descriptor format

Syntax	Bits	Bytes	Format
<code>extended_channel_name_descriptor() {</code>			
descriptor_tag	8	1	0xA0
descriptor_length	8	1	uimsbf
long_channel_name_text()	var		
<code>}</code>			

descriptor_tag — This 8-bit unsigned integer shall have the value 0xA0, identifying this descriptor as `extended_channel_name_descriptor()`.

descriptor_length — This 8-bit unsigned integer specifies the length (in bytes) immediately following this field up to the end of this descriptor.

long_channel_name_text() — The long channel name in the format of a Multiple String Structure (see Section 7.2).

6.10 Time Shifted Service Descriptor

This descriptor links one virtual channel with one or more virtual channels that carry the same programming on a time-shifted basis. The typical application is for Near Video On Demand (NVOD) services.

Note: For the L-VCT, the 10-bit major/minor number fields can be coded to represent a one-part channel number. The one-part representation is not applicable for the major/minor number fields in the `time_shifted_services_descriptor()` because this descriptor is not applicable to S-VCT (see Table A.2). The major/minor number fields in the `time_shifted_services_descriptor()` are only used to match against fields in the L-VCT.

The bit stream syntax for the `time_shifted_service_descriptor()` is shown in Table 6.8.

Table 6.8 Time Shifted Service Descriptor format

Syntax	Bits	Bytes	Format
<code>time_shifted_service_descriptor() {</code>			
descriptor_tag	8	1	0xA2
descriptor_length	8	1	uimsbf
reserved	3	1	'111'
number_of_services	5		uimsbf
for (i=0;i<number_of_services;i++) {			
reserved	6	1	'111111'
time_shift	10	1	uimsbf
reserved	4	2	'1111'
major_channel_number	10		uimsbf
minor_channel_number	10	2	uimsbf
}			
}			

descriptor_tag — This 8-bit unsigned integer shall have the value 0xA2, identifying this descriptor as `time_shifted_service_descriptor()`.

descriptor_length — This 8-bit unsigned integer specifies the length (in bytes) immediately following this field up to the end of this descriptor.

number_of_services — A 5-bit number in the range 1 to 20 that indicates the number of time-shifted services being defined here.

time_shift — A 10-bit number in the range 1 to 720 that represents the number of minutes the time-shifted service indicated by `major_channel_number` and `minor_channel_number` is time-shifted from the virtual channel associated with this descriptor.

major_channel_number — A 10-bit number in the range 1 to 999 that represents the “major” channel number associated with a time-shifted service.

minor_channel_number — A 10-bit number in the range 0 to 999 that, when non-zero, represents the “minor” or “sub-“ channel number of the virtual channel that carries a time-shifted service.

6.11 Component Name Descriptor

Table 6.9 defines the `component_name_descriptor()`, which serves to define an optional textual name tag for any component of the service.

Table 6.9 Component Name Descriptor format

Syntax	Bits	Bytes	Format
<code>component_name_descriptor() {</code>			
descriptor_tag	8	1	0xA3
descriptor_length	8	1	uimsbf
component_name_string()	var		
}			

descriptor_tag — This 8-bit unsigned integer shall have the value 0xA3, identifying this descriptor as `component_name_descriptor()`.

descriptor_length — This 8-bit unsigned integer specifies the length (in bytes) immediately following this field up to the end of this descriptor.

component_name_string() — The name string in the format of a Multiple String Structure (see Section 7.2).

6.12 Daylight Savings Time Descriptor

This descriptor is defined for optional carriage in the System Time Table section (and in no other type of table). Hosts may use the data in the descriptor if present. If not present, *no indication is being provided as to whether daylight savings time is in effect or not*. In other words, the Host shall not infer that the lack of a descriptor means that daylight savings time is not currently in effect.

A description of the use of the `daylight_savings_time_descriptor()` is provided in Annex E. The syntax is shown in the following Table.

Table 6.10 Daylight Savings Time Descriptor format

Syntax	Bits	Bytes	Format
<code>daylight_savings_time_descriptor() {</code>			
descriptor_tag	8	1	uimsbf value 0x96
descriptor_length	8	1	uimsbf
DS_status	1	1	bslbf
reserved	2		'11'
DS_day_of_month	5		uimsbf
DS_hour	8	8	uimsbf
<code>}</code>			

descriptor_tag — This 8-bit unsigned integer shall have the value 0x96, identifying this descriptor as `daylight_savings_time_descriptor`.

descriptor_length — This 8-bit unsigned integer specifies the length (in bytes) immediately following this field up to the end of this descriptor.

DS_status — This bit indicates the status of daylight savings.

DS_status = '0': Not in daylight savings time.

DS_status = '1': In daylight savings time.

DS_day_of_month — This 5-bit unsigned integer field indicates the local day of the month on which the transition into or out of daylight savings time is to occur (1-31).

DS_hour — This 8-bit unsigned integer field indicates the local hour at which the transition into or out of daylight savings time is to occur (0-18). This usually occurs at 2 a.m. in the U.S.

6.13 User Private Descriptors

Privately defined descriptors are those with `descriptor_tag` in the range 0xC0 through 0xFF. They may be placed at any location where descriptors may be included within the table sections described in this Service Information standard. Ownership of one or more user private

descriptors is indicated by the presence of an MPEG registration_descriptor() preceding the descriptor(s).

7 TEXT STRING CODING

This section describes the format of text strings in this Service Information standard. Two different formats are used in this document. Text strings in the Network Text Table uses a format called Multilingual Text String (MTS), consisting of one or more mode-length-segment blocks. The MTS format is described in Section 7.1. All other tables and descriptors use a data structure called Multiple String Structure, described in Section 7.2. The following tables summarize these rules.

Table 7.1 Text String Coding Format in Tables

Table ID Value (hex)	Table	Coding	Ref.
0xC3	Network Text Table (NTT)	MTS	Sec. 7.1
0xCA	Rating Region Table (RRT)	MSS	Sec. 7.2
0xD6	Aggregate Event Information Table (AEIT)	MSS	Sec. 7.2
0xD7	Aggregate Extended Text Table (AETT)	MSS	Sec. 7.2

Table 7.2 Text String Coding Format in Descriptors

Descriptor Tag Value (hex)	Descriptor	Coding	Ref.
0x87	Content advisory descriptor	MSS	Sec. 7.2
0xA0	Extended channel name descriptor	MSS	Sec. 7.2
0xA3	Component name descriptor	MSS	Sec. 7.2

7.1 Multilingual Text String (MTS) Format

The format of Multilingual Text Strings adheres to the following structure. Items in square brackets may be repeated one or more times:

<mode><length><segment> [<mode><length><segment>]

A `string_length` field always precedes the one or more instances of `mode`, `length`, `segment`. This field is described in each instance where multilingual text is used, and may be either 8- or 16-bits in length, as appropriate. The value of `string_length` represents the sum total of all `mode`, `length`, `segment` blocks comprising the multilingual text string to follow, and serves to indicate the end of the text string structure.

The multilingual text data structure is designed to accommodate the need to represent a text string composed of characters from a variety of alphabets, as well as ideographic characters. Whereas characters could be represented using 16- or 32-bit character codes (as does Unicode [ISO/IEC 10646-1]), that form is inefficient and wasteful of transmission bandwidth for strings composed primarily of alphabetic rather than ideographic characters. To accommodate the need to handle Chinese, Japanese, and Korean, modes are defined that allow 16-bit (double byte) character representations in standard formats.

References below to *ISO/IEC 10646-1* (Unicode) shall be to the Basic Multilingual Plane (BMP) within that standard.

mode — An 8-bit value representing the text mode to be used to interpret characters in the segment to follow. See Table 7.3 for definition. Mode bytes in the range zero through 0x3E select Unicode character code pages. Mode byte value 0x3F selects 16-bit Unicode character coding. Mode bytes in the range 0x40 through 0xFF represent selection of a format effector function such as *underline ON* or *new line*. If `mode` is in the range 0x40 to 0x9F, then the `length/segment` portion is omitted. Format effector codes in the range 0x40 through 0x9F involve no associated parametric data; hence the omission of the `length/segment` portion. Format effector codes in the range 0xA0 through 0xFF include one or more parameters specific to the particular format effector function.

length — An 8-bit unsigned integer number representing the number of bytes in the segment to follow in this block.

segment — An array of bytes representing a character string formatted according to the mode byte.

Table 7.3 Mode Byte Encoding

Mode Byte	Meaning	Language(s) or Script
0x00	Select ISO/IEC 10646-1 Page 0x00	ASCII, ISO Latin-1 (Roman)
0x01	Select ISO/IEC 10646-1 Page 0x01	European Latin (many) ⁹
0x02	Select ISO/IEC 10646-1 Page 0x02	Standard Phonetic
0x03	Select ISO/IEC 10646-1 Page 0x03	Greek
0x04	Select ISO/IEC 10646-1 Page 0x04	Russian, Slavic
0x05	Select ISO/IEC 10646-1 Page 0x05	Armenian, Hebrew
0x06	Select ISO/IEC 10646-1 Page 0x06	Arabic ¹⁰
0x07-0x08	Reserved	-
0x09	Select ISO/IEC 10646-1 Page 0x09	Devanagari ¹¹ , Bengali
0x0A	Select ISO/IEC 10646-1 Page 0x0A	Punjabi, Gujarti
0x0B	Select ISO/IEC 10646-1 Page 0x0B	Oriya, Tamil
0x0C	Select ISO/IEC 10646-1 Page 0x0C	Telugu, Kannada
0x0D	Select ISO/IEC 10646-1 Page 0x0D	Malayalam
0x0E	Select ISO/IEC 10646-1 Page 0x0E	Thai, Lao
0x0F	Select ISO/IEC 10646-1 Page 0x0F	Tibetan
0x10	Select ISO/IEC 10646-1 Page 0x10	Georgian
0x11-0x1F	Reserved	-
0x20	Select ISO/IEC 10646-1 Page 0x20	Miscellaneous ¹²
0x21	Select ISO/IEC 10646-1 Page 0x21	Misc. symbols, arrows
0x22	Select ISO/IEC 10646-1 Page 0x22	Mathematical operators
0x23	Select ISO/IEC 10646-1 Page 0x23	Misc. technical
0x24	Select ISO/IEC 10646-1 Page 0x24	OCR, enclosed alpha-num.
0x25	Select ISO/IEC 10646-1 Page 0x25	Form and chart components
0x26	Select ISO/IEC 10646-1 Page 0x26	Miscellaneous dingbats
0x27	Select ISO/IEC 10646-1 Page 0x27	Zapf dingbats
0x28-0x2F	Reserved	-
0x30	Select ISO/IEC 10646-1 Page 0x30	Hiragana, Katakana
0x31	Select ISO/IEC 10646-1 Page 0x31	Bopomopho, Hangul elem.
0x32	Select ISO/IEC 10646-1 Page 0x32	Enclosed CJK Letters, ideo.
0x33	Select ISO/IEC 10646-1 Page 0x33	Enclosed CJK Letters, ideo.
0x34-0x3E	Reserved	-
0x3F	Select 16-bit ISO/IEC 10646-1 mode	all
0x40-0x9F	Format effector (single byte)	see Table 6.2
0xA0-0xFF	Format effector (with parameter[s])	-

Table 7.4 describes the format of the `multilingual_text_string()`.

Table 7.4 Multilingual text string format

	Bits	Bytes	Format
--	------	-------	--------

⁹ When combined with page zero (ASCII and ISO Latin-1), covers Afrikaans, Breton, Basque, Catalan, Croatian, Czech, Danish, Dutch, Esperanto, Estonian, Faroese, Finnish, Flemish, Firsian, Greenlandic, Hungarian, Icelandic, Italian, Latin, Latvian, Lithuanian, Malay, Maltese, Norwegian, Polish, Portuguese, Provençal, Ghaeto-Romanic, Romanian, Romany, Slovak, Slovenian, Serbian, Spanish, Swedish, Turkish, and Welsh.

¹⁰ Also Persian, Urdu, Pashto, Sindhi, and Kurdish.

¹¹ Devanagari script is used for writing Sanskrit and Hindi, as well as other languages of northern India (such as Marathi) and of Nepal (Nepali). In addition, at least two dozen other Indian languages use Devanagari script.

¹² General punctuation, superscripts and subscripts, currency symbols, and other diacritics.

	Bits	Bytes	Format
multilingual_text_string() for (i=0; i<N; i++) { mode if (mode < 0x3F) { eightbit_string_length for (i=0; i<eightbit_string_length; I++) { eightbit_char } } else if (mode==0x3F) { sixteenbit_string_length for (i=0; i<(sixteenbit_string_length); i+=2) { sixteenbit_char } } else if (mode >= 0xA0) { format_effector_param_length for (i=0; i<(format_effector_param_length); i++) { format_effector_data } } } }	8 8 8 8 16 8 8	(1) ((1)) (((1))) ((1)) (((2))) ((1)) (((1)))	uimsbf uimsbf uimsbf uimsbf (even) uimsbf uimsbf uimsbf

7.1.1 Mode Byte Definition

The mode byte is used either to select an *ISO/IEC 10646-1* code page from the BMP (exact mapping, or in the case of page zero, an extended mapping as defined herein), or to indicate that the text segment is coded in one of a number of standard double-byte formats. Table 6.1 shows the encoding of the mode byte. Values in the zero to 0x33 range select ISO/IEC 10646-1 code pages.

Value 0x3F selects double-byte forms used with non-alphabetic script systems, where the segment consists of a sequence of 16-bit character codes according to the *ISO/IEC 10646-1* standard. Byte ordering is high-order byte first (Motorola 680xx style), also known as *big-endian*.

7.1.2 Format Effectors

Mode bytes in the 0x40 to 0xFF range are defined as format effectors. Table 7.5 defines the encoding for currently defined single-byte values. Format effectors in the range 0x40 through 0x9F are self-contained, and do not have a length or data field following them. Format effectors in the range 0xA0 through 0xFF include a multi-byte parameter field. No multi-byte format effectors are currently defined.

7.1.2.1 Line Justification

Values 0x80, 0x81, and 0x82 signify the end of a line of displayed text. Value 0x80 indicates that the text is displayed left justified within an enclosing rectangular region (defined outside the scope of the text string). Value 0x81 indicates that the text is displayed right justified. Value 0x82 indicates that the text is centered on the line. The dimensions and location on the screen of the box into which text is placed is defined outside the scope of the text string itself.

Table 7.5 Format Effector Function Codes

Mode Byte	Meaning
0x40-0x7F	Reserved
0x80	new line, left justify
0x81	new line, right justify
0x82	new line, center
0x83	italics ON
0x84	italics OFF
0x85	underline ON
0x86	underline OFF
0x87	bold ON
0x88	bold OFF
0x89-0x9F	Reserved

7.1.2.2 Italics, Underline, Bold Attributes

These format effectors toggle *italics*, underline, and **bold** display attributes. The italics, underline, and bold format effectors indicate the start or end of the associated formatting within a text string. Formatting extends through new lines. For example, to display three lines of bold text, only one instance of the *bold ON* format effector is required.

7.1.2.3 Processing of Unknown or Unsupported Format Effectors

Hosts must discard format effectors that are unknown, or known not to be supported within a specific Host model. If a parameter value carries an undefined value, that format effector is expected to be discarded.

7.1.3 Default Attributes

Upon entry to a multilingual text string, all mode toggles (bold, underline, italics) shall be assumed “OFF”.

7.1.4 Mode Zero

ISO/IEC 10646-1 page zero (U+0000 through U+00FF) includes ASCII in the lower half (U+0000 through U+007F), and Latin characters from ISO 8859-1, *Latin-1*, in U+0090 through U+00FF. This set of characters covers Danish, Dutch, Faroese, Finnish, French, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish. Many other languages can be written with this set of letters, including Hawaiian, Indonesian/Malay, and Swahili.

Table 7.6 shows encodings of page zero characters in the range 0x80 through 0x9F (these are undefined within *ISO/IEC 10646-1*).

Table 7.6 Encodings of Columns 8 and 9 of Mode Zero Latin Character Set

	8	9
0	<RESERVED>	<RESERVED>
1	<RESERVED>	<RESERVED>
2	<RESERVED>	<RESERVED>

	8	9
3	<RESERVED>	<RESERVED>
4	<RESERVED>	<RESERVED>
5	<RESERVED>	<RESERVED>
6	<RESERVED>	<RESERVED>
7	<RESERVED>	<RESERVED>
8	<RESERVED>	U+2030 — <PER MILLE>
9	<RESERVED>	<RESERVED>
A	<RESERVED>	U+266A — <MUSICAL NOTE>
B	<RESERVED>	<RESERVED>
C	<RESERVED>	U+2190 — <LEFT ARROW>
D	<RESERVED>	U+2191 — <UP ARROW>
E	<RESERVED>	U+2192 — <RIGHT ARROW>
F	<RESERVED>	U+2193 — <DOWN ARROW>

7.1.5 Supported Characters

Support for specific characters and languages depends upon the specific model of Standard-compatible Host. Not all Hosts support all defined character sets or character codes. Use of multilingual text must be predicated on the knowledge of limitations in character rendering inherent in different Host models for which text is available.

7.2 Multiple String Structure (MSS)

The Multiple String Structure is a general data structure used specifically for text strings. Text strings appear as event titles, long channel names, the ETT messages, and RRT text items. The bit stream syntax for the Multiple String Structure is shown in Table 7.7.

number_strings — This 8-bit unsigned integer field identifies the number of strings in the following data.

ISO_639_language_code — This 3-byte (24 bits) field, in conformance with ISO 639.2/B, specifies the language used for the i^{th} string.

number_segments — This 8-bit unsigned integer field identifies the number of segments in the following data. A specific mode is assigned for each segment.

Table 7.7 Multiple String Structure

Syntax	Bits	Format
multiple_string_structure () {		
number_strings	8	uimsbf
for (i= 0;i< number_strings;i++) {		
ISO_639_language_code	8*3	uimsbf
number_segments	8	uimsbf
for (j=0;j<number_segments;j++) {		
compression_type	8	uimsbf
mode	8	uimsbf
number_bytes	8	uimsbf
for (k= 0;k<number_bytes;k++)		
compressed_string_byte [k]	8	bslbf
}		
}		
}		

compression_type — This 8-bit field identifies the compression type for the j^{th} segment. Allowed values for this field are shown in Table 7.8.

Table 7.8 Compression Types

compression_type	compression method
0x00	No compression
0x01	Huffman coding using standard encode/decode tables defined in Table C.4 and C.5 in Annex C of Ref. [7].
0x02	Huffman coding using standard encode/decode tables defined in Table C.6 and C.7 in Annex C of Ref. [7].
0x03 to 0xAF	Reserved
0xB0 to 0xFF	User private

mode — An 8-bit value representing the text mode to be used to interpret characters in the segment to follow. See Table 7.9 for definition. Mode values in the range zero through 0x3E select 8-bit Unicode™ character code pages. Mode value 0x3F selects 16-bit Unicode™ character coding. Mode values 0x40 through 0xDF are reserved for future use by ATSC. Mode values 0xE0 through 0xFE are user private. Mode value 0xFF indicates the text mode is not applicable. Hosts shall ignore string bytes associated with unknown or unsupported mode values.

number_bytes — This 8-bit unsigned integer field identifies the number of bytes that follows.

compressed_string_byte[k] — The k^{th} byte of the j^{th} segment.

Table 7.9 Modes

Mode	Meaning	Language(s) or Script
0x00	Select ISO/IEC 10646-1 Page 0x00	ASCII, ISO Latin-1 (Roman) ¹³
0x01	Select ISO/IEC 10646-1 Page 0x01	European Latin (many) ¹⁴
0x02	Select ISO/IEC 10646-1 Page 0x02	Standard Phonetic
0x03	Select ISO/IEC 10646-1 Page 0x03	Greek
0x04	Select ISO/IEC 10646-1 Page 0x04	Russian, Slavic
0x05	Select ISO/IEC 10646-1 Page 0x05	Armenian, Hebrew
0x06	Select ISO/IEC 10646-1 Page 0x06	Arabic ¹⁵
0x07-0x08	Reserved	-
0x09	Select ISO/IEC 10646-1 Page 0x09	Devanagari ¹⁶ , Bengali
0x0A	Select ISO/IEC 10646-1 Page 0x0A	Punjabi, Gujarati
0x0B	Select ISO/IEC 10646-1 Page 0x0B	Oriya, Tamil
0x0C	Select ISO/IEC 10646-1 Page 0x0C	Telugu, Kannada
0x0D	Select ISO/IEC 10646-1 Page 0x0D	Malayalam
0x0E	Select ISO/IEC 10646-1 Page 0x0E	Thai, Lao
0x0F	Select ISO/IEC 10646-1 Page 0x0F	Tibetan
0x10	Select ISO/IEC 10646-1 Page 0x10	Georgian
0x11-0x1F	Reserved	-
0x20	Select ISO/IEC 10646-1 Page 0x20	Miscellaneous
0x21	Select ISO/IEC 10646-1 Page 0x21	Misc. symbols, arrows
0x22	Select ISO/IEC 10646-1 Page 0x22	Mathematical operators
0x23	Select ISO/IEC 10646-1 Page 0x23	Misc. technical
0x24	Select ISO/IEC 10646-1 Page 0x24	OCR, enclosed alpha-num.
0x25	Select ISO/IEC 10646-1 Page 0x25	Form and chart components
0x26	Select ISO/IEC 10646-1 Page 0x26	Miscellaneous dingbats
0x27	Select ISO/IEC 10646-1 Page 0x27	Zapf dingbats
0x28-0x2F	Reserved	-
0x30	Select ISO/IEC 10646-1 Page 0x30	Hiragana, Katakana
0x31	Select ISO/IEC 10646-1 Page 0x31	Bopomopho, Hangul elem.
0x32	Select ISO/IEC 10646-1 Page 0x32	Enclosed CJK Letters, ideo.
0x33	Select ISO/IEC 10646-1 Page 0x33	Enclosed CJK Letters, ideo.
0x34-0x3E	Reserved	-
0x3F	Select 16-bit ISO/IEC 10646-1 mode	all
0x40-0xDF	Reserved	
0xE0-0xFE	User private	
0xFF	Not applicable	

¹³ The languages supported by ASCII plus the Latin-1 supplement include Danish, Dutch, English, Faroese, Finnish, Flemish, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish. Many other languages can be written with this set of characters, including Hawaiian, Indonesian, and Swahili.

¹⁴ When combined with page zero (ASCII and ISO Latin-1), covers Afrikaans, Breton, Basque, Catalan, Croatian, Czech, Esperanto, Estonian, French, Frisian, Greenlandic, Hungarian, Latin, Latvian, Lithuanian, Maltese, Polish, Provençal, Rhaeto-Romanic, Romanian, Romany, Sami, Slovak, Slovenian, Sorbian, Turkish, Welsh, and many others.

¹⁵ Also Persian, Urdu, Pashto, Sindhi, and Kurdish.

¹⁶ Devanagari script is used for writing Sanskrit and Hindi, as well as other languages of northern India (such as Marathi) and of Nepal (Nepali). In addition, at least two dozen other Indian languages use Devanagari script.

ANNEX A

OPERATIONAL PROFILES FOR CABLE SERVICE INFORMATION DELIVERY

(Normative)

A.1 Operational Profiles

This document specifies Service Information tables that are required for delivery via an out-of-band channel on cable. Six profiles are described with required and optional data specified for out-of-band transport via cable. Adherence to these profile specifications is necessary for compliance with SCTE standard transport streams.

A.1.1 Profile 1 – Baseline

This Baseline Profile reflects a practice in cable where the Short-Form Virtual Channel Table, the Modulation Mode Subtable and the Carrier Definition Subtable are used for channel navigation.

A.1.2 Profile 2 – Revision Detection

Profile 2 uses the same channel navigation mechanism as Profile 1 while adding a detection mechanism that facilitates revision handling of tables. The revision detection mechanism is applicable to the Network Information Table, Network Text Table, and S-VCT that are also used in Profile 1.

A.1.3 Profile 3 – Parental Advisory

Profile 3 uses Profile 2 as the base and adds support for the Rating Region Table in order to be compliant with the FCC-mandated V-chip content advisory scheme. Since for the U.S. and its possessions, EIA-766 [25] defines the contents of version 0 RRT, use of RRT is more applicable to outside of North America. The channel navigation mechanism is the same as in Profile 1.

A.1.4 Profile 4 – Standard Electronic Program Guide Data

Profile 4 uses Profile 3 as the base and further defines a standard format for delivery of Electronic Program Guide data by using the Aggregate Event Information Table and the Aggregate Extended Text Table. The Master Guide Table shall be supported to manage the AEITs, AETTs and other applicable tables from Profile 3. The same mechanism as in Profile 1 is used for channel navigation.

A.1.5 Profile 5 – Combination

Support for channel navigation based on L-VCT and MGT is added. Backward compatibility with systems operating within profiles 1 to 4 is maintained. Using profile 5, a cable operator could have a mixture of devices requiring the S-VCT, NIT and NTT tables as well as ones requiring the long-form tables: i.e., L-VCT, MGT.

When using profile 5, both the S-VCT and the L-VCT shall be present, and each shall describe all available services.

A.1.6 Profile 6 – PSIP Only

Profile 6 is based solely on long-form tables and is an extension of the terrestrial broadcasting mechanism. Channel navigation is based on the Long-form Virtual Channel Table. The AEIT and the optional AETT streams are used to provide EPG data.

A.2 *Profile Definition Tables*

In order to conform to this SCTE Service Information standard, a cable operator shall send a collection of tables that corresponds to one or more of the defined operational profiles defined in Table A.1 and Table A.2.

Table A.1 Usage of Table Sections in Various Profiles

		Profile 1	Profile 2	Profile 3	Profile 4	Profile 5	Profile 6
Table Section	table ID	Baseline	Revision Detection	Parental Advisory	Standard EPG Data	Combination	PSIP only (a)
Network Information Table	0xC2						
Carrier Definition Subtable		M	M	M	M	M	-
Modulation Mode Subtable		M	M	M	M	M	-
Network Text Table	0xC3						
Source Name Subtable		O	O	O	M	M	-
Short-form Virtual Channel Table	0xC4						
Virtual Channel Map		M	M	M	M	M	-
Defined Channels Map		M	M	M	M	M	-
Inverse Channel Map		O	O	O	O	O	-
System Time Table	0xC5	M	M	M	M	M	M
Master Guide Table	0xC7	-	-	(b)	M	M	M
Rating Region Table	0xCA	-	-	(c)	(c)	(c)	(c)
Long-form Virtual Channel Table	0xC9	-	-	-	-	M	M
Aggregate Event Information Table	0xD6	-	-	-	M	M	M
Aggregate Extended Text Table	0xD7	-	-	-	O	O	O

Legend:

- M Mandatory (shall be present)
- O Optional (may or may not be present)
- Not applicable (shall not be present)

Notes:

- a. Exception: System Time Table (table ID 0xC5 is used here instead of table ID 0xCD defined in PSIP) and other modifications.
- b. Mandatory for outside of North America to describe any transmitted RRT. For region 0x01 (US and possessions), delivery of an RRT is optional, because this table is standardized in EIA-766 [25].
- c. Exception: delivery of the RRT corresponding to region 0x01 (US and possessions) is optional, because this table is standardized in EIA-766 [25].

Table A.2 Usage of Descriptors in Various Profiles

		Profile 1	Profile 2	Profile 3	Profile 4	Profile 5	Profile 6
Descriptor (and associated table)	tag	Baseline	Revision Detection	Parental Advisory	Standard EPG Data	Combination	PSIP only (a)
AC-3 audio (PMT, AEIT)	0x81	-	-	-	O	O	O
Caption service (PMT, AEIT)	0x86	-	-	-	O	O	O
Content advisory (PMT, AEIT)	0x87	-	-	(b)	(b)	(b)	(b)
Revision detection (NIT,NTT, S-VCT)	0x93	-	M	M	M	M	-
Two part channel number (S-VCT)	0x94	-	-	-	O	O	-
Channel properties (S-VCT)	0x95	-	-	-	O	O	-
Daylight savings time (STT)	0x96	-	-	O	M	M	M
Extended channel name (L-VCT)	0xA0	-	-	-	-	O	O
Time shifted service (L-VCT)	0xA2	-	-	-	-	O	O
Component name (PMT)	0xA3	-	-	-	O	O	O

Legend:

- M Mandatory (shall be present)
- O Optional (may or may not be present)
- Not applicable (shall not be present)

Notes:

- a. Exception: System Time Table (table ID 0xC5 is used here instead of table ID 0xCD defined in PSIP) and other modifications.
- b. The content_advisory_descriptor() shall be present in the AEIT and PMT for a given program when Content Advisory data is available for that program. It is not required for programs for which Content Advisory data is not available.

A.3 Operational Considerations for the use of profiles (Informative)

1. If devices deployed in a particular cable system require the S-VCT in Profiles 1-5 for navigation, cable operator's use of P6 will cause operational problems.
2. If devices in use require L-VCT for navigation, cable operator's use of Profiles 1-4 will cause operational problems.
3. To provide EPG data, cable-ready devices operating on a cable system conforming to Profiles 1, 2 or 3 must use alternative protocols and methods which are beyond the scope of this specification.

ANNEX B

IMPLEMENTATION RECOMMENDATIONS

(Informative)

B.1 Implications for Retail Digital Cable-Ready Devices

Given that a cable operator could choose to deliver SI tables according to any of the profiles defined in Annex A on any given hub, digital cable-ready devices offered for retail sale should be able to accept a Short-form Virtual Channel Table for basic navigation if the Long-form Virtual Channel is not provided. It should also accept the Long-form Virtual Channel Table if the Short-form table is not provided.

B.2 Channel Number Handling

Host devices are expected to support navigation based on virtual channel records associated with two-part channel numbers. If an S-VCT virtual channel record includes a `two_part_channel_number_descriptor()`, the Host is expected to use it, and to disregard the 12-bit `virtual_channel_number` field in the same `virtual_channel()` record.

If a `two_part_channel_number_descriptor()` is not present in the record-level descriptors loop of a particular S-VCT virtual channel record, the Host is expected to use the `virtual_channel_number` field in the `virtual_channel()` record, (see Table 5.17) as the channel number reference.

Both numbering schemes may co-exist in a channel map, but each individual channel must be considered labeled with either a one-part or a two-part number.

B.3 Processing of Dynamic Changes to Service Information

The Host is expected to monitor SI data on a continuous basis, and react to changes dynamically. For example, an update to an S-VCT or L-VCT may indicate that the definition of the currently acquired virtual channel has changed. The change could involve, for example, association of the channel with a different MPEG-2 `program_number` within a Transport Stream on a different carrier frequency. In response to such a change, the Host is expected to tune to and acquire the service as redefined.

For some types of changes, the Host is not expected to respond in a visible way. For example, the name of the current event may change, but the new name would be visible as the response to a regular user action to show the event name on-screen or in a program guide display.

B.4 AEITs May Include Event Information for Inaccessible Channels

In the out-of-band system, depending on the data delivery methods employed by the cable headend and POD module, there may be occasions where AEITs are broadcast for which some Hosts do not have corresponding virtual channel assignments. In these cases, the Host is

expected to discard portions of the AEITs corresponding to `source_ID` values not present in the Virtual Channel Table (short- or long-form).

For example, the AEIT may include data describing the program schedule for a service identified with `source_ID` value 0x0123. Let's say the Virtual Channel Table does not include a channel associated with `source_ID` 0x0123. When constructing a program guide display, the channel name, number and physical location associated with events tied to `source_ID` 0x0123 will not be available. Therefore, the events described in the AEIT data for this channel are inaccessible, and the AEIT records for this `source_ID` should be discarded.

B.5 Splice Flag Processing

The S-VCT includes a flag called `splice`. Hosts supporting application of virtual channel changes tied to video splice point timing are expected to execute the change after two seconds following the `activation_time`, in the absence of a video splice point prior to that time.

Support of the splice timing function is optional in Hosts. A Host not supporting the splice timing feature is expected to apply the data delivered in the `VCM_structure()` at the indicated activation time (i.e. the `splice` flag may be simply disregarded).

ANNEX C

SERVICE INFORMATION OVERVIEW AND GUIDE

(Informative)

C.1 Table Hierarchy

Figure C.1 through Figure C.5 describe the relationships between SI tables for profiles 1 through 6 in a simplified form. A mandatory table is shown in solid box. An optional table is shown in dotted box. An italicized name indicates a sub-table or a map carried within the table.

The Short-form Virtual Channel Table section (*table_ID* 0xC4) or the Long-form Virtual Channel Table (*table_ID* 0xC9) provide navigation data on the out-of-band path. If MGT is provided, it references all tables present in Service Information (except the System Time Table).

The Master Guide Table provides general information about all of the other tables including the S-VCT, L-VCT, RRT, AEIT, and AETT. It defines table sizes necessary for memory allocation during decoding; it defines version numbers to identify those tables that need to be updated; and it gives the packet identifier (PID) values associated with instances of AEITs and AETTs.

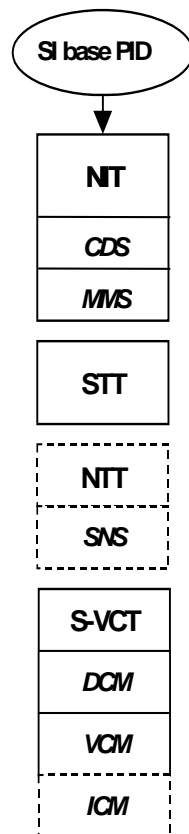


Figure C.1 Hierarchy of Table Sections -- Profiles 1 and 2

In Profile 3 and higher, the Rating Region Table must be included, with one exception, to describe rating regions in use. The exception is that delivery of version 0 of the RRT for region 0x01 (US and possessions), need not be sent because this table is standardized in EIA-766 [25]. Furthermore, for Profile 3, the MGT need not be sent if no RRT is sent.

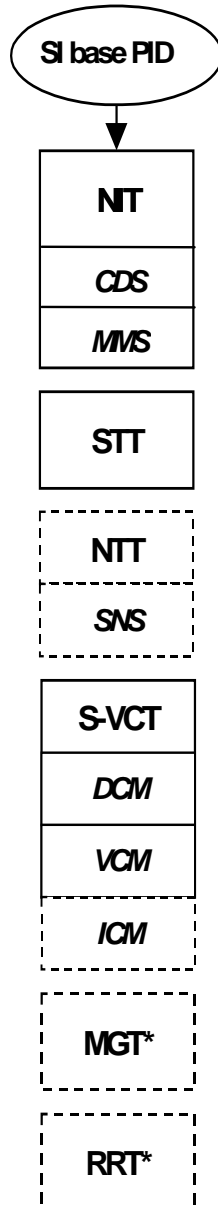


Figure C.2 Hierarchy of Table Sections -- Profile 3

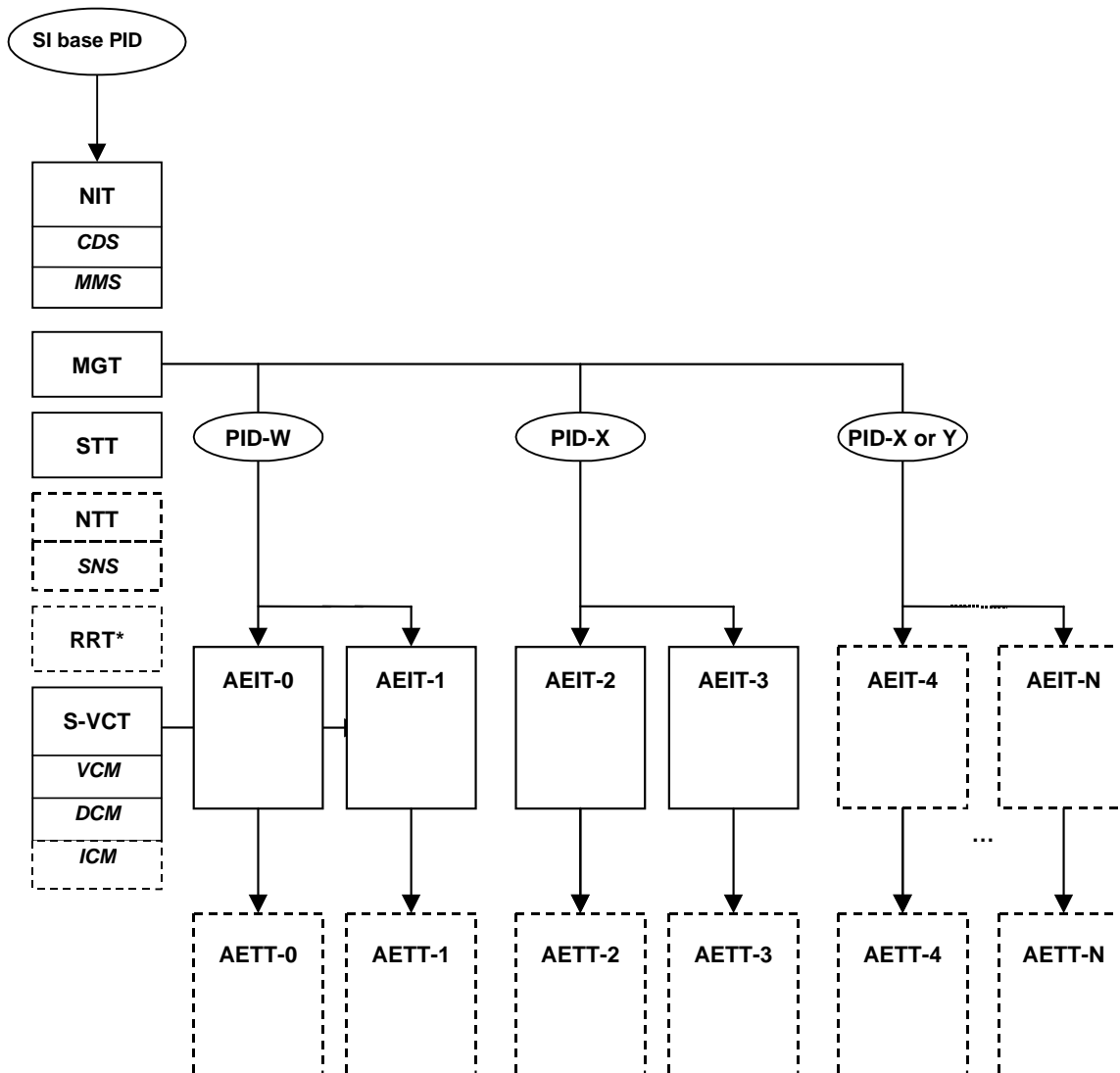


Figure C.3 Hierarchy of Table Sections -- Profile 4

Aggregate Event Information Tables are included in the out-of-band data in Profiles 4-6. Each AEIT instance describes the events or TV programs associated with a particular three-hour time slot. In the AEIT table structure, program schedule and title data for all virtual channels is aggregated together.

Each AEIT instance is valid for a time interval of three hours. As shown in Figure C.3, at minimum, AEIT-0 through AEIT-3 must be sent. Therefore, when Profiles 4-6 are used, current program information and information covering nine to twelve hours of future programming will be available to the Host.

Up to 256 AEITs may be transmitted; over 30 days of future programming may therefore be described. For the fourth timeslot and beyond (AEIT-4 through AEIT-N), the tables may be associated with the same or different PID values.

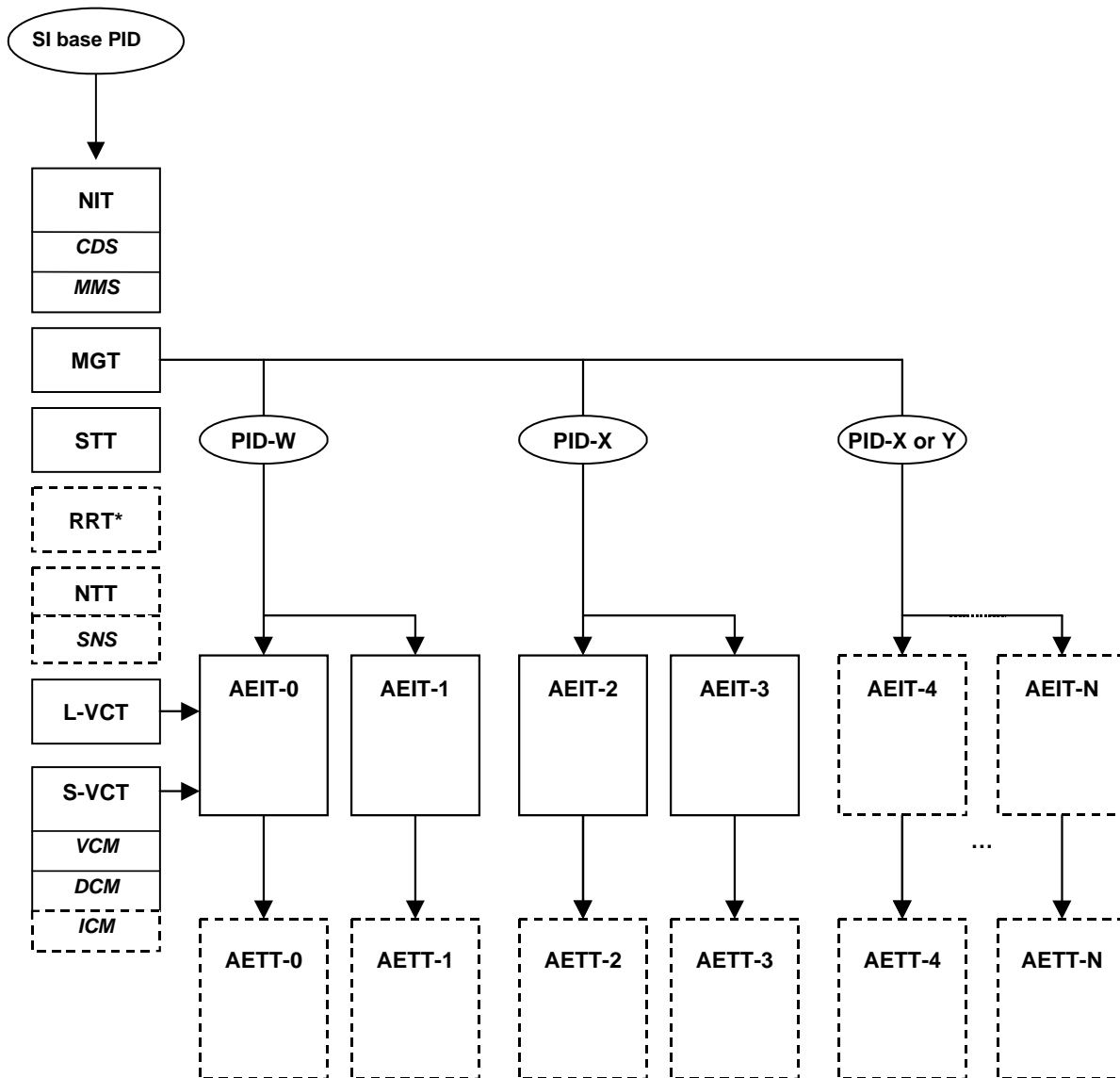


Figure C.4 Hierarchy of Table Sections -- Profile 5

The start time for any AEIT is constrained to be one of the following UTC times: 00:00 (midnight), 03:00, 06:00, 09:00, 12:00 (noon), 15:00, 18:00, and 21:00. Imposing constraints on the start times as well as the interval duration simplifies re-multiplexing. During re-multiplexing, AEIT tables coming from several distinct Transport Streams may end up grouped together or *vice versa*. If no constraints were imposed, re-multiplexing equipment would have to parse AEIT by content in real time, which is a difficult task.

However, it is also possible to regenerate one or several AEIT at any time for correcting and/or updating the content (e.g. in cases where “to be assigned” events become known). Regeneration of an AEIT may be flagged by updating version fields in the MGT. A new AEIT may also be associated with a PID value not in current use. The MGT may be updated to show this new PID value association.

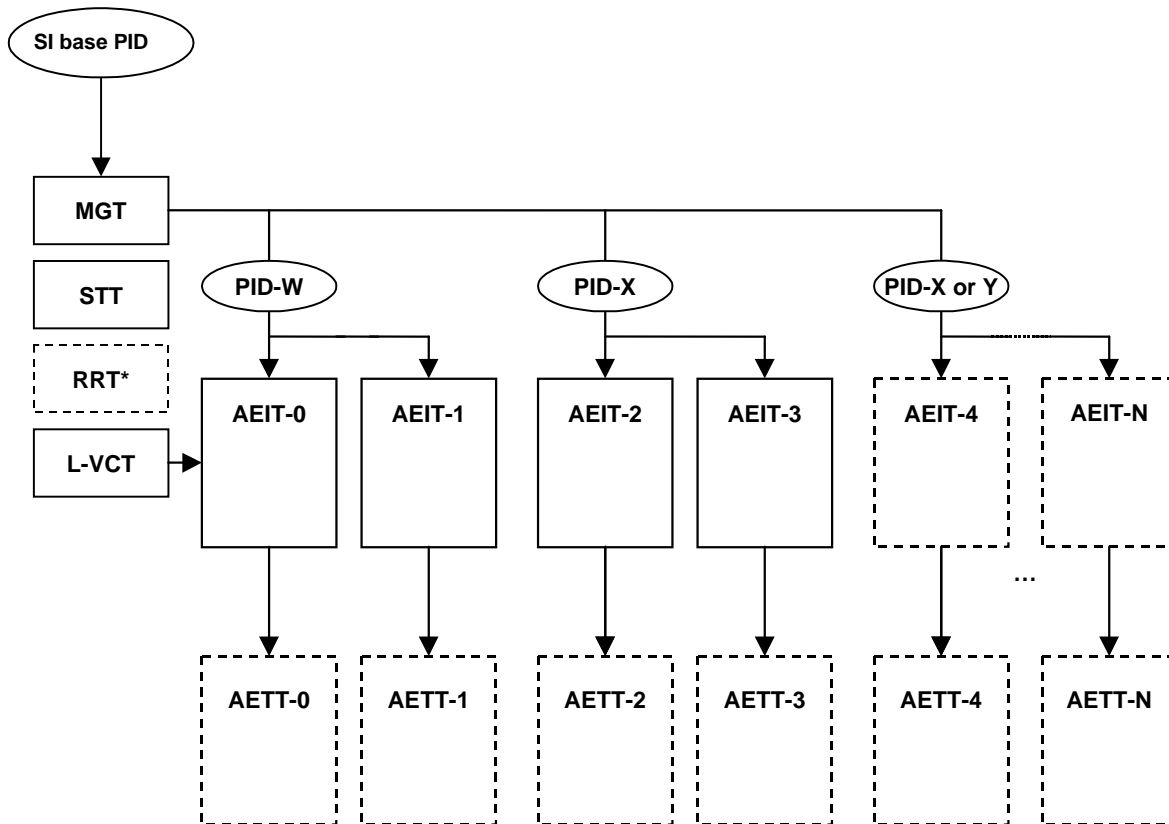


Figure C.5 Hierarchy of Table Sections -- Profile 6

In Profiles 4-6, there can be several Aggregate Extended Text Tables, each of them having its associated PID defined in the MGT. As its name indicates, the purpose of an Aggregate Extended Text Table is to carry textual data. For example, for an event such as a movie listed in the AEIT, the typical data is a short paragraph that describes the movie itself. Each Aggregate Event Information Table can have one associated AETT. Each AETT instance includes all the text associated with events starting within a particular timeslot. Aggregate Extended Text Tables are optional in Profiles 4-6.

C.2 SI_base PID

Data associated with the `SI_base` PID defines information of system-wide applicability such as frequency plans, channel maps, and channel names. The `SI_base` PID value is 0x1FFC. The types of table sections that may be included in the Network Stream include:

- ◆ Network Information Table, carrying the
 - ◆ Carrier Definition Subtable
 - ◆ Modulation Mode Subtable
- ◆ Network Text Table, carrying the Source Name Subtable
- ◆ Short-form Virtual Channel Table, carrying the
 - ◆ Virtual Channel Map

- ◆ Defined Channels Map
- ◆ Inverse Channels Map
- ◆ Long-form Virtual Channel Table
- ◆ Master Guide Table
- ◆ Rating Region Table
- ◆ System Time Table

Carrier Definition Subtable

The Carrier Definition Subtable provides a foundation for the definition of frequency plans by defining a set of carrier frequencies appropriate to a particular transmission medium. The CDS is stored in the Host as an array of as many as 255 CDS records, each consisting of:

- Carrier frequency, 15 bits, in units of 10 or 125 kHz

Modulation Mode Subtable

The Modulation Mode Subtable provides a foundation for quick acquisition of digitally modulated waveforms. A separate MMS shall be transmitted in Network data for each transmission medium supported by that network. An MMS is stored in the Host as an array of up to 255 MMS records, each consisting of:

- Modulation format: analog NTSC or QAM
- Transmission system: ITU-T (North America) or ATSC
- Symbol rate, in units of 1 Hz
- Inner coding mode, expressed as either “none” or an integer ratio such as 1/2 or 3/4
- For QAM modulation, the number of levels

Each MMS contains entries for each modulation mode currently in use by any digital waveform, plus entries for any modes anticipated to be used. As with the CDS, changes to the table are rare.

Parameters defined within the MMS are not specifically manipulated by Hosts compliant with the SI protocol, but are referenced by the Host when attempting to acquire a digitally encoded and modulated waveform.

Short-form Virtual Channel Table and Virtual Channel Record

The Short-form Virtual Channel Table is a hierarchical data structure that may carry within it the Virtual Channel Map and Virtual Channel record, for support of up to 4096 channel definition records. Each virtual channel is associated with a 16-bit reference ID number called the `source_ID`. Each record in the VCM consists of:

- The MPEG program number, associating the virtual channel record with a program defined in the Program Association Table and TS Program Map Table

- For virtual channels associated with programs carried in a program guide, the `source_ID`, a number that may be used to link the virtual channel to entries in the Electronic Program Guide (EPG) database
- For virtual channels used as access paths to application code or data (such as EPG), the *application ID*¹⁷

Source ID

Source ID is a 16-bit number associated with each program source, defined in such a way that every programming source offered anywhere in the system described in this Service Information standard is uniquely identified. For example, HBO/W has a different assigned source ID than HBO/E, and both are different from HBO-2 or HBO-3. Uniqueness is necessary to maintain correct linkages between an EPG database and virtual channel tables. See below for a discussion of the relationship between `source_ID`, virtual channels, and an EPG database.

Source Names and Source Name Subtable

The Source Name is a variable length multilingual text string associating a source ID with a textual name. The Source Name Subtable is delivered within the Network Text Table section.

Source name information is delivered in a table format separate from the table containing other information comprising the virtual channel table. Name information is not strictly necessary for channel acquisition, and (depending on the memory management scheme employed in the Host) may not always be available from memory at acquisition time. Source name information may be refreshed often, and can be available within several seconds of acquisition.

An EPG database may define textual reference names associated with given program sources (referenced by source ID). Such a database may be used to derive virtual channel names in some applications, though in an EPG database the name is generally abbreviated due to display considerations.

Name data is, unlike the regular VCT data, language tagged, so that multilingual source names may be defined. Transmission format for multilingual text is defined to include references to multiple phonetic and ideographic character sets.

Defined Channels Map and Inverse Channels Map

For a given Standard-compliant channel, DCM data consist of a series of bytes that, taken as a whole, specify which channels in the map are defined, and which are not.

Each Virtual Channel Table has associated with it a table listing `source_IDS` and their associated virtual channel numbers. The `source_ID` values are sorted by value from the lowest to

¹⁷ Source ID and application ID need never be defined in the same virtual channel record, therefore they share a common 16-bit field in the stored map. Channels are defined as for “application access” or not; if they are application access, the field defines the application ID, if not, it defines the source ID.

the highest in the table, to facilitate (using a binary search) lookup of a virtual channel given a source ID.

Master Guide Table

Use of the MGT is optional in certain profiles. Table C.1 shows a typical Master Guide Table indicating, in this case, the existence in the Transport Stream of a Long-form Virtual Channel Table, the Rating Region Table, four Aggregate Event Information Tables, and two Aggregate Extended Text Tables describing the first six hours' events.

The first entry of the MGT describes the version number and size of the Long-form Virtual Channel Table. The second entry corresponds to an instance of the Rating Region Table for region 6. If some region's policy makers decided to use more than one instance of an RRT, the MGT would list each PID, version number, and size.

The next entries in the MGT correspond to the four AEITs that must be supplied in the Transport Stream for profiles 4-6. After the AEITs, the MGT references four Aggregate Extended Text Tables. The PID values for AEIT-0 and AEIT-1 are both 0x1DD2. MGT_tag values 56 and 57 are used for these. For AEIT-2 AEIT-3, PID 0x1DD3 is used. The last four references are to Aggregate ETTs.

Note that AETT-n shares a common PID value with AEIT-n for every value of n. AEIT-0 and AETT-0 are associated with PID 0x1DD2, as are AEIT-1 and AETT-1. AEIT-2 and AETT-2 are associated with PID 0x1DD3, etc.

Descriptors can be added for each entry as well as for the entire MGT. By using descriptors, future improvements can be incorporated without modifying the basic structure of the MGT. The MGT is like a flag table that continuously informs the Host about the status of all the other tables (except the System Time which has an independent function). The MGT is continuously monitored at the Host to prepare and anticipate changes in the channel/event structure. When tables are changed at the broadcast side and the PID association is unchanged, their version numbers are incremented and the new numbers are listed in the MGT. Another method that can be used to change tables is to associate the updated tables with different PID values, and then update the MGT to reference the new PID values. Based on the MGT version or PID updates and on the memory requirements, the Host can reload the newly defined tables for proper operation.

Table C.1 Example Master Guide Table content

table_type	PID	version_number	table size (bytes)
LVCT	0x1FFC	4	5922
RRT – region 6	0x1FFC	0	1020
AEIT-0 – MGT_tag = 56	0x1DD2	6	29,250
AEIT-1 – MGT_tag = 57	0x1DD2	4	28,440
AEIT-2 – MGT_tag = 58	0x1DD3	10	25,704
AEIT-3 – MGT_tag = 59	0x1DD3	2	27,606
AETT-0 – MGT_tag = 56	0x1DD2	2	24,004
AETT-1 – MGT_tag = 57	0x1DD2	7	25,922
AETT-2 – MGT_tag = 58	0x1DD3	8	27,711
AETT-3 – MGT_tag = 59	0x1DD3	0	19,945

Table C.2 is an example MGT that may be sent after the instance in Table C.1 has expired due to the passage of time. In this example, three hours have passed, and the time slot covered in the old AEIT-0 is in the past. The AEIT with MGT_tag = 57 moves now to become AEIT-0. The AEIT with MGT_tag = 58, the new AEIT-1, moves to PID 0x1DD2. A new AEIT is added to the mix, the AEIT with MGT_tag = 60.

Table C.2 Example Revised Master Guide Table content

table_type	PID	version_number	table size (bytes)
LVCT	0x1FFC	4	5922
RRT – region 6	0x1FFC	0	1020
AEIT-0 – MGT_tag = 57	0x1DD2	4	28,440
AEIT-1 – MGT_tag = 58	0x1DD2	10	25,704
AEIT-2 – MGT_tag = 59	0x1DD3	2	27,606
AEIT-3 – MGT_tag = 60	0x1DD3	0	30,055
AETT-0 – MGT_tag = 57	0x1DD2	7	25,922
AETT-1 – MGT_tag = 58	0x1DD2	8	27,711
AETT-2 – MGT_tag = 59	0x1DD3	0	19,945
AETT-3 – MGT_tag = 60	0x1DD3	0	22,522

L-VCT

The L-VCT combines all the data pertinent to the description of a virtual channel into a single table. Use of the L-VCT instead of the S-VCT eliminates the need to send CDS, MMS,

SNS, DCM, or ICM. The L-VCT follows the standard MPEG-2 long-form section syntax (section_syntax_indicator = 1).

Rating Region Table

The Rating Region Table is a fixed data structure in the sense that its content remains mostly unchanged. It defines the rating standard that is applicable for each region and/or country. The concept of table instance introduced in the previous Section is also used for the RRT. Several instances of the RRT can be constructed and carried in the Transport Stream simultaneously. Each instance is identified by a different table_id_extension value (which becomes the rating_region in the RRT syntax) and corresponds to one and only one particular region. Each instance has a different version number which is also carried in the MGT. This feature allows updating each instance separately.

Figure C.6 shows an example of one instance of an RRT, defined for rating region 99 and carrying an example rating system. Each event listed in any of the EITs may carry a content advisory descriptor. This descriptor is an index or pointer to one or more instances of the RRT.

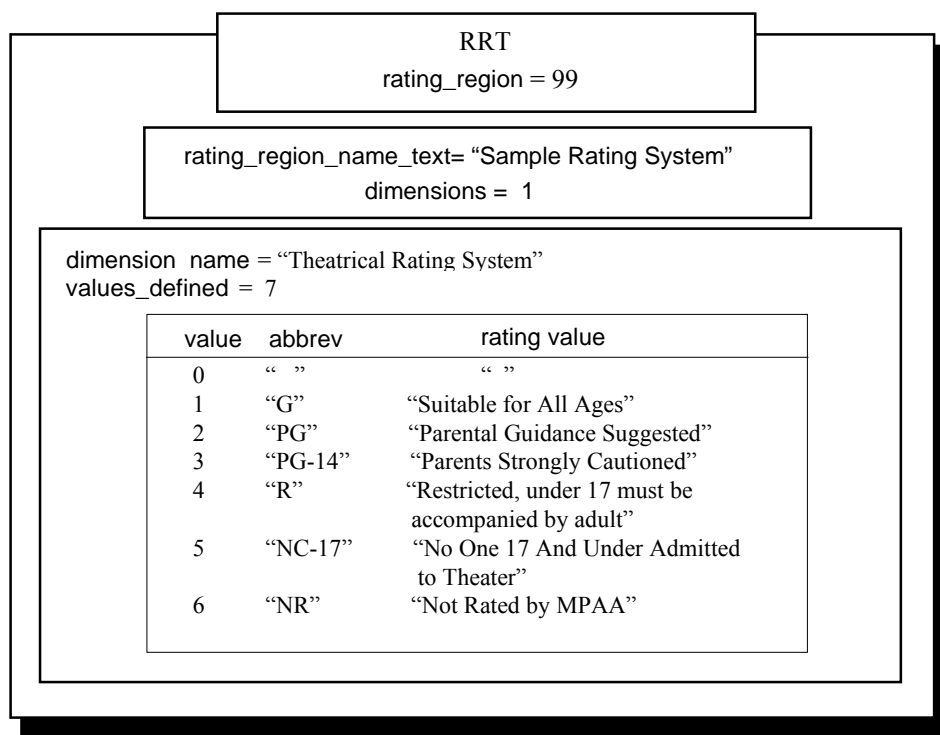


Figure C.6 An instance of a Rating Region Table

Aggregate Event Information Tables and Aggregate Extended Text Tables

The purpose of an AEIT is to list all events for those channels that appear in the VCT for a given time window. As mentioned before, AEIT-0 describes the events for the first 3 hours and AEIT-1 for the second 3 hours. AEIT-0 and AEIT-1 share a common associated PID value as defined in the MGT. In MPEG, tables can have a multitude of instances. When different

instances of a table share the same `table_id` value and PID, they are distinguished by differences in the 16-bit `table_id_extension` field.

In this SI standard for out-of-band use, each instance of AEIT-k contains a list of events for a each virtual channel. Linkage to each channel in the VCT is made via the `source_id`. For the AEIT, the `table_id_extension` field appears as `MGT_tag`.

Figure C.7 shows, for example, a program provider's instance for AEIT-0.

AEIT-0 is unique in that it must list all events starting within the three-hour time period it covers, as well as any events that started earlier but extend into the covered period. For all other AEITs, only those events actually starting within the three hour time period are included. The Host is expected to collect AEITs in order of their time coverage. If AEIT-4 is available to the Host but AEIT-3 is not, for example, information for events that started in the time period covered by AEIT-3 but extending into AEIT-4 will not be available for display.

Figure C.7 shows an example of a small AEIT-0, including event data for two sources, a channel called "TSPN" (`source_ID` 22) and one called "MOOV" (`source_ID` 80). For the three-hour period covered by AEIT-0, 9am to noon, three events are listed for TSPN and two for MOOV. The field `event_id` is a number used to identify each event. The `event_id` is used to link events with associated text delivered in the AETT. The assignment of an `event_ID` value must be unique within a source ID and a 3-hour interval defined by one AEIT instance. The `event_id` is followed by the `start_time` and then the `length_in_seconds`. Notice that for AEIT-0 only, events can have start times before the activation time of the table. ETMs are simply long textual descriptions. The collection of ETMs constitutes an Aggregate Extended Text Table (ETT).

An example of an ETM for the Car Racing event may be:

"Live coverage from Indianapolis. This car race has become the largest single-day sporting event in the world. Two hundred laps of full action and speed."

Several descriptors can be associated with each event. The most important is the content advisory descriptor which assigns a rating value according to one or more systems. Recall that the actual rating system definitions are tabulated within the RRT.

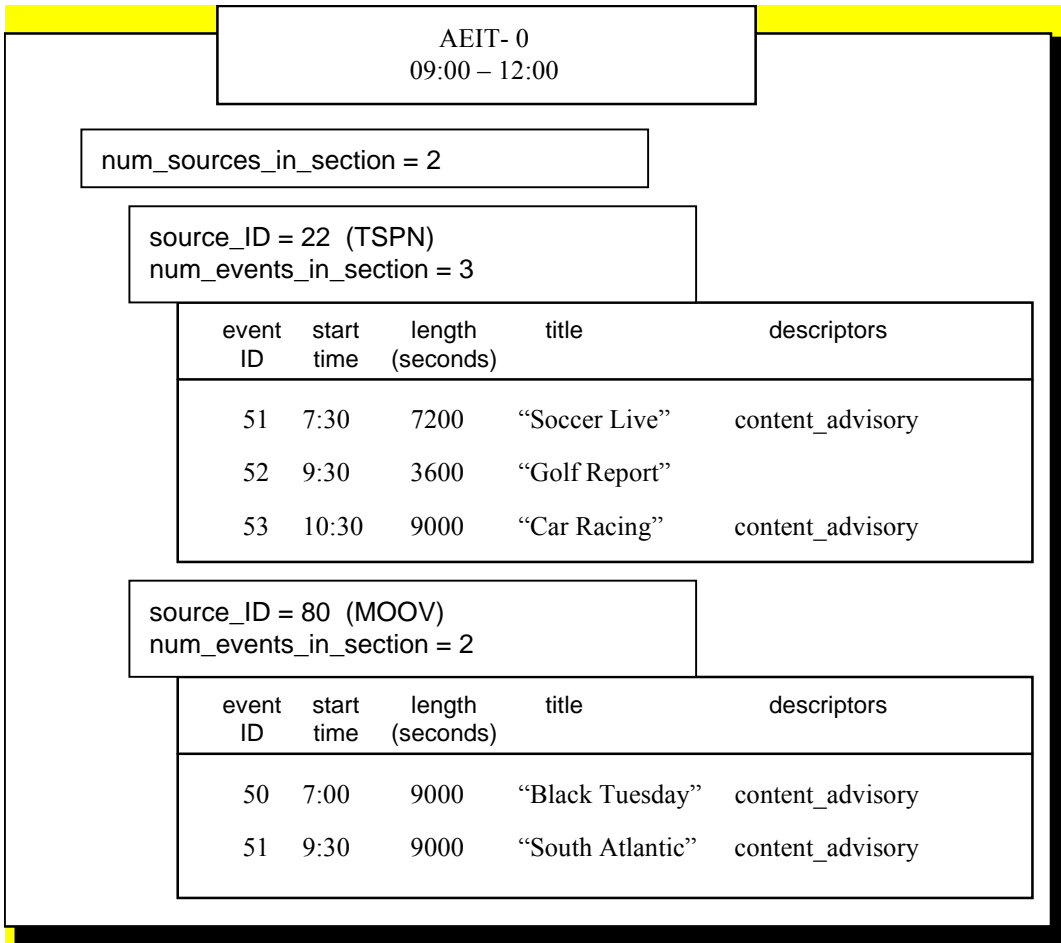
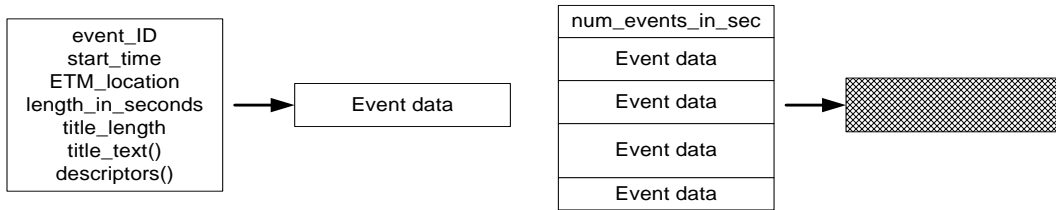


Figure C.7 Example AEIT-0

Figure C.8 diagrams the AEIT data structure. As shown, the AEIT includes event data for all sources listed in the VCT. In the figure, the hatched box represents one or more "event data" blocks, each comprised of the data items shown in the upper left.



AEIT structure:

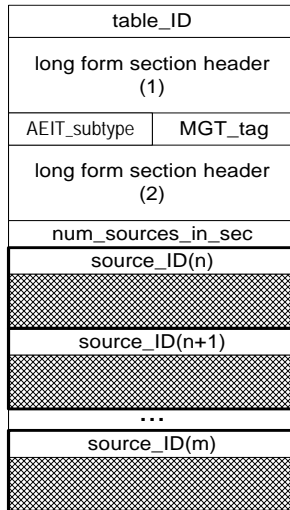


Figure C.8 AEIT data structure

Figure C.9 diagrams the AETT data structure. The AETT aggregates text for a given timeslot into one sectioned MPEG table.

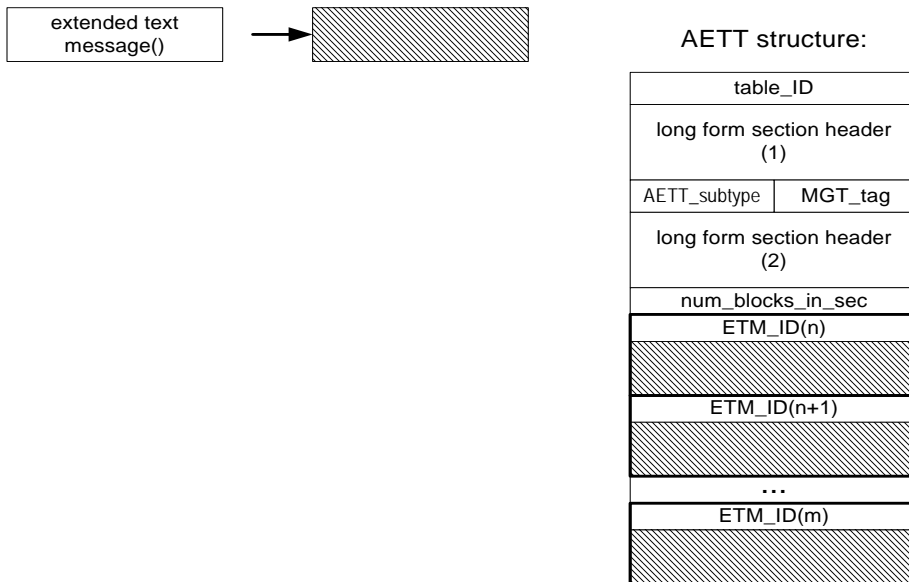


Figure C.9 Structure of AETT

An AETT-*n* instance for a given value of *n* (timeslot) is associated with the same PID value as AEIT-*n*. This means that they can be collected using a single Extended Channel data flow between Host and POD.

Inactive Channels

Any channels in the L-VCT which are not currently active shall have the hidden attribute set to 1 and the hide_guide attribute set to 0. Inactive channels in the S-VCT shall have the hidden attribute in channel_type, and the hide_guide flag in the channel_properties_descriptor() set to 0.

The following table shows expected DTV behavior for the various combinations of the hidden and hide_guide attributes. In the table the “x” entry indicates “don’t care.” A check in the “surf” column indicates the channel is available by channel surfing and via direct channel number entry. A check in the “guide” column indicates that the channel may appear in the program guide listing.

Table C.3 Receiver Behavior with hidden and hide_guide attributes

hidden	hide_guide	Receiver Behavior		
		Surf	Guide	
0	x	✓	✓	Normal channel
1	1			Special access only
1	0		✓	Inactive channel

C.3 Representation of Time

The System Time Table provides time of day information to Hosts. In this Service Information standard, time of day is represented as the number of seconds that have elapsed since the beginning of “GPS time,” 0000 Hours UTC, January 6th, 1980. GPS time is referenced to the Master Clock at the US Naval Observatory and steered to Coordinated Universal Time (UTC). UTC is the current time of day at the time zone local to Greenwich, England, and is the time source we use to set our clocks.

The cycle of the seasons, technically known as the tropical year, is approximately 365.2422 days. Using the Gregorian calendar we adjust for the fractional day by occasionally adding an extra day to the year. Every fourth year is a leap year, except that three leap years in every 400 are skipped (the centennial years not divisible by 400). With this scheme there are 97 leap years in each 400 year span, yielding an average year that is 365.2425 days long.

UTC is occasionally adjusted by one second increments to ensure that the difference between a uniform time scale defined by atomic clocks does not differ from the Earth’s rotational time by more than 0.9 seconds. The timing of occurrence of these “leap seconds” is determined by careful observations of the Earth’s rotation; each is announced months in advance. On the days it is scheduled to occur, the leap second is inserted just following 12:59:59 PM UTC.

UTC can be directly computed from the count of GPS seconds since January 6th, 1980 by subtracting from it the count of leap seconds that have occurred since the beginning of GPS time. In the months just following January 1, 1999, this offset was 13 seconds.

This protocol defines various time-related events and activities, including starting times for programs, text display, changes to VCTs, and others. Two methods of time distribution are used in headend systems. One method derives time in the form of GPS seconds from GPS Hosts. These Hosts also provide current GPS/UTC offset data. The second method of time distribution relies on the Internet Standard Network Time Protocol (NTP). NTP servers provide output in the form of UTC time, and do not provide GPS/UTC offset data. The Standard-compliant Host is synchronized to system time by the System Time Table, which provides time either in the form of GPS seconds since week zero of GPS time, January 6th, 1980, or directly in UTC time. The interpretation depends on the value of the GPS/UTC offset field. The special value of zero is used to indicate that the system is being driven by a UTC time source directly, and that GPS/UTC offset data is not available.

System Time

GPS satellites typically output GPS time in a format consisting of a week count (T_w) and a seconds within the week count (T_s), where week zero is defined as starting January 6th, 1980. For purposes of building the System Time Table, the following formula may be used:

$$T = (T_w * 604,800) + T_s$$

There are 604,800 seconds per week.

When converting between GPS seconds and current local time in hours/minutes/seconds, the following factors must be taken into account:

- **GPS to UTC offset** — Given a time represented as GPS seconds, the Host first subtracts the GPS/UTC offset to convert to UTC.
- **1980** — The first year of GPS time started on January 6th, yielding 361 days in the first year (1980 was also a leap year).
- **Leap years** — The number of leap years that occurred between the current GPS second and 1980 must be accounted for. A leap year is a year whose number is evenly divisible by four, or, in the case of century years, by 400.

Note: According to this rule, the year 2000 *is* a leap year even though it is a century year, because it is also divisible by 400.

- **Time zones** — Time zones are signed integer values in the range -12 to +13 hours, where positive numbers represent zones east of the Greenwich meridian and negative numbers west of it. Pacific Standard Time (PST) is 8 hours behind standard time, and Eastern Standard Time (EST) is 5 hours behind. The system defined by this Service Information standard accommodates time zones that are not an integral number of hours offset from Greenwich by defining time zone as an 11-bit signed integer number in units of minutes. To convert to local time, the time zone is added to Greenwich time using signed integer arithmetic.

- **Daylight savings time** — If applicable, daylight savings time must be taken into account. On a unit by unit basis, each Host may be given a definition for when daylight savings time is entered into in Spring, and when it is exited in Fall. Entry/exit points are given as absolute times (GPS seconds), and hence are given in one second resolution.

Transmission Format for Event Times

In this messaging protocol, the absolute time of action is specified for most events in terms of an unsigned 32-bit integer number, the count of GPS seconds since January 6th, 1980. This count does not wrap until after the year 2116¹⁸.

Handling of Leap Second Events

In this Service Information protocol, times of future events (such as event start times in the EIT) are specified the same as time of day, as the count of seconds since January 6th, 1980. Converting an event start time to UTC and local time involves the same calculation as the conversion of system time to local time. In both cases, the leap seconds count is subtracted from the count of GPS seconds to derive UTC.

GPS time is used to represent future times because it allows the Host to compute the time interval to the future event without regard for the possible leap second that may occur in the meantime. Also, if UTC were to be used instead, it wouldn't be possible to specify an event time that occurred right at the point in time where a leap second was added. UTC is discontinuous at those points.

Around the time a leap second event occurs, program start times represented in local time (UTC adjusted by local time zone and [as needed] daylight savings time) may appear to be off by plus or minus one second. Generating equipment may use one of two methods to handle leap seconds.

In method A, generating equipment does not anticipate the future occurrence of a leap second. In this case, prior to the leap second, program start times will appear correct. An event starting at exactly 10 AM will be computed as starting at 10:00:00. But just following the leap second, that same event time will be computed as 9:59:59. The generating equipment should re-compute the start times in all the EITs and introduce the leap second correction. Once that happens, and Hosts have updated their EIT data, the computed time will again show as 10:00:00. In this way the disruption can be limited to a matter of seconds.

In method B, generating equipment does anticipate the occurrence of a leap second, and adjusts program start times for events happening after the new leap second is added. If the leap second event is to occur at midnight tonight, an event starting at 10 AM tomorrow will be computed by receiving equipment as starting at 10:00:01.

¹⁸ Prior to that time, all initial Receivers will surely be out of service, and new ones can be designed to handle the wrap condition.

For certain types of events, the precision of method B is necessary. By specifying events using a time system that involves no discontinuities, difficulties involving leap seconds are avoided. Events such as program start times do not require that level of precision. Therefore, method A works well.

Handling of Leap Second Events

Consider the following example. Times are given relative to UTC, and would be corrected to local time zone and daylight savings time as necessary.

- ♦ Time of day (UTC): 1:00 PM, December 30th, 1998
- ♦ Event start time (UTC): 2:00 PM, January 2nd, 1999
- ♦ A leap second event will occur just after 12:59:59 PM on December 31st, 1998
- ♦ Leap seconds count on December 30th is 12

The data in the System Time Table is:

- ♦ GPS seconds = 599,058,012 = 0x23B4E65C
- ♦ GPS to UTC offset = 12

Using method A (upcoming leap second event is not accounted for):

- ♦ Event start time in EIT: 599,320,812 = 0x23B8E8EC
- ♦ Converted to UTC: 2:00:00 PM, January 2nd, 1999
- ♦ Number of seconds to event: 262,800 = 73 hours, 0 minutes, 0 seconds

Using method B (upcoming leap second event is anticipated):

- ♦ Event start time in EIT: 599,320,813 = 0x23B8E8ED
- ♦ Converted to UTC: 2:00:01 PM, January 2nd, 1999
- ♦ Number of seconds to event: 262,801 = 73 hours, 0 minutes, 1 second

Note that using method B, the number of seconds to event is correct, and does not need to be recomputed when the leap seconds count moves from 12 to 13 at year-end.

ANNEX D

PACKET RATES

(Normative)

D.1 Maximum cycle times

Table D.1 lists the maximum cycle time for Service Information table sections for out-of-band cable operation, when the indicated table is present.

Table D.1 Maximum cycle time for the STT, MGT, S-VCT, L-VCT and RRT

Table Section	STT	MGT	S-VCT	L-VCT	RRT
Cycle time	1 min.	500 msec.	2 min.	2 min.	1 min.

D.2 Maximum Transmission Rates

Table D.2 lists the maximum transmission rate for SI packet streams.

Table D.2 Maximum rate for each packet stream

PID	SI_base PID	any AEIT/AETT PID
Rate (bps)	150,000	150,000

D.3 MINIMUM Transmission Rates

Table D.3 lists the minimum transmission rate for SI packet streams. Minimum per-PID bit rates are required to ensure efficiency of recovery of EPG data covering the current time period (3 hours minimum) across an Extended Channel Interface, given the small number of PID values that can be used concurrently.

Table D.3 Minimum rate for each packet stream

PID	AEIT-0,1/AETT-0,1 PID
Rate (bps)	10,000

ANNEX E

DAYLIGHT SAVINGS TIME CONTROL

(Informative)

In order to convert GPS into local time, the Host needs to store a time offset (from GPS to local time) in local memory and an indicator as to whether daylight savings is observed. These two quantities can be obtained from the user interface (indicating time zone and daylight savings observance) or from the conditional access system, if present, and stored in non-volatile Host memory.

Since there is a common time (GPS) transmitted in SI, a mechanism to indicate when the Host should switch into (or out of) daylight savings time at the appropriate local time can be very useful. Once all the Hosts have transitioned at their local times, the entire system can be shifted into daylight savings time. This is accomplished by appropriate setting of the `daylight_savings` in the `daylight_savings_time_descriptor()` the STT. The basic use of daylight savings fields through the year is shown in Table E.1.

Table E.1 Basic Use of Daylight Savings Fields Through the Year

Conditions	DS status	DS_day of_month	DS_hour
At the beginning of the year (January) daylight savings is off. This is the status of the fields until:	0	0	0
<input type="checkbox"/> When the transition into daylight savings time is within less than one month, the DS_day_of_month field takes the value day_in, and the DS_hour field takes the value hour_in. The DS_status bit is 0 indicating it is not yet daylight savings time. (The transition is to occur on the day_in day of the month at hour=hour_in; for example, if the transition were on April 15 at 2 a.m., then day_in=15 and hour_in=2)	0	day_in	hour_in
<input type="checkbox"/> After all time zone daylight transitions (within the span of the network) have occurred, the DS_status bit takes the value 1, indicating that daylight savings time is on. The DS_day_of_month field and the DS_hour field take the value 0. (In the U.S., this transition has to occur no later than 7 p.m. Pacific Time on the day day_in). This is the status of the fields until:	1	0	0
When the transition out of daylight savings time is within less than one month, the DS_day_of_month field takes the value day_out, and the DS_hour field takes the value hour_out. The DS_status bit is 1 indicating it is still daylight savings time. (The transition is to occur on the day_out day of the month at hour=hour_out; for example, if the transition were on October 27 at 2 a.m., then day_out=27 and hour_out=2)	1	day_out	hour_out
<input type="checkbox"/> After all time zones (within the span of the network) have shifted out of daylight savings time, the DS_status bit takes the value 0, indicating that daylight savings time is off. The DS_day_of_month field and the DS_hour field take the value 0. (In the U.S., this transition has to occur no later than 7 p.m. Pacific Time on the day day_out). This finishes the cycle.	0	0	0

ANNEX F

STANDARD HUFFMAN TABLES FOR TEXT COMPRESSION

(Normative)

This Annex describes the compression method adopted for the transmission of English-language text strings in PSIP. The method distinguishes two types of text strings: titles and program descriptions. For each of these types, Huffman tables are defined based on 1st-order conditional probabilities. Section F.2 defines standard Huffman encode and decode tables optimized for English-language text such as that typically found in program titles. Section F.3 defines Huffman encode and decode tables optimized for English-language text such as that typically found in program descriptions. Hosts supporting the English language are expected to support decoding of text using either of these two standard Huffman compression tables.

The encode tables provide necessary and sufficient information to build the Huffman trees that need to be implemented for decoding. The decode tables described in Tables F.5 and F.7 are a particular mapping of those trees into a numerical array suitable for storage. This array can be easily implemented and used with the decoding algorithm. However, the user is free to design its own decoding tables as long as they follow the Huffman trees and rules defined in this Annex.

F.1 Character Set Definition

This compression method supports the full ISO/IEC 8859-1 (Latin-1) character set, although only characters in the ASCII range (character codes 1 to 127) can be compressed. The following characters have special definitions:

Table F.1 Characters with Special Definitions

Character	Value (Decimal)	Meaning
String Terminate (ASCII Null)	0	The <i>Terminate</i> character is used to terminate strings. The Terminate character is appended to the string in either compressed or uncompressed form. The first encoded character in a compressed string is encoded/decoded from the Terminate sub-tree. In other words, when encoding or decoding the first character in a compressed string, assume that the previous character was a Terminate character.
Order-1 Escape (ASCII ESC)	27	Used to escape from first-order context to uncompressed context. The character which follows the Escape character is uncompressed.

F.1.1 First Order Escape

The order-1 Huffman trees are *partial*, that is, codes are not defined for every possible character sequence. For example, the standard decode tables do not contain codes for the character sequence *qp*. When uncompressed text contains a character sequence which is not defined in the decode table, the order-1 escape character is used to escape back to the uncompressed context. Uncompressed symbols are coded as 8-bit ASCII (Latin I). For example, the character sequence *qpa* would be coded with *compressed q*, *compressed ESC*, *uncompressed p*, *compressed a*.

First-order escape rules for compressed strings:

- Any character which follows a first-order escape character is an uncompressed (8-bit) character. (Any character which follows an uncompressed escape character is compressed).
- Characters (128 .. 255) cannot be compressed.
- Any character which follows a character from the set (128 .. 255) is uncompressed.

F.1.2 Decode Table Data Structures

Decode tables have two sections:

- **Tree Root Offset List:** Provides the table offsets, in *bytes* from the start of the decode table, for the roots of the 128 first-order decode trees. The list is contained in bytes (0 .. 255) of the decode table, and is defined by the first “for” loop in Table F.1.
- **Order-1 Decode Trees:** Each and every character in the range (0 .. 127) has a corresponding first-order decode tree. For example, if the previous character was "s", then the decoder would use the "s" first-order decode tree (decode tree #115) to decode the next character (ASCII "s" equals 115 decimal). These 128 decode trees are delimited by the second “for” loop in Table F.2.

Decode tables have the following format:

Table F.2 Decode Table Format

Syntax	Bits	Format
decode_table() { for (i==0; i<128; i++) { byte_offset_of_char_i_tree_root } for (i==0; i<128; i++) { character_i_order_1_tree() } }	16	uimsbf
	8*M	

Note that even though the ISO Latin-1 character set supports up to 256 characters, only the first 128 characters may be represented in compressed form.

F.1.2.1 Tree Root Byte Offsets

byte_offset_of_character_i_tree_root—A 16-bit unsigned integer specifying the location, in bytes from the beginning of the decode table, of the root for the i^{th} character’s order-1 tree.

F.1.2.2 Order-1 Decode Trees

Order-1 decode trees are binary trees. The roots of the decode trees are located at the table offsets specified in the tree root offset list. The left and right children of a given node are specified as *word* offsets from the root of the tree (a *word* is equivalent to two bytes).

Decode trees have the following format:

Table F.3 Decode Tree Format

Syntax	Bits	Format
character_i_order_1_tree() {		
for (j==0; j<N; j++) {		
left_child_word_offset_or_char_leaf	8	uimsbf
right_child_word_offset_or_char_leaf	8	uimsbf
}		
}		

left_child_word_offset_or_character_leaf—An 8-bit unsigned integer number with the following interpretation: If the highest bit is cleared (i.e. bit 7 is zero), the number specifies the offset, in words, of the left child from the root of the order-1 decode tree; if the highest bit is set (bit 7 is one), the lower 7 bits give the code (e.g., in ASCII) for a leaf character.

right_child_word_offset_or_character_leaf—An 8-bit unsigned integer number with the following interpretation: If the highest bit is cleared (i.e. bit 7 is zero), the number specifies the offset, in words, of the right child from the root of the order-1 decode tree; if the highest bit is set (bit 7 is one), the lower 7 bits give the code (e.g., in ASCII) for a leaf character.

Each node (corresponding to one iteration of the for-loop) has a byte for the left child or character, and a byte for the right child or character.

Characters are *leaves* of the order-1 decode trees, and are differentiated from intermediate nodes by the byte’s most significant bit. When the most significant bit is set, the byte is a character leaf. When the most significant bit is not set, the byte contains the tabular word offset of the child node.

F.2 Standard Compression Type 1 Encode/Decode Tables

The following encode/decode tables are optimized for English-language program title text. These tables correspond to `multiple_string_structure()` with `compression_type` value `0x01`, and a mode equal to `0xFF`.

Table F.4 English-language Program Title Encode Table

Prior Symbol: 0 Symbol: 27 Code: 11001011	Prior Symbol: '' Symbol: '&' Code: 010001	Prior Symbol: '*' Symbol: 'S' Code: 11
Prior Symbol: 0 Symbol: 'S' Code: 1100101011	Prior Symbol: '' Symbol: '' Code: 010000100	Prior Symbol: '+' Symbol: 27 Code: 1
Prior Symbol: 0 Symbol: '2' Code: 011010010	Prior Symbol: '' Symbol: ':' Code: 00000001	Prior Symbol: ',' Symbol: 27 Code: 0
Prior Symbol: 0 Symbol: '4' Code: 1100101010	Prior Symbol: '' Symbol: '1' Code: 010000101	Prior Symbol: '.' Symbol: '' Code: 1
Prior Symbol: 0 Symbol: '7' Code: 011010011	Prior Symbol: '' Symbol: '2' Code: 00000010	Prior Symbol: '/' Symbol: 27 Code: 01
Prior Symbol: 0 Symbol: 'A' Code: 0111	Prior Symbol: '' Symbol: '3' Code: 01000001	Prior Symbol: '' Symbol: '' Code: 111
Prior Symbol: 0 Symbol: 'B' Code: 1001	Prior Symbol: '' Symbol: '9' Code: 00000000	Prior Symbol: '' Symbol: ':' Code: 1101
Prior Symbol: 0 Symbol: 'C' Code: 1011	Prior Symbol: '' Symbol: 'A' Code: 10111	Prior Symbol: '' Symbol: '1' Code: 1000
Prior Symbol: 0 Symbol: 'D' Code: 11011	Prior Symbol: '' Symbol: 'B' Code: 0010	Prior Symbol: '' Symbol: 'A' Code: 001
Prior Symbol: 0 Symbol: 'E' Code: 10001	Prior Symbol: '' Symbol: 'C' Code: 1100	Prior Symbol: '' Symbol: 'M' Code: 000
Prior Symbol: 0 Symbol: 'F' Code: 11000	Prior Symbol: '' Symbol: 'D' Code: 11100	Prior Symbol: '' Symbol: 'R' Code: 1001
Prior Symbol: 0 Symbol: 'G' Code: 11100	Prior Symbol: '' Symbol: 'E' Code: 011010	Prior Symbol: '' Symbol: 'S' Code: 1010
Prior Symbol: 0 Symbol: 'H' Code: 11111	Prior Symbol: '' Symbol: 'F' Code: 10011	Prior Symbol: '' Symbol: 'T' Code: 1011
Prior Symbol: 0 Symbol: 'I' Code: 10000	Prior Symbol: '' Symbol: 'G' Code: 00001	Prior Symbol: '' Symbol: 'U' Code: 1100
Prior Symbol: 0 Symbol: 'J' Code: 01100	Prior Symbol: '' Symbol: 'H' Code: 10101	Prior Symbol: '' Symbol: 0 Code: 111
Prior Symbol: 0 Symbol: 'K' Code: 1100110	Prior Symbol: '' Symbol: 'I' Code: 111111	Prior Symbol: '' Symbol: 27 Code: 101
Prior Symbol: 0 Symbol: 'L' Code: 11101	Prior Symbol: '' Symbol: 'J' Code: 111110	Prior Symbol: '' Symbol: '' Code: 0
Prior Symbol: 0 Symbol: 'M' Code: 1010	Prior Symbol: '' Symbol: 'K' Code: 010011	Prior Symbol: '' Symbol: '' Code: 110
Prior Symbol: 0 Symbol: 'N' Code: 0011	Prior Symbol: '' Symbol: 'L' Code: 11110	Prior Symbol: '' Symbol: 'T' Code: 10010
Prior Symbol: 0 Symbol: 'O' Code: 011011	Prior Symbol: '' Symbol: 'M' Code: 0101	Prior Symbol: '' Symbol: 'S' Code: 1000
Prior Symbol: 0 Symbol: 'P' Code: 11110	Prior Symbol: '' Symbol: 'N' Code: 10110	Prior Symbol: '' Symbol: 'W' Code: 10011
Prior Symbol: 0 Symbol: 'Q' Code: 01101000	Prior Symbol: '' Symbol: 'O' Code: 011011	Prior Symbol: '/' Symbol: 27 Code: 1
Prior Symbol: 0 Symbol: 'R' Code: 11010	Prior Symbol: '' Symbol: 'P' Code: 11101	Prior Symbol: '0' Symbol: 0 Code: 01
Prior Symbol: 0 Symbol: 'S' Code: 000	Prior Symbol: '' Symbol: 'Q' Code: 100100011	Prior Symbol: '0' Symbol: 27 Code: 001
Prior Symbol: 0 Symbol: 'T' Code: 010	Prior Symbol: '' Symbol: 'R' Code: 10100	Prior Symbol: '0' Symbol: '' Code: 10
Prior Symbol: 0 Symbol: 'U' Code: 0110101	Prior Symbol: '' Symbol: 'S' Code: 1101	Prior Symbol: '0' Symbol: ':' Code: 000
Prior Symbol: 0 Symbol: 'V' Code: 1100111	Prior Symbol: '' Symbol: 'T' Code: 1000	Prior Symbol: '0' Symbol: '0' Code: 11
Prior Symbol: 0 Symbol: 'W' Code: 0010	Prior Symbol: '' Symbol: 'U' Code: 1001001	Prior Symbol: '!' Symbol: 0 Code: 010
Prior Symbol: 0 Symbol: 'Y' Code: 1100100	Prior Symbol: '' Symbol: 'V' Code: 1001011	Prior Symbol: '!' Symbol: 27 Code: 011
Prior Symbol: 0 Symbol: 'Z' Code: 110010100	Prior Symbol: '' Symbol: 'W' Code: 0011	Prior Symbol: '!' Symbol: '' Code: 110
Prior Symbol: 1 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'X' Code: 0000000010	Prior Symbol: '!' Symbol: '0' Code: 111
Prior Symbol: 2 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'Y' Code: 000001	Prior Symbol: '!' Symbol: '1' Code: 100
Prior Symbol: 3 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'Z' Code: 00000011	Prior Symbol: '!' Symbol: '2' Code: 101
Prior Symbol: 4 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'a' Code: 01100	Prior Symbol: '!' Symbol: '9' Code: 00
Prior Symbol: 5 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'b' Code: 10010101	Prior Symbol: '2' Symbol: 0 Code: 11
Prior Symbol: 6 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'c' Code: 01000000	Prior Symbol: '2' Symbol: 27 Code: 10
Prior Symbol: 7 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'd' Code: 01000011	Prior Symbol: '2' Symbol: '0' Code: 01
Prior Symbol: 8 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'e' Code: 0000000011	Prior Symbol: '2' Symbol: '1' Code: 000
Prior Symbol: 9 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'f' Code: 10010000	Prior Symbol: '2' Symbol: '' Code: 001
Prior Symbol: 10 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'g' Code: 10010010	Prior Symbol: '3' Symbol: 0 Code: 0
Prior Symbol: 11 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'h' Code: 100100010	Prior Symbol: '3' Symbol: 27 Code: 11
Prior Symbol: 12 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'i' Code: 0001	Prior Symbol: '3' Symbol: '0' Code: 10
Prior Symbol: 13 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'j' Code: 0111	Prior Symbol: '3' Symbol: '1' Code: 0
Prior Symbol: 14 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'k' Code: 100100010	Prior Symbol: '4' Symbol: 27 Code: 1
Prior Symbol: 15 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'l' Code: 0	Prior Symbol: '4' Symbol: '8' Code: 1
Prior Symbol: 16 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'm' Code: 0001	Prior Symbol: '5' Symbol: 27 Code: 1
Prior Symbol: 17 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'n' Code: 0111	Prior Symbol: '5' Symbol: '1' Code: 100
Prior Symbol: 18 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'o' Code: 1	Prior Symbol: '6' Symbol: 27 Code: 1
Prior Symbol: 19 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'p' Code: 0	Prior Symbol: '7' Symbol: 27 Code: 0
Prior Symbol: 20 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'q' Code: 01	Prior Symbol: '7' Symbol: '0' Code: 1
Prior Symbol: 21 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'r' Code: 00	Prior Symbol: '8' Symbol: 27 Code: 0
Prior Symbol: 22 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 's' Code: 1	Prior Symbol: '8' Symbol: '' Code: 1
Prior Symbol: 23 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 't' Code: 1	Prior Symbol: '9' Symbol: 27 Code: 11
Prior Symbol: 24 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'u' Code: 0	Prior Symbol: '9' Symbol: '0' Code: 01
Prior Symbol: 25 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'v' Code: 1	Prior Symbol: '9' Symbol: '1' Code: 100
Prior Symbol: 26 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'w' Code: 0	Prior Symbol: '9' Symbol: '3' Code: 101
Prior Symbol: 27 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'x' Code: 1	Prior Symbol: '9' Symbol: '9' Code: 00
Prior Symbol: 28 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'y' Code: 1	Prior Symbol: ':' Symbol: 27 Code: 0
Prior Symbol: 29 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'z' Code: 0	Prior Symbol: '' Symbol: '' Code: 1
Prior Symbol: 30 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '0' Code: 0001	Prior Symbol: ';' Symbol: 27 Code: 1
Prior Symbol: 31 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '1' Code: 1	Prior Symbol: '<' Symbol: 27 Code: 1
Prior Symbol: '' Symbol: 27 Code: 10010100	Prior Symbol: '' Symbol: '2' Code: 0	Prior Symbol: '=' Symbol: 27 Code: 1
	Prior Symbol: '' Symbol: '3' Code: 1	Prior Symbol: '>' Symbol: 27 Code: 1
	Prior Symbol: '' Symbol: '4' Code: 0000	Prior Symbol: '?' Symbol: 0 Code: 1
	Prior Symbol: '' Symbol: '5' Code: 1	Prior Symbol: '?' Symbol: 27 Code: 0
	Prior Symbol: '' Symbol: '6' Code: 001	Prior Symbol: '@' Symbol: 27 Code: 1
	Prior Symbol: '' Symbol: '7' Code: 1	Prior Symbol: 'A' Symbol: 27 Code: 00010
	Prior Symbol: '' Symbol: '8' Code: 1	Prior Symbol: 'A' Symbol: '' Code: 010
	Prior Symbol: '' Symbol: '9' Code: 001	Prior Symbol: 'A' Symbol: '*' Code: 1101000
	Prior Symbol: '' Symbol: 'A' Code: 01	
	Prior Symbol: '' Symbol: 'B' Code: 10	

Prior Symbol: 'l' Symbol: 'e' Code: 110
 Prior Symbol: 'l' Symbol: 'f' Code: 1111000
 Prior Symbol: 'l' Symbol: 'g' Code: 001
 Prior Symbol: 'l' Symbol: 'k' Code: 011001
 Prior Symbol: 'l' Symbol: 't' Code: 101
 Prior Symbol: 'l' Symbol: 'm' Code: 1111010
 Prior Symbol: 'l' Symbol: 'o' Code: 11111
 Prior Symbol: 'l' Symbol: 'r' Code: 11110010
 Prior Symbol: 'l' Symbol: 's' Code: 01101
 Prior Symbol: 'l' Symbol: 't' Code: 011101
 Prior Symbol: 'l' Symbol: 'u' Code: 01111
 Prior Symbol: 'l' Symbol: 'v' Code: 1111011
 Prior Symbol: 'l' Symbol: 'w' Code: 01110001
 Prior Symbol: 'l' Symbol: 'y' Code: 1001
 Prior Symbol: 'm' Symbol: '0' Code: 0100
 Prior Symbol: 'm' Symbol: '27' Code: 010101
 Prior Symbol: 'm' Symbol: '' Code: 001
 Prior Symbol: 'm' Symbol: 'a' Code: 101
 Prior Symbol: 'm' Symbol: 'b' Code: 0000
 Prior Symbol: 'm' Symbol: 'e' Code: 11
 Prior Symbol: 'm' Symbol: 'f' Code: 011
 Prior Symbol: 'm' Symbol: 'm' Code: 0001
 Prior Symbol: 'm' Symbol: 'o' Code: 1001
 Prior Symbol: 'm' Symbol: 'p' Code: 1000
 Prior Symbol: 'm' Symbol: 's' Code: 010111
 Prior Symbol: 'm' Symbol: 'u' Code: 010110
 Prior Symbol: 'm' Symbol: 'y' Code: 010100
 Prior Symbol: 'n' Symbol: '0' Code: 000
 Prior Symbol: 'n' Symbol: '27' Code: 01110011
 Prior Symbol: 'n' Symbol: '' Code: 110
 Prior Symbol: 'n' Symbol: 'm' Code: 011101
 Prior Symbol: 'n' Symbol: 't' Code: 1001010
 Prior Symbol: 'n' Symbol: 'a' Code: 11100
 Prior Symbol: 'n' Symbol: 'b' Code: 111010000
 Prior Symbol: 'n' Symbol: 'c' Code: 01111
 Prior Symbol: 'n' Symbol: 'd' Code: 001
 Prior Symbol: 'n' Symbol: 'e' Code: 010
 Prior Symbol: 'n' Symbol: 'f' Code: 1001011
 Prior Symbol: 'n' Symbol: 'g' Code: 101
 Prior Symbol: 'n' Symbol: 'h' Code: 111010101
 Prior Symbol: 'n' Symbol: 'i' Code: 1000
 Prior Symbol: 'n' Symbol: 'j' Code: 111010001
 Prior Symbol: 'n' Symbol: 'k' Code: 1110110
 Prior Symbol: 'n' Symbol: 'l' Code: 111010110
 Prior Symbol: 'n' Symbol: 'm' Code: 111010111
 Prior Symbol: 'n' Symbol: 'n' Code: 10011
 Prior Symbol: 'n' Symbol: 'o' Code: 1110111
 Prior Symbol: 'n' Symbol: 'r' Code: 111010100
 Prior Symbol: 'n' Symbol: 's' Code: 0110
 Prior Symbol: 'n' Symbol: 't' Code: 1111
 Prior Symbol: 'n' Symbol: 'u' Code: 11101001
 Prior Symbol: 'n' Symbol: 'v' Code: 0111000
 Prior Symbol: 'n' Symbol: 'y' Code: 100100
 Prior Symbol: 'n' Symbol: 'z' Code: 01110010
 Prior Symbol: 'o' Symbol: '0' Code: 00101
 Prior Symbol: 'o' Symbol: '27' Code: 01110001
 Prior Symbol: 'o' Symbol: '' Code: 0101
 Prior Symbol: 'o' Symbol: 'm' Code: 01110000
 Prior Symbol: 'o' Symbol: 'n' Code: 011101010
 Prior Symbol: 'o' Symbol: '?' Code: 011101100
 Prior Symbol: 'o' Symbol: 'a' Code: 1100010
 Prior Symbol: 'o' Symbol: 'b' Code: 001001
 Prior Symbol: 'o' Symbol: 'c' Code: 110000
 Prior Symbol: 'o' Symbol: 'd' Code: 01111
 Prior Symbol: 'o' Symbol: 'e' Code: 0111001
 Prior Symbol: 'o' Symbol: 'f' Code: 1001
 Prior Symbol: 'o' Symbol: 'g' Code: 00010
 Prior Symbol: 'o' Symbol: 'h' Code: 0111010
 Prior Symbol: 'o' Symbol: 'i' Code: 01110111
 Prior Symbol: 'o' Symbol: 'k' Code: 1100011
 Prior Symbol: 'o' Symbol: 'l' Code: 0100
 Prior Symbol: 'o' Symbol: 'm' Code: 1000
 Prior Symbol: 'o' Symbol: 'n' Code: 111
 Prior Symbol: 'o' Symbol: 'o' Code: 0011
 Prior Symbol: 'o' Symbol: 'p' Code: 01101
 Prior Symbol: 'o' Symbol: 'r' Code: 101
 Prior Symbol: 'o' Symbol: 's' Code: 11001
 Prior Symbol: 'o' Symbol: 't' Code: 00011
 Prior Symbol: 'o' Symbol: 'u' Code: 1101
 Prior Symbol: 'o' Symbol: 'v' Code: 01100
 Prior Symbol: 'o' Symbol: 'w' Code: 0000
 Prior Symbol: 'o' Symbol: 'y' Code: 0010001
 Prior Symbol: 'o' Symbol: 'z' Code: 011101011
 Prior Symbol: 'p' Symbol: '0' Code: 1101
 Prior Symbol: 'p' Symbol: '27' Code: 101110
 Prior Symbol: 'p' Symbol: '' Code: 010
 Prior Symbol: 'p' Symbol: 'm' Code: 1100101
 Prior Symbol: 'p' Symbol: 'a' Code: 1001
 Prior Symbol: 'p' Symbol: 'd' Code: 101111
 Prior Symbol: 'p' Symbol: 'e' Code: 111
 Prior Symbol: 'p' Symbol: 'h' Code: 11000
 Prior Symbol: 'p' Symbol: 'i' Code: 1010
 Prior Symbol: 'p' Symbol: 'l' Code: 0110
 Prior Symbol: 'p' Symbol: 'm' Code: 1100100
 Prior Symbol: 'p' Symbol: 'o' Code: 00
 Prior Symbol: 'p' Symbol: 'u' Code: 0111
 Prior Symbol: 'p' Symbol: 'r' Code: 10001
 Prior Symbol: 'p' Symbol: 's' Code: 10000
 Prior Symbol: 'p' Symbol: 't' Code: 10110
 Prior Symbol: 'p' Symbol: 'y' Code: 110011
 Prior Symbol: 'q' Symbol: '27' Code: 0
 Prior Symbol: 'q' Symbol: 'u' Code: 1
 Prior Symbol: 'r' Symbol: '0' Code: 1001
 Prior Symbol: 'r' Symbol: '27' Code: 01100101
 Prior Symbol: 'r' Symbol: '' Code: 1111
 Prior Symbol: 'r' Symbol: 'm' Code: 0110011
 Prior Symbol: 'r' Symbol: 'n' Code: 110011101
 Prior Symbol: 'r' Symbol: 't' Code: 0111100
 Prior Symbol: 'r' Symbol: 'u' Code: 110011100
 Prior Symbol: 'r' Symbol: 'a' Code: 000
 Prior Symbol: 'r' Symbol: 'b' Code: 01111101
 Prior Symbol: 'r' Symbol: 'c' Code: 0111111
 Prior Symbol: 'r' Symbol: 'd' Code: 11000
 Prior Symbol: 'r' Symbol: 'e' Code: 101
 Prior Symbol: 'r' Symbol: 'f' Code: 11001111
 Prior Symbol: 'r' Symbol: 'g' Code: 0111101
 Prior Symbol: 'r' Symbol: 'h' Code: 010
 Prior Symbol: 'r' Symbol: 'k' Code: 110010
 Prior Symbol: 'r' Symbol: 'l' Code: 0011
 Prior Symbol: 'r' Symbol: 'm' Code: 011000
 Prior Symbol: 'r' Symbol: 'n' Code: 01101
 Prior Symbol: 'r' Symbol: 'o' Code: 1101
 Prior Symbol: 'r' Symbol: 'p' Code: 01111100
 Prior Symbol: 'r' Symbol: 'r' Code: 01110
 Prior Symbol: 'r' Symbol: 's' Code: 1110
 Prior Symbol: 'r' Symbol: 't' Code: 1000
 Prior Symbol: 'r' Symbol: 'u' Code: 1100110
 Prior Symbol: 'r' Symbol: 'v' Code: 01100100
 Prior Symbol: 'r' Symbol: 'y' Code: 0010
 Prior Symbol: 's' Symbol: '0' Code: 11
 Prior Symbol: 's' Symbol: '27' Code: 0010011
 Prior Symbol: 's' Symbol: '' Code: 01
 Prior Symbol: 's' Symbol: 'm' Code: 001011010
 Prior Symbol: 's' Symbol: 'n' Code: 001011011
 Prior Symbol: 's' Symbol: 't' Code: 00100101
 Prior Symbol: 's' Symbol: 'u' Code: 0000001
 Prior Symbol: 's' Symbol: '?' Code: 001011100
 Prior Symbol: 's' Symbol: 'C' Code: 001011101
 Prior Symbol: 's' Symbol: 'H' Code: 001011110
 Prior Symbol: 's' Symbol: 'a' Code: 101010
 Prior Symbol: 's' Symbol: 'd' Code: 101011
 Prior Symbol: 's' Symbol: 'f' Code: 001011111
 Prior Symbol: 's' Symbol: 'e' Code: 1011
 Prior Symbol: 's' Symbol: 'f' Code: 0000000
 Prior Symbol: 's' Symbol: 'h' Code: 00001
 Prior Symbol: 's' Symbol: 'i' Code: 0011
 Prior Symbol: 's' Symbol: 'k' Code: 000001
 Prior Symbol: 's' Symbol: 'l' Code: 00101010
 Prior Symbol: 's' Symbol: 'm' Code: 00000001
 Prior Symbol: 's' Symbol: 'n' Code: 001011011
 Prior Symbol: 's' Symbol: 'o' Code: 10100
 Prior Symbol: 's' Symbol: 'p' Code: 00100100
 Prior Symbol: 's' Symbol: 'r' Code: 0001
 Prior Symbol: 's' Symbol: 't' Code: 0010100
 Prior Symbol: 's' Symbol: 'u' Code: 001010010
 Prior Symbol: 's' Symbol: 'v' Code: 001010011
 Prior Symbol: 's' Symbol: 'w' Code: 001010010
 Prior Symbol: 's' Symbol: 'x' Code: 001010011
 Prior Symbol: 's' Symbol: 'y' Code: 001010010
 Prior Symbol: 's' Symbol: 'z' Code: 001010011
 Prior Symbol: 't' Symbol: '0' Code: 010
 Prior Symbol: 't' Symbol: '27' Code: 0010100
 Prior Symbol: 't' Symbol: '' Code: 001110
 Prior Symbol: 't' Symbol: 'a' Code: 0000
 Prior Symbol: 't' Symbol: 'b' Code: 100000
 Prior Symbol: 't' Symbol: 'c' Code: 1101101
 Prior Symbol: 't' Symbol: 'd' Code: 11000000
 Prior Symbol: 't' Symbol: 'e' Code: 011
 Prior Symbol: 't' Symbol: 'f' Code: 111
 Prior Symbol: 't' Symbol: 'g' Code: 11000011
 Prior Symbol: 't' Symbol: 'h' Code: 1101101
 Prior Symbol: 't' Symbol: 'i' Code: 11000000
 Prior Symbol: 't' Symbol: 'j' Code: 1011
 Prior Symbol: 't' Symbol: 'k' Code: 0010100
 Prior Symbol: 't' Symbol: 'l' Code: 101
 Prior Symbol: 't' Symbol: 'm' Code: 00000001
 Prior Symbol: 't' Symbol: 'n' Code: 110110101
 Prior Symbol: 't' Symbol: 'o' Code: 10100
 Prior Symbol: 't' Symbol: 'p' Code: 0010100
 Prior Symbol: 't' Symbol: 'q' Code: 110110000
 Prior Symbol: 't' Symbol: 'r' Code: 10000000
 Prior Symbol: 't' Symbol: 's' Code: 011
 Prior Symbol: 't' Symbol: 'u' Code: 11000011
 Prior Symbol: 't' Symbol: 'v' Code: 110110100
 Prior Symbol: 't' Symbol: 'w' Code: 110110000
 Prior Symbol: 't' Symbol: 'x' Code: 110110001
 Prior Symbol: 't' Symbol: 'y' Code: 110110000
 Prior Symbol: 't' Symbol: 'z' Code: 110110001
 Prior Symbol: 'u' Symbol: '0' Code: 0011110
 Prior Symbol: 'u' Symbol: '27' Code: 000100
 Prior Symbol: 'u' Symbol: '' Code: 001110
 Prior Symbol: 'u' Symbol: 'a' Code: 00110
 Prior Symbol: 'u' Symbol: 'b' Code: 10011
 Prior Symbol: 'u' Symbol: 'c' Code: 11100
 Prior Symbol: 'u' Symbol: 'd' Code: 10000
 Prior Symbol: 'u' Symbol: 'e' Code: 0010
 Prior Symbol: 'u' Symbol: 'f' Code: 0011111
 Prior Symbol: 'u' Symbol: 'g' Code: 11101
 Prior Symbol: 'u' Symbol: 'h' Code: 00011
 Prior Symbol: 'u' Symbol: 'i' Code: 000110
 Prior Symbol: 'u' Symbol: 'j' Code: 00011
 Prior Symbol: 'u' Symbol: 'k' Code: 0001010
 Prior Symbol: 'u' Symbol: 'l' Code: 0000
 Prior Symbol: 'u' Symbol: 'm' Code: 10010
 Prior Symbol: 'u' Symbol: 'n' Code: 110
 Prior Symbol: 'u' Symbol: 'p' Code: 10001
 Prior Symbol: 'u' Symbol: 'q' Code: 01
 Prior Symbol: 'u' Symbol: 'r' Code: 101
 Prior Symbol: 'u' Symbol: 's' Code: 1111
 Prior Symbol: 'u' Symbol: 't' Code: 1111
 Prior Symbol: 'u' Symbol: 'z' Code: 0001011
 Prior Symbol: 'v' Symbol: '27' Code: 0010
 Prior Symbol: 'v' Symbol: 'a' Code: 000
 Prior Symbol: 'v' Symbol: 'e' Code: 1
 Prior Symbol: 'v' Symbol: 'f' Code: 01
 Prior Symbol: 'v' Symbol: 'o' Code: 00111
 Prior Symbol: 'v' Symbol: 's' Code: 00110
 Prior Symbol: 'v' Symbol: '0' Code: 001
 Prior Symbol: 'v' Symbol: '27' Code: 01010
 Prior Symbol: 'v' Symbol: '' Code: 011
 Prior Symbol: 'v' Symbol: 'm' Code: 010010
 Prior Symbol: 'v' Symbol: 'n' Code: 000
 Prior Symbol: 'v' Symbol: 'a' Code: 000
 Prior Symbol: 'v' Symbol: 'b' Code: 010011
 Prior Symbol: 'v' Symbol: 'c' Code: 010111
 Prior Symbol: 'v' Symbol: 'e' Code: 1111
 Prior Symbol: 'v' Symbol: 'f' Code: 1100
 Prior Symbol: 'v' Symbol: 't' Code: 010110
 Prior Symbol: 'v' Symbol: 'n' Code: 1110
 Prior Symbol: 'v' Symbol: 'o' Code: 1101
 Prior Symbol: 'v' Symbol: 'r' Code: 01000
 Prior Symbol: 'v' Symbol: 's' Code: 10
 Prior Symbol: 'x' Symbol: '0' Code: 110
 Prior Symbol: 'x' Symbol: '27' Code: 1010
 Prior Symbol: 'x' Symbol: '' Code: 1011
 Prior Symbol: 'x' Symbol: 'a' Code: 000
 Prior Symbol: 'x' Symbol: 'e' Code: 001

Prior Symbol: 'x' Symbol: 'i' Code: 100	Prior Symbol: 'y' Symbol: 'e' Code: 11001	Prior Symbol: 'z' Symbol: '' Code: 000
Prior Symbol: 'x' Symbol: 'j' Code: 111	Prior Symbol: 'y' Symbol: 'i' Code: 1100001	Prior Symbol: 'z' Symbol: 'a' Code: 01
Prior Symbol: 'x' Symbol: 't' Code: 01	Prior Symbol: 'y' Symbol: 'l' Code: 111111	Prior Symbol: 'z' Symbol: 'e' Code: 1010
Prior Symbol: 'y' Symbol: '0' Code: 10	Prior Symbol: 'y' Symbol: 'm' Code: 1101111	Prior Symbol: 'z' Symbol: 'i' Code: 111
Prior Symbol: 'y' Symbol: '27' Code: 111110	Prior Symbol: 'y' Symbol: 'n' Code: 1100010	Prior Symbol: 'z' Symbol: 'y' Code: 001
Prior Symbol: 'y' Symbol: '' Code: 0	Prior Symbol: 'y' Symbol: 'o' Code: 1100011	Prior Symbol: 'z' Symbol: 'z' Code: 1011
Prior Symbol: 'y' Symbol: '!' Code: 1101101	Prior Symbol: 'y' Symbol: 'p' Code: 1101000	Prior Symbol: '}' Symbol: '27' Code: 1
Prior Symbol: 'y' Symbol: '"' Code: 110101	Prior Symbol: 'y' Symbol: 's' Code: 1110	Prior Symbol: ' ' Symbol: '27' Code: 1
Prior Symbol: 'y' Symbol: '' Code: 11110101	Prior Symbol: 'y' Symbol: 't' Code: 1101001	Prior Symbol: '}' Symbol: '27' Code: 1
Prior Symbol: 'y' Symbol: 'a' Code: 1101110	Prior Symbol: 'y' Symbol: 'v' Code: 1101100	Prior Symbol: '^' Symbol: '27' Code: 1
Prior Symbol: 'y' Symbol: 'b' Code: 1111011	Prior Symbol: 'y' Symbol: 'w' Code: 111100	Prior Symbol: '127' Symbol: '27' Code: 1
Prior Symbol: 'y' Symbol: 'c' Code: 11110100	Prior Symbol: 'z' Symbol: '0' Code: 110	
Prior Symbol: 'y' Symbol: 'd' Code: 1100000	Prior Symbol: 'z' Symbol: '27' Code: 100	

Table F.5 English-language Program Title Decode Table

0 1	62 1	124 2	186 4	248 7	310 23	372 155
1 0	63 118	125 78	187 20	249 140	311 24	373 155
2 1	64 1	126 2	188 4	250 7	312 25	374 155
3 58	65 120	127 80	189 22	251 142	313 26	375 155
4 1	66 1	128 2	190 4	252 7	314 155	376 41
5 60	67 206	129 82	191 24	253 144	315 155	377 42
6 1	68 1	130 2	192 4	254 7	316 155	378 216
7 62	69 210	131 84	193 26	255 146	317 155	379 229
8 1	70 1	132 2	194 4	256 27	318 155	380 185
9 64	71 212	133 126	195 28	257 28	319 155	381 1
10 1	72 1	134 2	196 4	258 180	320 155	382 167
11 66	73 214	135 146	197 82	259 164	321 155	383 177
12 1	74 1	136 2	198 4	260 178	322 155	384 236
13 68	75 216	137 172	199 106	261 183	323 155	385 209
14 1	76 1	138 2	200 4	262 218	324 155	386 2
15 70	77 218	139 186	201 142	263 1	325 155	387 173
16 1	78 1	140 2	202 4	264 209	326 155	388 178
17 72	79 220	141 210	203 174	265 2	327 155	389 218
18 1	80 1	142 2	204 4	266 3	328 155	390 227
19 74	81 230	143 228	205 238	267 155	329 155	391 179
20 1	82 1	144 2	206 5	268 4	330 155	392 3
21 76	83 232	145 250	207 6	269 213	331 155	393 228
22 1	84 1	146 3	208 5	270 217	332 155	394 230
23 78	85 234	147 6	209 40	271 5	333 155	395 4
24 1	86 1	148 3	210 5	272 203	334 155	396 155
25 80	87 240	149 30	211 68	273 214	335 155	397 226
26 1	88 1	150 3	212 5	274 6	336 155	398 5
27 82	89 242	151 38	213 114	275 207	337 155	399 6
28 1	90 1	152 3	214 5	276 7	338 155	400 7
29 84	91 244	153 50	215 118	277 8	339 155	401 8
30 1	92 2	154 3	216 5	278 202	340 155	402 9
31 86	93 6	155 62	217 144	279 9	341 155	403 213
32 1	94 2	156 3	218 5	280 201	342 155	404 10
33 88	95 18	157 82	219 190	281 197	343 155	405 214
34 1	96 2	158 3	220 5	282 198	344 155	406 11
35 90	97 20	159 100	221 214	283 10	345 155	407 217
36 1	98 2	160 3	222 6	284 210	346 155	408 12
37 92	99 28	161 122	223 10	285 196	347 155	409 166
38 1	100 2	162 3	224 6	286 199	348 155	410 233
39 94	101 40	163 148	225 68	287 204	349 155	411 203
40 1	102 2	164 3	226 6	288 208	350 155	412 197
41 96	103 48	165 152	227 100	289 200	351 155	413 207
42 1	104 2	166 3	228 6	290 215	352 155	414 13
43 98	105 52	167 164	229 102	291 206	353 155	415 14
44 1	106 2	168 3	230 6	292 11	354 155	416 202
45 100	107 54	169 200	231 154	293 193	355 155	417 201
46 1	108 2	170 3	232 6	294 12	356 155	418 15
47 102	109 56	171 222	233 208	295 194	357 155	419 199
48 1	110 2	172 3	234 6	296 205	358 155	420 16
49 104	111 58	173 230	235 252	297 195	359 155	421 17
50 1	112 2	174 3	236 7	298 13	360 155	422 225
51 106	113 60	175 244	237 34	299 14	361 155	423 18
52 1	114 2	176 4	238 7	300 15	362 155	424 19
53 108	115 62	177 4	239 44	301 16	363 155	425 198
54 1	116 2	178 4	240 7	302 211	364 155	426 210
55 110	117 70	179 6	241 70	303 17	365 155	427 200
56 1	118 2	180 4	242 7	304 212	366 155	428 206
57 112	119 72	181 12	243 84	305 18	367 155	429 193
58 1	120 2	182 4	244 7	306 19	368 155	430 196
59 114	121 74	183 16	245 124	307 20	369 155	431 208
60 1	122 2	184 4	246 7	308 21	370 155	432 204
61 116	123 76	185 18	247 138	309 22	371 155	433 20

434 21	512 3	590 155	668 233	746 232	824 2	902 3
435 239	513 160	591 155	669 2	747 1	825 229	903 4
436 194	514 4	592 155	670 160	748 155	826 239	904 5
437 215	515 155	593 128	671 3	749 2	827 3	905 233
438 22	516 5	594 155	672 4	750 3	828 225	906 236
439 205	517 6	595 155	673 5	751 4	829 233	907 6
440 23	518 160	596 19	674 242	752 225	830 8	908 229
441 244	519 5	597 20	675 6	753 245	831 9	909 7
442 212	520 201	598 170	676 236	754 233	832 170	910 239
443 24	521 215	599 173	677 7	755 5	833 212	911 8
444 25	522 211	600 174	678 225	756 229	834 1	912 225
445 26	523 1	601 246	679 8	757 6	835 155	913 9
446 195	524 2	602 231	680 9	758 242	836 227	914 242
447 211	525 155	603 244	681 232	759 239	837 2	915 10
448 27	526 174	604 226	682 10	760 7	838 242	916 1
449 28	527 128	605 233	683 239	761 8	839 3	917 245
450 29	528 3	606 1	684 5	762 239	840 229	918 155
451 30	529 4	607 2	685 6	763 5	841 4	919 214
452 31	530 155	608 194	686 249	764 128	842 245	920 4
453 32	531 155	609 240	687 155	765 155	843 249	921 5
454 33	532 2	610 155	688 1	766 245	844 233	922 232
455 34	533 3	611 243	689 245	767 1	845 5	923 155
456 35	534 173	612 227	690 2	768 2	846 239	924 1
457 36	535 155	613 230	691 242	769 233	847 6	925 245
458 37	536 1	614 247	692 233	770 225	848 7	926 2
459 38	537 128	615 3	693 229	771 3	849 225	927 225
460 39	538 160	616 245	694 239	772 229	850 229	928 233
461 40	539 176	617 4	695 3	773 4	851 8	929 239
462 1	540 4	618 5	696 225	774 238	852 206	930 3
463 128	541 5	619 6	697 4	775 11	853 160	931 229
464 160	542 128	620 242	698 10	776 186	854 198	932 16
465 155	543 155	621 7	699 11	777 212	855 245	933 17
466 155	544 177	622 8	700 241	778 174	856 1	934 170
467 155	545 178	623 9	701 245	779 242	857 2	935 236
468 155	546 160	624 10	702 243	780 227	858 155	936 241
469 155	547 176	625 11	703 1	781 1	859 194	937 174
470 177	548 185	626 12	704 237	782 160	860 3	938 160
471 155	549 1	627 228	705 249	783 2	861 225	939 247
472 155	550 2	628 160	706 195	784 128	862 4	940 237
473 155	551 3	629 13	707 2	785 155	863 239	941 238
474 155	552 2	630 236	708 236	786 237	864 5	942 1
475 160	553 3	631 238	709 238	787 3	865 233	943 2
476 4	554 177	632 14	710 228	788 201	866 6	944 155
477 243	555 186	633 237	711 248	789 243	867 7	945 235
478 228	556 1	634 15	712 3	790 244	868 9	946 3
479 185	557 176	635 16	713 155	791 4	869 10	947 4
480 1	558 155	636 17	714 246	792 5	870 228	948 5
481 244	559 128	637 18	715 4	793 6	871 243	949 6
482 160	560 128	638 8	716 5	794 7	872 230	950 227
483 155	561 1	639 9	717 225	795 8	873 246	951 7
484 2	562 176	640 193	718 6	796 9	874 247	952 239
485 3	563 155	641 211	719 7	797 10	875 240	953 8
486 155	564 155	642 155	720 8	798 2	876 242	954 233
487 155	565 184	643 1	721 9	799 3	877 1	955 245
488 155	566 155	644 195	722 7	800 155	878 236	956 9
489 155	567 155	645 2	723 8	801 245	879 2	957 225
490 1	568 155	646 233	724 160	802 1	880 3	958 229
491 2	569 155	647 236	725 155	803 225	881 160	959 240
492 155	570 155	648 3	726 204	804 239	882 155	960 232
493 193	571 176	649 242	727 1	805 229	883 4	961 10
494 200	572 155	650 245	728 229	806 5	884 5	962 11
495 211	573 160	651 4	729 2	807 233	885 245	963 12
496 155	574 2	652 239	730 236	808 225	886 6	964 13
497 155	575 3	653 225	731 245	809 239	887 7	965 244
498 155	576 177	654 5	732 239	810 245	888 238	966 14
499 160	577 179	655 229	733 3	811 238	889 8	967 15
500 7	578 185	656 6	734 233	812 155	890 11	968 232
501 8	579 176	657 7	735 242	813 229	891 12	969 10
502 177	580 1	658 11	736 4	814 1	892 160	970 173
503 210	581 155	659 12	737 5	815 2	893 243	971 206
504 211	582 155	660 193	738 225	816 3	894 249	972 155
505 212	583 160	661 249	739 6	817 4	895 174	973 1
506 213	584 155	662 1	740 9	818 4	896 210	974 214
507 173	585 155	663 194	741 10	819 5	897 199	975 2
508 205	586 155	664 207	742 174	820 160	898 1	976 245
509 193	587 155	665 229	743 236	821 155	899 155	977 247
510 1	588 155	666 245	744 249	822 1	900 2	978 3
511 2	589 155	667 155	745 193	823 245	901 245	979 4

980	225	1058	248	1136	227	1214	167	1292	155	1370	242	1448	9
981	229	1059	1	1137	1	1215	226	1293	1	1371	237	1449	245
982	233	1060	2	1138	128	1216	233	1294	231	1372	246	1450	10
983	5	1061	230	1139	236	1217	5	1295	236	1373	9	1451	239
984	242	1062	167	1140	249	1218	6	1296	2	1374	228	1452	11
985	6	1063	3	1141	2	1219	7	1297	238	1375	10	1453	12
986	239	1064	250	1142	243	1220	230	1298	3	1376	239	1454	128
987	7	1065	232	1143	3	1221	237	1299	239	1377	244	1455	249
988	8	1066	4	1144	245	1222	231	1300	245	1378	236	1456	225
989	9	1067	247	1145	4	1223	235	1301	4	1379	243	1457	13
990	238	1068	5	1146	5	1224	8	1302	242	1380	231	1458	228
991	3	1069	245	1147	242	1225	9	1303	5	1381	229	1459	233
992	236	1070	226	1148	6	1226	246	1304	6	1382	11	1460	160
993	174	1071	6	1149	233	1227	240	1305	233	1383	227	1461	14
994	1	1072	235	1150	160	1228	10	1306	7	1384	12	1462	15
995	155	1073	7	1151	7	1229	239	1307	243	1385	13	1463	236
996	2	1074	240	1152	8	1230	11	1308	225	1386	14	1464	229
997	240	1075	8	1153	239	1231	227	1309	8	1387	15	1465	16
998	6	1076	128	1154	244	1232	12	1310	9	1388	16	1466	17
999	233	1077	246	1155	9	1233	13	1311	10	1389	17	1467	18
1000	160	1078	231	1156	10	1234	14	1312	11	1390	18	1468	19
1001	195	1079	9	1157	225	1235	249	1313	229	1391	19	1469	20
1002	239	1080	228	1158	11	1236	15	1314	128	1392	238	1470	10
1003	155	1081	10	1159	232	1237	228	1315	12	1393	20	1471	11
1004	229	1082	160	1160	235	1238	236	1316	232	1394	239	1472	249
1005	1	1083	233	1161	229	1239	16	1317	160	1395	1	1473	155
1006	128	1084	11	1162	12	1240	229	1318	13	1396	155	1474	245
1007	2	1085	227	1163	13	1241	17	1319	14	1397	225	1475	243
1008	3	1086	249	1164	14	1242	244	1320	229	1398	11	1476	1
1009	225	1087	12	1165	15	1243	247	1321	13	1399	12	1477	2
1010	4	1088	13	1166	14	1244	18	1322	226	1400	212	1478	226
1011	5	1089	237	1167	15	1245	19	1323	245	1401	239	1479	237
1012	6	1090	14	1168	174	1246	225	1324	247	1402	230	1480	128
1013	7	1091	15	1169	245	1247	20	1325	155	1403	236	1481	3
1014	198	1092	243	1170	247	1248	21	1326	236	1404	247	1482	240
1015	215	1093	16	1171	1	1249	22	1327	1	1405	225	1483	239
1016	1	1094	17	1172	236	1250	238	1328	249	1406	1	1484	4
1017	155	1095	236	1173	2	1251	243	1329	238	1407	186	1485	160
1018	242	1096	18	1174	228	1252	23	1330	2	1408	2	1486	5
1019	2	1097	244	1175	231	1253	128	1331	3	1409	155	1487	233
1020	3	1098	242	1176	242	1254	24	1332	4	1410	249	1488	6
1021	232	1099	19	1177	3	1255	25	1333	242	1411	3	1489	225
1022	229	1100	238	1178	155	1256	242	1334	5	1412	4	1490	7
1023	225	1101	20	1179	239	1257	26	1335	128	1413	5	1491	8
1024	4	1102	21	1180	4	1258	27	1336	6	1414	243	1492	9
1025	233	1103	22	1181	246	1259	160	1337	160	1415	6	1493	229
1026	239	1104	23	1182	5	1260	28	1338	225	1416	7	1494	24
1027	5	1105	24	1183	6	1261	29	1339	239	1417	8	1495	25
1028	155	1106	10	1184	249	1262	160	1340	7	1418	233	1496	226
1029	155	1107	11	1185	243	1263	11	1341	244	1419	160	1497	234
1030	2	1108	243	1186	7	1264	245	1342	233	1420	9	1498	242
1031	239	1109	155	1187	233	1265	155	1343	8	1421	128	1499	232
1032	225	1110	245	1188	225	1266	1	1344	9	1422	229	1500	236
1033	155	1111	226	1189	8	1267	236	1345	10	1423	10	1501	237
1034	1	1112	1	1190	9	1268	243	1346	11	1424	21	1502	250
1035	229	1113	128	1191	128	1269	242	1347	12	1425	22	1503	155
1036	1	1114	160	1192	10	1270	128	1348	21	1426	167	1504	1
1037	239	1115	2	1193	11	1271	225	1349	22	1427	186	1505	245
1038	155	1116	229	1194	229	1272	2	1350	161	1428	227	1506	2
1039	225	1117	242	1195	12	1273	3	1351	248	1429	247	1507	3
1040	155	1118	233	1196	13	1274	244	1352	233	1430	242	1508	246
1041	155	1119	3	1197	160	1275	233	1353	235	1431	173	1509	4
1042	155	1120	236	1198	30	1276	239	1354	1	1432	226	1510	186
1043	155	1121	4	1199	31	1277	230	1355	128	1433	1	1511	230
1044	155	1122	249	1200	155	1278	4	1356	155	1434	2	1512	5
1045	155	1123	5	1201	161	1279	5	1357	250	1435	155	1513	6
1046	155	1124	239	1202	173	1280	6	1358	226	1436	230	1514	235
1047	155	1125	6	1203	232	1281	7	1359	2	1437	3	1515	239
1048	155	1126	225	1204	234	1282	229	1360	3	1438	237	1516	7
1049	155	1127	7	1205	241	1283	8	1361	4	1439	246	1517	167
1050	155	1128	8	1206	245	1284	9	1362	160	1440	4	1518	249
1051	155	1129	9	1207	250	1285	10	1363	240	1441	235	1519	8
1052	25	1130	16	1208	1	1286	15	1364	5	1442	5	1520	9
1053	26	1131	17	1209	2	1287	16	1365	6	1443	244	1521	10
1054	155	1132	195	1210	3	1288	186	1366	7	1444	6	1522	11
1055	186	1133	204	1211	4	1289	249	1367	225	1445	7	1523	227
1056	229	1134	199	1212	186	1290	167	1368	8	1446	8	1524	12
1057	234	1135	155	1213	248	1291	244	1369	230	1447	243	1525	238

1526	225	1586	14	1646	1	1706	2	1766	8	1826	4	1886	246
1527	13	1587	15	1647	230	1707	3	1767	245	1827	229	1887	161
1528	243	1588	237	1648	2	1708	4	1768	242	1828	243	1888	225
1529	14	1589	230	1649	167	1709	186	1769	9	1829	239	1889	237
1530	233	1590	16	1650	174	1710	5	1770	225	1830	155	1890	1
1531	15	1591	245	1651	231	1711	155	1771	243	1831	1	1891	226
1532	16	1592	17	1652	3	1712	245	1772	10	1832	225	1892	2
1533	244	1593	18	1653	227	1713	6	1773	239	1833	2	1893	3
1534	128	1594	19	1654	245	1714	7	1774	11	1834	3	1894	4
1535	228	1595	20	1655	4	1715	8	1775	12	1835	233	1895	167
1536	229	1596	21	1656	237	1716	9	1776	13	1836	11	1896	5
1537	17	1597	242	1657	5	1717	235	1777	233	1837	12	1897	6
1538	18	1598	22	1658	6	1718	240	1778	128	1838	167	1898	247
1539	231	1599	238	1659	7	1719	10	1779	229	1839	226	1899	7
1540	160	1600	23	1660	235	1720	11	1780	14	1840	236	1900	155
1541	19	1601	24	1661	8	1721	12	1781	160	1841	227	1901	236
1542	20	1602	25	1662	9	1722	225	1782	15	1842	242	1902	8
1543	21	1603	26	1663	238	1723	227	1783	232	1843	1	1903	229
1544	22	1604	14	1664	242	1724	13	1784	16	1844	155	1904	9
1545	23	1605	15	1665	10	1725	232	1785	17	1845	2	1905	10
1546	27	1606	237	1666	228	1726	14	1786	18	1846	3	1906	11
1547	28	1607	167	1667	11	1727	15	1787	19	1847	4	1907	12
1548	174	1608	155	1668	249	1728	239	1788	17	1848	233	1908	13
1549	250	1609	228	1669	236	1729	16	1789	18	1849	239	1909	14
1550	191	1610	1	1670	12	1730	17	1790	235	1850	238	1910	243
1551	1	1611	249	1671	13	1731	243	1791	250	1851	229	1911	15
1552	167	1612	243	1672	244	1732	18	1792	128	1852	225	1912	16
1553	155	1613	242	1673	128	1733	233	1793	230	1853	128	1913	17
1554	2	1614	244	1674	14	1734	19	1794	155	1854	5	1914	128
1555	233	1615	2	1675	239	1735	229	1795	1	1855	160	1915	18
1556	248	1616	232	1676	243	1736	20	1796	160	1856	6	1916	5
1557	249	1617	3	1677	160	1737	21	1797	2	1857	7	1917	6
1558	3	1618	236	1678	225	1738	244	1798	3	1858	8	1918	229
1559	229	1619	240	1679	15	1739	22	1799	233	1859	9	1919	250
1560	232	1620	4	1680	233	1740	23	1800	225	1860	243	1920	160
1561	4	1621	225	1681	16	1741	160	1801	4	1861	10	1921	249
1562	225	1622	233	1682	17	1742	24	1802	228	1862	5	1922	155
1563	235	1623	5	1683	229	1743	128	1803	240	1863	6	1923	1
1564	5	1624	6	1684	18	1744	20	1804	237	1864	155	1924	128
1565	226	1625	128	1685	19	1745	21	1805	226	1865	160	1925	233
1566	6	1626	160	1686	20	1746	186	1806	227	1866	225	1926	2
1567	7	1627	7	1687	21	1747	191	1807	231	1867	229	1927	225
1568	227	1628	8	1688	22	1748	228	1808	236	1868	233	1928	3
1569	8	1629	9	1689	23	1749	247	1809	5	1869	1	1929	4
1570	231	1630	10	1690	25	1750	155	1810	229	1870	128	1930	155
1571	244	1631	229	1691	26	1751	167	1811	6	1871	240	1931	155
1572	9	1632	239	1692	167	1752	1	1812	7	1872	2	1932	155
1573	128	1633	11	1693	172	1753	238	1813	8	1873	244	1933	155
1574	246	1634	12	1694	191	1754	2	1814	9	1874	3	1934	155
1575	240	1635	13	1695	195	1755	3	1815	244	1875	4	1935	155
1576	10	1636	155	1696	200	1756	4	1816	10	1876	160	1936	155
1577	228	1637	245	1697	228	1757	227	1817	11	1877	19	1937	155
1578	11	1638	24	1698	230	1758	226	1818	12	1878	227	1938	155
1579	243	1639	25	1699	237	1759	237	1819	243	1879	173	1939	155
1580	247	1640	186	1700	242	1760	5	1820	238	1880	228		
1581	12	1641	172	1701	174	1761	249	1821	13	1881	233		
1582	13	1642	246	1702	236	1762	6	1822	14	1882	238		
1583	239	1643	155	1703	238	1763	244	1823	242	1883	239		
1584	236	1644	240	1704	249	1764	7	1824	15	1884	240		
1585	160	1645	226	1705	1	1765	236	1825	16	1885	244		

F.3 Standard Compression Type 2 Huffman Encode/Decode Tables

The following encode/decode tables are optimized for English-language program description text. These tables correspond to `multiple_string_structure()` with `compression_type` value `0x02`, and `mode` equal to `0xFF`.

Table F.6 English-language Program Description Encode Table

Prior Symbol: 0	Symbol: 27	Code: 1110000	Prior Symbol: 0	Symbol: 'D'	Code: 11101	Prior Symbol: 0	Symbol: 'I'	Code: 011000
Prior Symbol: 0	Symbol: ""	Code: 111001	Prior Symbol: 0	Symbol: 'E'	Code: 10010	Prior Symbol: 0	Symbol: 'J'	Code: 1100
Prior Symbol: 0	Symbol: 'A'	Code: 010	Prior Symbol: 0	Symbol: 'F'	Code: 10110	Prior Symbol: 0	Symbol: 'K'	Code: 00101
Prior Symbol: 0	Symbol: 'B'	Code: 0011	Prior Symbol: 0	Symbol: 'G'	Code: 011011	Prior Symbol: 0	Symbol: 'L'	Code: 10011
Prior Symbol: 0	Symbol: 'C'	Code: 0111	Prior Symbol: 0	Symbol: 'H'	Code: 10111	Prior Symbol: 0	Symbol: 'M'	Code: 1111

Prior Symbol: 0 Symbol: 'N' Code: 00100	Prior Symbol: '' Symbol: 'e' Code: 100010	Prior Symbol: '4' Symbol: 27 Code: 10
Prior Symbol: 0 Symbol: 'O' Code: 011001	Prior Symbol: '' Symbol: 'f' Code: 11101	Prior Symbol: '4' Symbol: '' Code: 11
Prior Symbol: 0 Symbol: 'P' Code: 000	Prior Symbol: '' Symbol: 'g' Code: 100011	Prior Symbol: '4' Symbol: '' Code: 0
Prior Symbol: 0 Symbol: 'R' Code: 1000	Prior Symbol: '' Symbol: 'h' Code: 0001	Prior Symbol: '5' Symbol: 27 Code: 11
Prior Symbol: 0 Symbol: 'S' Code: 1010	Prior Symbol: '' Symbol: 'i' Code: 10101	Prior Symbol: '5' Symbol: '' Code: 10
Prior Symbol: 0 Symbol: 'T' Code: 1101	Prior Symbol: '' Symbol: 'j' Code: 11001111	Prior Symbol: '5' Symbol: '' Code: 0
Prior Symbol: 0 Symbol: 'V' Code: 1110001	Prior Symbol: '' Symbol: 'k' Code: 11111010	Prior Symbol: '6' Symbol: 27 Code: 1
Prior Symbol: 0 Symbol: 'W' Code: 011010	Prior Symbol: '' Symbol: 'l' Code: 010111	Prior Symbol: '7' Symbol: 27 Code: 0
Prior Symbol: 1 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'm' Code: 00000	Prior Symbol: '7' Symbol: '' Code: 10
Prior Symbol: 2 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'n' Code: 1010001	Prior Symbol: '7' Symbol: '' Code: 11
Prior Symbol: 3 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'o' Code: 0010	Prior Symbol: '8' Symbol: 27 Code: 1
Prior Symbol: 4 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'p' Code: 10110	Prior Symbol: '8' Symbol: '' Code: 110
Prior Symbol: 5 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'q' Code: 110010101	Prior Symbol: '9' Symbol: '' Code: 111
Prior Symbol: 6 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'r' Code: 00111	Prior Symbol: '9' Symbol: '5' Code: 00
Prior Symbol: 7 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 's' Code: 11100	Prior Symbol: '9' Symbol: '6' Code: 01
Prior Symbol: 8 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 't' Code: 1101	Prior Symbol: '9' Symbol: '8' Code: 10
Prior Symbol: 9 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'u' Code: 11111011	Prior Symbol: '' Symbol: 27 Code: 0
Prior Symbol: 10 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'v' Code: 11111100	Prior Symbol: '' Symbol: '' Code: 1
Prior Symbol: 11 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'w' Code: 11000	Prior Symbol: '' Symbol: 27 Code: 0
Prior Symbol: 12 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'y' Code: 11001110	Prior Symbol: '' Symbol: '' Code: 1
Prior Symbol: 13 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 27 Code: 1	Prior Symbol: '<' Symbol: 27 Code: 1
Prior Symbol: 14 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 0 Code: 000	Prior Symbol: '=' Symbol: 27 Code: 1
Prior Symbol: 15 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 27 Code: 10	Prior Symbol: '>' Symbol: 27 Code: 1
Prior Symbol: 16 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '' Code: 11	Prior Symbol: '?' Symbol: 27 Code: 0
Prior Symbol: 17 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '' Code: 001	Prior Symbol: '?' Symbol: '' Code: 1
Prior Symbol: 18 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'H' Code: 010	Prior Symbol: '@' Symbol: 27 Code: 1
Prior Symbol: 19 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 'T' Code: 011	Prior Symbol: 'A' Symbol: 27 Code: 10010
Prior Symbol: 20 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '#' Symbol: 27 Code: 1	Prior Symbol: 'A' Symbol: '' Code: 11
Prior Symbol: 21 Symbol: 27 Code: 1	Prior Symbol: 'S' Symbol: 27 Code: 1	Prior Symbol: 'A' Symbol: 'd' Code: 10011
Prior Symbol: 22 Symbol: 27 Code: 1	Prior Symbol: '%>' Symbol: 27 Code: 1	Prior Symbol: 'A' Symbol: 'l' Code: 101000
Prior Symbol: 23 Symbol: 27 Code: 1	Prior Symbol: '&' Symbol: 27 Code: 1	Prior Symbol: 'A' Symbol: 'l' Code: 00
Prior Symbol: 24 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 27 Code: 00	Prior Symbol: 'A' Symbol: 'm' Code: 10101
Prior Symbol: 25 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '' Code: 010	Prior Symbol: 'A' Symbol: 'n' Code: 01
Prior Symbol: 26 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: 's' Code: 1	Prior Symbol: 'A' Symbol: 'r' Code: 1011
Prior Symbol: 27 Symbol: 27 Code: 1	Prior Symbol: '' Symbol: '' Code: 011	Prior Symbol: 'A' Symbol: 's' Code: 10000
Prior Symbol: 28 Symbol: 27 Code: 1	Prior Symbol: 'l' Symbol: 27 Code: 1	Prior Symbol: 'A' Symbol: 't' Code: 10001
Prior Symbol: 29 Symbol: 27 Code: 1	Prior Symbol: 'y' Symbol: 27 Code: 1	Prior Symbol: 'A' Symbol: 'u' Code: 101001
Prior Symbol: 30 Symbol: 27 Code: 1	Prior Symbol: 'j' Symbol: '' Code: 0	Prior Symbol: 'A' Symbol: 'v' Code: 10010
Prior Symbol: 31 Symbol: 27 Code: 1	Prior Symbol: '*' Symbol: 27 Code: 1	Prior Symbol: 'B' Symbol: 'a' Code: 101
Prior Symbol: '' Symbol: 27 Code: 101000001	Prior Symbol: '+' Symbol: 27 Code: 1	Prior Symbol: 'B' Symbol: 'e' Code: 111
Prior Symbol: '' Symbol: '' Code: 111111010	Prior Symbol: '' Symbol: '' Code: 00	Prior Symbol: 'B' Symbol: 'f' Code: 00
Prior Symbol: '' Symbol: 'l' Code: 1111111100	Prior Symbol: '' Symbol: '' Code: 1	Prior Symbol: 'B' Symbol: 'l' Code: 10011
Prior Symbol: '' Symbol: '' Code: 11111111110	Prior Symbol: '' Symbol: '' Code: 01	Prior Symbol: 'B' Symbol: 'o' Code: 110
Prior Symbol: '' Symbol: 'l' Code: 11111111111	Prior Symbol: '' Symbol: 27 Code: 10	Prior Symbol: 'B' Symbol: 'r' Code: 01
Prior Symbol: '' Symbol: 'l' Code: 0101011	Prior Symbol: '' Symbol: '' Code: 1110	Prior Symbol: 'B' Symbol: 'u' Code: 1000
Prior Symbol: '' Symbol: '2' Code: 0100010	Prior Symbol: '' Symbol: 'a' Code: 000	Prior Symbol: 'C' Symbol: 27 Code: 01110
Prior Symbol: '' Symbol: '3' Code: 1111111101	Prior Symbol: '' Symbol: 'b' Code: 0010	Prior Symbol: 'C' Symbol: 'l' Code: 00
Prior Symbol: '' Symbol: '4' Code: 110010100	Prior Symbol: '' Symbol: 'c' Code: 110	Prior Symbol: 'C' Symbol: 'h' Code: 10
Prior Symbol: '' Symbol: '5' Code: 1111111110	Prior Symbol: '' Symbol: 'd' Code: 0011	Prior Symbol: 'C' Symbol: 'l' Code: 01111
Prior Symbol: '' Symbol: '7' Code: 1010000000	Prior Symbol: '' Symbol: 'e' Code: 0100	Prior Symbol: 'C' Symbol: 'e' Code: 111
Prior Symbol: '' Symbol: 'A' Code: 10010	Prior Symbol: '' Symbol: 'f' Code: 0101	Prior Symbol: 'C' Symbol: 'r' Code: 0101
Prior Symbol: '' Symbol: 'B' Code: 010100	Prior Symbol: '' Symbol: 'r' Code: 1111	Prior Symbol: 'C' Symbol: 'r' Code: 0110
Prior Symbol: '' Symbol: 'C' Code: 111100	Prior Symbol: '' Symbol: 's' Code: 011	Prior Symbol: 'C' Symbol: 'u' Code: 0110
Prior Symbol: '' Symbol: 'D' Code: 1111010	Prior Symbol: '' Symbol: 0 Code: 1	Prior Symbol: 'C' Symbol: 'y' Code: 0100
Prior Symbol: '' Symbol: 'E' Code: 0100011	Prior Symbol: '' Symbol: 27 Code: 000	Prior Symbol: 'D' Symbol: 27 Code: 1111
Prior Symbol: '' Symbol: 'F' Code: 0101010	Prior Symbol: '' Symbol: '' Code: 01	Prior Symbol: 'D' Symbol: 'a' Code: 01
Prior Symbol: '' Symbol: 'G' Code: 000010	Prior Symbol: '' Symbol: '' Code: 0010	Prior Symbol: 'D' Symbol: 'e' Code: 100
Prior Symbol: '' Symbol: 'H' Code: 1111011	Prior Symbol: '' Symbol: 'J' Code: 00110	Prior Symbol: 'D' Symbol: 'l' Code: 00
Prior Symbol: '' Symbol: 'I' Code: 11001011	Prior Symbol: '' Symbol: 'S' Code: 00111	Prior Symbol: 'D' Symbol: 'o' Code: 101
Prior Symbol: '' Symbol: 'J' Code: 000011	Prior Symbol: '' Symbol: 27 Code: 0	Prior Symbol: 'D' Symbol: 'u' Code: 1110
Prior Symbol: '' Symbol: 'K' Code: 1100100	Prior Symbol: '' Symbol: '' Code: 1	Prior Symbol: 'D' Symbol: 'y' Code: 1100
Prior Symbol: '' Symbol: 'L' Code: 010110	Prior Symbol: '0' Symbol: 27 Code: 100	Prior Symbol: 'D' Symbol: '' Code: 10
Prior Symbol: '' Symbol: 'M' Code: 101001	Prior Symbol: '' Symbol: '' Code: 111	Prior Symbol: 'E' Symbol: 'a' Code: 0110
Prior Symbol: '' Symbol: 'N' Code: 001100	Prior Symbol: '0' Symbol: '0' Code: 00	Prior Symbol: 'E' Symbol: 'd' Code: 000
Prior Symbol: '' Symbol: 'O' Code: 10100001	Prior Symbol: '0' Symbol: '7' Code: 101	Prior Symbol: 'E' Symbol: 'l' Code: 0111
Prior Symbol: '' Symbol: 'P' Code: 001101	Prior Symbol: '0' Symbol: 's' Code: 01	Prior Symbol: 'E' Symbol: 'l' Code: 001
Prior Symbol: '' Symbol: 'R' Code: 1111100	Prior Symbol: '0' Symbol: 't' Code: 110	Prior Symbol: 'E' Symbol: 'n' Code: 1100
Prior Symbol: '' Symbol: 'S' Code: 01001	Prior Symbol: '1' Symbol: 27 Code: 111	Prior Symbol: 'E' Symbol: 'r' Code: 111
Prior Symbol: '' Symbol: 'T' Code: 1100110	Prior Symbol: '' Symbol: '' Code: 10	Prior Symbol: 'E' Symbol: 's' Code: 010
Prior Symbol: '' Symbol: 'U' Code: 111111011	Prior Symbol: '1' Symbol: '8' Code: 110	Prior Symbol: 'E' Symbol: 'v' Code: 1101
Prior Symbol: '' Symbol: 'V' Code: 111111100	Prior Symbol: '1' Symbol: '9' Code: 0	Prior Symbol: 'E' Symbol: '' Code: 1101
Prior Symbol: '' Symbol: 'W' Code: 010000	Prior Symbol: '2' Symbol: 27 Code: 101	Prior Symbol: 'F' Symbol: 27 Code: 00
Prior Symbol: '' Symbol: 'Y' Code: 111111101	Prior Symbol: '2' Symbol: '' Code: 11	Prior Symbol: 'F' Symbol: 'e' Code: 100
Prior Symbol: '' Symbol: 'Z' Code: 1010000001	Prior Symbol: '2' Symbol: '' Code: 0	Prior Symbol: 'F' Symbol: 'l' Code: 101
Prior Symbol: '' Symbol: 'a' Code: 011	Prior Symbol: '2' Symbol: '6' Code: 100	Prior Symbol: 'F' Symbol: 'o' Code: 01
Prior Symbol: '' Symbol: 'b' Code: 10111	Prior Symbol: '3' Symbol: 27 Code: 10	Prior Symbol: 'F' Symbol: 'r' Code: 11
Prior Symbol: '' Symbol: 'c' Code: 10011	Prior Symbol: '3' Symbol: '' Code: 0	Prior Symbol: 'G' Symbol: 27 Code: 000
Prior Symbol: '' Symbol: 'd' Code: 10000	Prior Symbol: '3' Symbol: '0' Code: 11	Prior Symbol: 'G' Symbol: 'a' Code: 110

Prior Symbol: 'G' Symbol: 'e' Code: 01
Prior Symbol: 'G' Symbol: 'l' Code: 100
Prior Symbol: 'G' Symbol: 't' Code: 001
Prior Symbol: 'G' Symbol: 'o' Code: 1011
Prior Symbol: 'G' Symbol: 'r' Code: 111
Prior Symbol: 'G' Symbol: 'u' Code: 1010
Prior Symbol: 'H' Symbol: 27 Code: 010
Prior Symbol: 'H' Symbol: 'a' Code: 00
Prior Symbol: 'H' Symbol: 'e' Code: 011
Prior Symbol: 'H' Symbol: 'i' Code: 110
Prior Symbol: 'H' Symbol: 'o' Code: 10
Prior Symbol: 'H' Symbol: 'u' Code: 111
Prior Symbol: 'T' Symbol: 27 Code: 011
Prior Symbol: 'T' Symbol: ' ' Code: 000
Prior Symbol: 'T' Symbol: 'l' Code: 100
Prior Symbol: 'T' Symbol: 't' Code: 001
Prior Symbol: 'T' Symbol: 'n' Code: 11
Prior Symbol: 'T' Symbol: 'r' Code: 101
Prior Symbol: 'T' Symbol: 's' Code: 010
Prior Symbol: 'J' Symbol: 27 Code: 1000
Prior Symbol: 'J' Symbol: ' ' Code: 1001
Prior Symbol: 'J' Symbol: 'a' Code: 111
Prior Symbol: 'J' Symbol: 'e' Code: 1101
Prior Symbol: 'J' Symbol: 'i' Code: 1100
Prior Symbol: 'J' Symbol: 'l' Code: 0
Prior Symbol: 'J' Symbol: 'u' Code: 101
Prior Symbol: 'K' Symbol: 27 Code: 111
Prior Symbol: 'K' Symbol: 'a' Code: 100
Prior Symbol: 'K' Symbol: 'e' Code: 0
Prior Symbol: 'K' Symbol: 'i' Code: 101
Prior Symbol: 'K' Symbol: 'l' Code: 110
Prior Symbol: 'L' Symbol: 27 Code: 0110
Prior Symbol: 'L' Symbol: 'a' Code: 11
Prior Symbol: 'L' Symbol: 'e' Code: 00
Prior Symbol: 'L' Symbol: 'l' Code: 0111
Prior Symbol: 'L' Symbol: 'o' Code: 10
Prior Symbol: 'L' Symbol: 'u' Code: 010
Prior Symbol: 'M' Symbol: 27 Code: 110110
Prior Symbol: 'M' Symbol: 'a' Code: 0
Prior Symbol: 'M' Symbol: 'c' Code: 11011
Prior Symbol: 'M' Symbol: 'e' Code: 1111
Prior Symbol: 'M' Symbol: 'i' Code: 10
Prior Symbol: 'M' Symbol: 'o' Code: 1100
Prior Symbol: 'M' Symbol: 'u' Code: 1110
Prior Symbol: 'N' Symbol: 27 Code: 1100
Prior Symbol: 'N' Symbol: 'a' Code: 111
Prior Symbol: 'N' Symbol: 'e' Code: 0
Prior Symbol: 'N' Symbol: 'i' Code: 1101
Prior Symbol: 'N' Symbol: 'o' Code: 10
Prior Symbol: 'O' Symbol: 27 Code: 10
Prior Symbol: 'O' Symbol: ' ' Code: 010
Prior Symbol: 'O' Symbol: 'l' Code: 110
Prior Symbol: 'O' Symbol: 'n' Code: 011
Prior Symbol: 'O' Symbol: 'r' Code: 111
Prior Symbol: 'O' Symbol: 's' Code: 00
Prior Symbol: 'P' Symbol: 27 Code: 10010
Prior Symbol: 'P' Symbol: 'a' Code: 0
Prior Symbol: 'P' Symbol: 'e' Code: 111
Prior Symbol: 'P' Symbol: 'h' Code: 10011
Prior Symbol: 'P' Symbol: 'i' Code: 1000
Prior Symbol: 'P' Symbol: 'l' Code: 1101
Prior Symbol: 'P' Symbol: 'o' Code: 101
Prior Symbol: 'P' Symbol: 'r' Code: 1100
Prior Symbol: 'Q' Symbol: 27 Code: 1
Prior Symbol: 'R' Symbol: 27 Code: 0000
Prior Symbol: 'R' Symbol: ' ' Code: 0001
Prior Symbol: 'R' Symbol: 'a' Code: 01
Prior Symbol: 'R' Symbol: 'e' Code: 10
Prior Symbol: 'R' Symbol: 'i' Code: 001
Prior Symbol: 'R' Symbol: 'o' Code: 11
Prior Symbol: 'S' Symbol: 27 Code: 1011
Prior Symbol: 'S' Symbol: ' ' Code: 0001
Prior Symbol: 'S' Symbol: 'a' Code: 100
Prior Symbol: 'S' Symbol: 'c' Code: 0010
Prior Symbol: 'S' Symbol: 'e' Code: 1110
Prior Symbol: 'S' Symbol: 'h' Code: 110
Prior Symbol: 'S' Symbol: 'i' Code: 0011
Prior Symbol: 'S' Symbol: 'o' Code: 1111

Prior Symbol: 'S' Symbol: 't' Code: 01
Prior Symbol: 'S' Symbol: 'u' Code: 1010
Prior Symbol: 'S' Symbol: 'v' Code: 00000
Prior Symbol: 'S' Symbol: 'y' Code: 00001
Prior Symbol: 'T' Symbol: 27 Code: 1010
Prior Symbol: 'T' Symbol: 'V' Code: 1000
Prior Symbol: 'T' Symbol: 'a' Code: 1001
Prior Symbol: 'T' Symbol: 'e' Code: 11010
Prior Symbol: 'T' Symbol: 'h' Code: 0
Prior Symbol: 'T' Symbol: 'i' Code: 1011
Prior Symbol: 'T' Symbol: 'o' Code: 111
Prior Symbol: 'T' Symbol: 'r' Code: 1100
Prior Symbol: 'T' Symbol: 'w' Code: 11011
Prior Symbol: 'U' Symbol: 27 Code: 10
Prior Symbol: 'U' Symbol: ' ' Code: 0
Prior Symbol: 'U' Symbol: 'n' Code: 11
Prior Symbol: 'V' Symbol: 27 Code: 111
Prior Symbol: 'V' Symbol: ' ' Code: 10
Prior Symbol: 'V' Symbol: 'e' Code: 110
Prior Symbol: 'V' Symbol: 'i' Code: 0
Prior Symbol: 'W' Symbol: 27 Code: 010
Prior Symbol: 'W' Symbol: 'a' Code: 111
Prior Symbol: 'W' Symbol: 'e' Code: 110
Prior Symbol: 'W' Symbol: 'h' Code: 011
Prior Symbol: 'W' Symbol: 'l' Code: 10
Prior Symbol: 'W' Symbol: 'o' Code: 00
Prior Symbol: 'X' Symbol: 27 Code: 1
Prior Symbol: 'Y' Symbol: 27 Code: 0
Prior Symbol: 'Y' Symbol: 'o' Code: 1
Prior Symbol: 'Z' Symbol: 27 Code: 1
Prior Symbol: 'J' Symbol: 27 Code: 1
Prior Symbol: 'V' Symbol: 27 Code: 1
Prior Symbol: 'J' Symbol: 27 Code: 1
Prior Symbol: 'M' Symbol: 27 Code: 1
Prior Symbol: 'L' Symbol: 27 Code: 1
Prior Symbol: ' ' Symbol: 27 Code: 1
Prior Symbol: ' ' Symbol: 27 Code: 1
Prior Symbol: 'a' Symbol: 27 Code: 111001101
Prior Symbol: 'a' Symbol: ' ' Code: 101
Prior Symbol: 'a' Symbol: ' ' Code: 111001110
Prior Symbol: 'a' Symbol: ' ' Code: 1110010
Prior Symbol: 'a' Symbol: 'b' Code: 001011
Prior Symbol: 'a' Symbol: 'c' Code: 11001
Prior Symbol: 'a' Symbol: 'd' Code: 00111
Prior Symbol: 'a' Symbol: 'e' Code: 0011001
Prior Symbol: 'a' Symbol: 'f' Code: 001010
Prior Symbol: 'a' Symbol: 'g' Code: 00100
Prior Symbol: 'a' Symbol: 'h' Code: 001100010
Prior Symbol: 'a' Symbol: 'i' Code: 111000
Prior Symbol: 'a' Symbol: 'k' Code: 110000
Prior Symbol: 'a' Symbol: 'l' Code: 1101
Prior Symbol: 'a' Symbol: 'm' Code: 11101
Prior Symbol: 'a' Symbol: 'n' Code: 01
Prior Symbol: 'a' Symbol: 'o' Code: 001100011
Prior Symbol: 'a' Symbol: 'p' Code: 00000
Prior Symbol: 'a' Symbol: 'r' Code: 100
Prior Symbol: 'a' Symbol: 's' Code: 0001
Prior Symbol: 'a' Symbol: 't' Code: 1111
Prior Symbol: 'a' Symbol: 'u' Code: 110001
Prior Symbol: 'a' Symbol: 'v' Code: 001101
Prior Symbol: 'a' Symbol: 'w' Code: 111001111
Prior Symbol: 'a' Symbol: 'x' Code: 111001100
Prior Symbol: 'a' Symbol: 'y' Code: 00001
Prior Symbol: 'a' Symbol: 'z' Code: 00110000
Prior Symbol: 'b' Symbol: 27 Code: 101000
Prior Symbol: 'b' Symbol: ' ' Code: 0101
Prior Symbol: 'b' Symbol: ' ' Code: 101001
Prior Symbol: 'b' Symbol: 'a' Code: 100
Prior Symbol: 'b' Symbol: 'b' Code: 101010
Prior Symbol: 'b' Symbol: 'd' Code: 1010110
Prior Symbol: 'b' Symbol: 'e' Code: 00
Prior Symbol: 'b' Symbol: 'i' Code: 1011
Prior Symbol: 'b' Symbol: 'l' Code: 0100
Prior Symbol: 'b' Symbol: 'o' Code: 110
Prior Symbol: 'b' Symbol: 'r' Code: 1110
Prior Symbol: 'b' Symbol: 's' Code: 1010111
Prior Symbol: 'b' Symbol: 'u' Code: 1111
Prior Symbol: 'b' Symbol: 'y' Code: 011
Prior Symbol: 'c' Symbol: 27 Code: 00010

Prior Symbol: 'c' Symbol: ' ' Code: 10000
Prior Symbol: 'c' Symbol: ' ' Code: 010000
Prior Symbol: 'c' Symbol: ' ' Code: 0100011
Prior Symbol: 'c' Symbol: 'D' Code: 0100110
Prior Symbol: 'c' Symbol: 'a' Code: 110
Prior Symbol: 'c' Symbol: 'c' Code: 010010
Prior Symbol: 'c' Symbol: 'e' Code: 011
Prior Symbol: 'c' Symbol: 'h' Code: 111
Prior Symbol: 'c' Symbol: 'i' Code: 0101
Prior Symbol: 'c' Symbol: 'k' Code: 1001
Prior Symbol: 'c' Symbol: 'l' Code: 10001
Prior Symbol: 'c' Symbol: 'o' Code: 1001
Prior Symbol: 'c' Symbol: 'q' Code: 0100010
Prior Symbol: 'c' Symbol: 'r' Code: 00011
Prior Symbol: 'c' Symbol: 's' Code: 001
Prior Symbol: 'c' Symbol: 'u' Code: 0000
Prior Symbol: 'c' Symbol: 'y' Code: 0100111
Prior Symbol: 'd' Symbol: 27 Code: 1010001
Prior Symbol: 'd' Symbol: ' ' Code: 11
Prior Symbol: 'd' Symbol: ' ' Code: 01111010
Prior Symbol: 'd' Symbol: ' ' Code: 101011
Prior Symbol: 'd' Symbol: ' ' Code: 0100
Prior Symbol: 'd' Symbol: ' ' Code: 01111011
Prior Symbol: 'd' Symbol: 'a' Code: 1000
Prior Symbol: 'd' Symbol: 'b' Code: 01010
Prior Symbol: 'd' Symbol: 'e' Code: 00
Prior Symbol: 'd' Symbol: 'f' Code: 10100000
Prior Symbol: 'd' Symbol: 'g' Code: 10101011
Prior Symbol: 'd' Symbol: 'i' Code: 1011
Prior Symbol: 'd' Symbol: 'l' Code: 011111
Prior Symbol: 'd' Symbol: 'm' Code: 10100001
Prior Symbol: 'd' Symbol: 'n' Code: 1010100
Prior Symbol: 'd' Symbol: 'o' Code: 0110
Prior Symbol: 'd' Symbol: 'r' Code: 01110
Prior Symbol: 'd' Symbol: 's' Code: 1001
Prior Symbol: 'd' Symbol: 'u' Code: 101001
Prior Symbol: 'd' Symbol: 'v' Code: 0111100
Prior Symbol: 'd' Symbol: 'w' Code: 10101010
Prior Symbol: 'd' Symbol: 'y' Code: 01011
Prior Symbol: 'e' Symbol: 27 Code: 101110011
Prior Symbol: 'e' Symbol: ' ' Code: 111
Prior Symbol: 'e' Symbol: ' ' Code: 10111010
Prior Symbol: 'e' Symbol: ' ' Code: 100110000
Prior Symbol: 'e' Symbol: ' ' Code: 000111
Prior Symbol: 'e' Symbol: ' ' Code: 10011001
Prior Symbol: 'e' Symbol: ' ' Code: 00110
Prior Symbol: 'e' Symbol: ' ' Code: 10011010
Prior Symbol: 'e' Symbol: 'a' Code: 1000
Prior Symbol: 'e' Symbol: 'b' Code: 0001100
Prior Symbol: 'e' Symbol: 'c' Code: 10010
Prior Symbol: 'e' Symbol: 'd' Code: 0000
Prior Symbol: 'e' Symbol: 'e' Code: 10100
Prior Symbol: 'e' Symbol: 'f' Code: 10111011
Prior Symbol: 'e' Symbol: 'g' Code: 0001101
Prior Symbol: 'e' Symbol: 'h' Code: 100110001
Prior Symbol: 'e' Symbol: 'i' Code: 000100
Prior Symbol: 'e' Symbol: 'k' Code: 10011011
Prior Symbol: 'e' Symbol: 'l' Code: 0010
Prior Symbol: 'e' Symbol: 'm' Code: 100111
Prior Symbol: 'e' Symbol: 'n' Code: 010
Prior Symbol: 'e' Symbol: 'o' Code: 001110
Prior Symbol: 'e' Symbol: 'p' Code: 001111
Prior Symbol: 'e' Symbol: 'r' Code: 110
Prior Symbol: 'e' Symbol: 's' Code: 011
Prior Symbol: 'e' Symbol: 't' Code: 10101
Prior Symbol: 'e' Symbol: 'u' Code: 101110010
Prior Symbol: 'e' Symbol: 'v' Code: 101100
Prior Symbol: 'e' Symbol: 'w' Code: 101111
Prior Symbol: 'e' Symbol: 'x' Code: 000101
Prior Symbol: 'e' Symbol: 'y' Code: 101101
Prior Symbol: 'e' Symbol: 'z' Code: 10111000
Prior Symbol: 'f' Symbol: 27 Code: 1110111
Prior Symbol: 'f' Symbol: ' ' Code: 10
Prior Symbol: 'f' Symbol: ' ' Code: 1110110
Prior Symbol: 'f' Symbol: 'a' Code: 1111
Prior Symbol: 'f' Symbol: 'b' Code: 000
Prior Symbol: 'f' Symbol: 'f' Code: 0101
Prior Symbol: 'f' Symbol: 'i' Code: 001

Prior Symbol: 'f' Symbol: 'l' Code: 111010
Prior Symbol: 'f' Symbol: 'o' Code: 110
Prior Symbol: 'f' Symbol: 'r' Code: 011
Prior Symbol: 'f' Symbol: 't' Code: 0100
Prior Symbol: 'f' Symbol: 'u' Code: 11100
Prior Symbol: 'g' Symbol: '27' Code: 1111010
Prior Symbol: 'g' Symbol: ' ' Code: 10
Prior Symbol: 'g' Symbol: ' ' Code: 1111011
Prior Symbol: 'g' Symbol: ' ' Code: 111110
Prior Symbol: 'g' Symbol: ' ' Code: 0101010
Prior Symbol: 'g' Symbol: ' ' Code: 01011
Prior Symbol: 'g' Symbol: 'a' Code: 1110
Prior Symbol: 'g' Symbol: 'e' Code: 00
Prior Symbol: 'g' Symbol: 'g' Code: 0101011
Prior Symbol: 'g' Symbol: 'g' Code: 011
Prior Symbol: 'g' Symbol: 'i' Code: 1101
Prior Symbol: 'g' Symbol: 'l' Code: 111100
Prior Symbol: 'g' Symbol: 'o' Code: 0100
Prior Symbol: 'g' Symbol: 'u' Code: 111111
Prior Symbol: 'g' Symbol: 's' Code: 11000
Prior Symbol: 'g' Symbol: 'u' Code: 11001
Prior Symbol: 'g' Symbol: 'u' Code: 010100
Prior Symbol: 'h' Symbol: '27' Code: 1011100
Prior Symbol: 'h' Symbol: ' ' Code: 100
Prior Symbol: 'h' Symbol: ' ' Code: 10101000
Prior Symbol: 'h' Symbol: ' ' Code: 10101001
Prior Symbol: 'h' Symbol: ' ' Code: 10101011
Prior Symbol: 'h' Symbol: ' ' Code: 101001
Prior Symbol: 'h' Symbol: 'a' Code: 011
Prior Symbol: 'h' Symbol: 'e' Code: 11
Prior Symbol: 'h' Symbol: 'i' Code: 00
Prior Symbol: 'h' Symbol: 'n' Code: 101011
Prior Symbol: 'h' Symbol: 'o' Code: 010
Prior Symbol: 'h' Symbol: ' ' Code: 101111
Prior Symbol: 'h' Symbol: 's' Code: 10101010
Prior Symbol: 'h' Symbol: 't' Code: 10110
Prior Symbol: 'h' Symbol: 'u' Code: 101000
Prior Symbol: 'h' Symbol: 'y' Code: 1011101
Prior Symbol: 'i' Symbol: '27' Code: 00011101
Prior Symbol: 'i' Symbol: ' ' Code: 0001111
Prior Symbol: 'i' Symbol: ' ' Code: 100110100
Prior Symbol: 'i' Symbol: ' ' Code: 10011000
Prior Symbol: 'i' Symbol: 'a' Code: 11010
Prior Symbol: 'i' Symbol: 'b' Code: 100110101
Prior Symbol: 'i' Symbol: 'c' Code: 1111
Prior Symbol: 'i' Symbol: 'd' Code: 10000
Prior Symbol: 'i' Symbol: 'e' Code: 1110
Prior Symbol: 'i' Symbol: 'f' Code: 100111
Prior Symbol: 'i' Symbol: 'g' Code: 10010
Prior Symbol: 'i' Symbol: 'h' Code: 10011011
Prior Symbol: 'i' Symbol: 'l' Code: 1100
Prior Symbol: 'i' Symbol: 'm' Code: 10001
Prior Symbol: 'i' Symbol: 'n' Code: 01
Prior Symbol: 'i' Symbol: 'o' Code: 11011
Prior Symbol: 'i' Symbol: 'p' Code: 000110
Prior Symbol: 'i' Symbol: 'r' Code: 0000
Prior Symbol: 'i' Symbol: 's' Code: 101
Prior Symbol: 'i' Symbol: 't' Code: 001
Prior Symbol: 'i' Symbol: 'v' Code: 00010
Prior Symbol: 'i' Symbol: 'x' Code: 00011100
Prior Symbol: 'i' Symbol: 'z' Code: 10011001
Prior Symbol: 'j' Symbol: '27' Code: 000
Prior Symbol: 'j' Symbol: 'a' Code: 001
Prior Symbol: 'j' Symbol: 'e' Code: 010
Prior Symbol: 'j' Symbol: 'o' Code: 1
Prior Symbol: 'j' Symbol: 'u' Code: 011
Prior Symbol: 'k' Symbol: '27' Code: 0000
Prior Symbol: 'k' Symbol: ' ' Code: 01
Prior Symbol: 'k' Symbol: ' ' Code: 10000
Prior Symbol: 'k' Symbol: ' ' Code: 10011
Prior Symbol: 'k' Symbol: ' ' Code: 0001
Prior Symbol: 'k' Symbol: 'e' Code: 11
Prior Symbol: 'k' Symbol: 'l' Code: 101
Prior Symbol: 'k' Symbol: 'l' Code: 100100
Prior Symbol: 'k' Symbol: 'n' Code: 10001
Prior Symbol: 'k' Symbol: 's' Code: 001
Prior Symbol: 'k' Symbol: 'y' Code: 100101
Prior Symbol: 'l' Symbol: '27' Code: 0011100

Prior Symbol: 'l' Symbol: ' ' Code: 110
Prior Symbol: 'l' Symbol: ' ' Code: 00111100
Prior Symbol: 'l' Symbol: ' ' Code: 001101
Prior Symbol: 'l' Symbol: ' ' Code: 00111101
Prior Symbol: 'l' Symbol: ' ' Code: 00100
Prior Symbol: 'l' Symbol: 'a' Code: 000
Prior Symbol: 'l' Symbol: 'b' Code: 0011101
Prior Symbol: 'l' Symbol: 'c' Code: 00111111
Prior Symbol: 'l' Symbol: 'd' Code: 10111
Prior Symbol: 'l' Symbol: 'e' Code: 111
Prior Symbol: 'l' Symbol: 'f' Code: 010110
Prior Symbol: 'l' Symbol: 'i' Code: 011
Prior Symbol: 'l' Symbol: 'k' Code: 10110110
Prior Symbol: 'l' Symbol: 'l' Code: 100
Prior Symbol: 'l' Symbol: 'm' Code: 010111
Prior Symbol: 'l' Symbol: 'n' Code: 00111110
Prior Symbol: 'l' Symbol: 'o' Code: 1010
Prior Symbol: 'l' Symbol: 'p' Code: 00101
Prior Symbol: 'l' Symbol: 'r' Code: 10110111
Prior Symbol: 'l' Symbol: 's' Code: 01010
Prior Symbol: 'l' Symbol: 't' Code: 001100
Prior Symbol: 'l' Symbol: 'u' Code: 1011010
Prior Symbol: 'l' Symbol: 'v' Code: 101100
Prior Symbol: 'l' Symbol: 'y' Code: 0100
Prior Symbol: 'm' Symbol: '27' Code: 101010
Prior Symbol: 'm' Symbol: ' ' Code: 111
Prior Symbol: 'm' Symbol: ' ' Code: 1010110
Prior Symbol: 'm' Symbol: ' ' Code: 110101
Prior Symbol: 'm' Symbol: ' ' Code: 1010111
Prior Symbol: 'm' Symbol: 'a' Code: 00
Prior Symbol: 'm' Symbol: 'b' Code: 10100
Prior Symbol: 'm' Symbol: 'e' Code: 01
Prior Symbol: 'm' Symbol: 'f' Code: 1100
Prior Symbol: 'm' Symbol: 'm' Code: 10110
Prior Symbol: 'm' Symbol: 'o' Code: 1000
Prior Symbol: 'm' Symbol: 'p' Code: 1001
Prior Symbol: 'm' Symbol: 's' Code: 10111
Prior Symbol: 'm' Symbol: 'u' Code: 10111
Prior Symbol: 'm' Symbol: 'y' Code: 110100
Prior Symbol: 'n' Symbol: '27' Code: 0100000
Prior Symbol: 'n' Symbol: ' ' Code: 10
Prior Symbol: 'n' Symbol: ' ' Code: 0100011
Prior Symbol: 'n' Symbol: ' ' Code: 111100
Prior Symbol: 'n' Symbol: ' ' Code: 011011010
Prior Symbol: 'n' Symbol: ' ' Code: 01100
Prior Symbol: 'n' Symbol: ' ' Code: 011011011
Prior Symbol: 'n' Symbol: 'a' Code: 11111
Prior Symbol: 'n' Symbol: 'b' Code: 011011100
Prior Symbol: 'n' Symbol: 'c' Code: 01001
Prior Symbol: 'n' Symbol: 'd' Code: 110
Prior Symbol: 'n' Symbol: 'e' Code: 001
Prior Symbol: 'n' Symbol: 'f' Code: 01000101
Prior Symbol: 'n' Symbol: 'g' Code: 000
Prior Symbol: 'n' Symbol: 'i' Code: 01111
Prior Symbol: 'n' Symbol: 'j' Code: 011011101
Prior Symbol: 'n' Symbol: 'k' Code: 1111010
Prior Symbol: 'n' Symbol: 'l' Code: 01101100
Prior Symbol: 'n' Symbol: 'm' Code: 011011110
Prior Symbol: 'n' Symbol: 'n' Code: 01110
Prior Symbol: 'n' Symbol: 'o' Code: 1111011
Prior Symbol: 'n' Symbol: 'r' Code: 011011111
Prior Symbol: 'n' Symbol: 's' Code: 0101
Prior Symbol: 'n' Symbol: 't' Code: 1110
Prior Symbol: 'n' Symbol: 'u' Code: 0100001
Prior Symbol: 'n' Symbol: 'v' Code: 0110100
Prior Symbol: 'n' Symbol: 'y' Code: 0110101
Prior Symbol: 'n' Symbol: 'z' Code: 01000100
Prior Symbol: 'o' Symbol: '27' Code: 101010011
Prior Symbol: 'o' Symbol: ' ' Code: 001
Prior Symbol: 'o' Symbol: ' ' Code: 01001111
Prior Symbol: 'o' Symbol: ' ' Code: 01001110
Prior Symbol: 'o' Symbol: ' ' Code: 0100110
Prior Symbol: 'o' Symbol: 'B' Code: 101010010
Prior Symbol: 'o' Symbol: 'a' Code: 100001
Prior Symbol: 'o' Symbol: 'b' Code: 110111
Prior Symbol: 'o' Symbol: 'c' Code: 100000
Prior Symbol: 'o' Symbol: 'd' Code: 110101
Prior Symbol: 'o' Symbol: 'e' Code: 1010101

Prior Symbol: 'o' Symbol: 'f' Code: 000
Prior Symbol: 'o' Symbol: 'g' Code: 1101000
Prior Symbol: 'o' Symbol: 'h' Code: 1101001
Prior Symbol: 'o' Symbol: 'i' Code: 1101101
Prior Symbol: 'o' Symbol: 'k' Code: 010010
Prior Symbol: 'o' Symbol: 'l' Code: 0101
Prior Symbol: 'o' Symbol: 'm' Code: 1100
Prior Symbol: 'o' Symbol: 'n' Code: 111
Prior Symbol: 'o' Symbol: 'o' Code: 10100
Prior Symbol: 'o' Symbol: 'p' Code: 01000
Prior Symbol: 'o' Symbol: 'r' Code: 011
Prior Symbol: 'o' Symbol: 's' Code: 10001
Prior Symbol: 'o' Symbol: 't' Code: 10010
Prior Symbol: 'o' Symbol: 'u' Code: 1011
Prior Symbol: 'o' Symbol: 'v' Code: 101011
Prior Symbol: 'o' Symbol: 'w' Code: 10011
Prior Symbol: 'o' Symbol: 'x' Code: 10101000
Prior Symbol: 'o' Symbol: 'y' Code: 1101100
Prior Symbol: 'o' Symbol: '27' Code: 011011
Prior Symbol: 'p' Symbol: ' ' Code: 000
Prior Symbol: 'p' Symbol: ' ' Code: 1010010
Prior Symbol: 'p' Symbol: ' ' Code: 101000
Prior Symbol: 'p' Symbol: 'a' Code: 001
Prior Symbol: 'p' Symbol: 'e' Code: 110
Prior Symbol: 'p' Symbol: 'f' Code: 1111
Prior Symbol: 'p' Symbol: 'i' Code: 1011
Prior Symbol: 'p' Symbol: 'l' Code: 010
Prior Symbol: 'p' Symbol: 'm' Code: 1010011
Prior Symbol: 'p' Symbol: 'o' Code: 0111
Prior Symbol: 'p' Symbol: 'p' Code: 11101
Prior Symbol: 'p' Symbol: 'r' Code: 100
Prior Symbol: 'p' Symbol: 's' Code: 01100
Prior Symbol: 'p' Symbol: 't' Code: 11100
Prior Symbol: 'p' Symbol: 'u' Code: 10101
Prior Symbol: 'p' Symbol: 'v' Code: 011010
Prior Symbol: 'q' Symbol: '27' Code: 0
Prior Symbol: 'q' Symbol: 'u' Code: 1
Prior Symbol: 'r' Symbol: '27' Code: 10011111
Prior Symbol: 'r' Symbol: ' ' Code: 111
Prior Symbol: 'r' Symbol: ' ' Code: 1001110
Prior Symbol: 'r' Symbol: ' ' Code: 10011100
Prior Symbol: 'r' Symbol: ' ' Code: 11001100
Prior Symbol: 'r' Symbol: ' ' Code: 10001
Prior Symbol: 'r' Symbol: ' ' Code: 100111101
Prior Symbol: 'r' Symbol: ' ' Code: 1101
Prior Symbol: 'r' Symbol: 'b' Code: 11001101
Prior Symbol: 'r' Symbol: 'c' Code: 100001
Prior Symbol: 'r' Symbol: 'd' Code: 11000
Prior Symbol: 'r' Symbol: 'e' Code: 101
Prior Symbol: 'r' Symbol: 'f' Code: 110011111
Prior Symbol: 'r' Symbol: 'g' Code: 100101
Prior Symbol: 'r' Symbol: 'i' Code: 010
Prior Symbol: 'r' Symbol: 'k' Code: 110010
Prior Symbol: 'r' Symbol: 'l' Code: 00100
Prior Symbol: 'r' Symbol: 'm' Code: 00101
Prior Symbol: 'r' Symbol: 'n' Code: 01100
Prior Symbol: 'r' Symbol: 'o' Code: 000
Prior Symbol: 'r' Symbol: 'p' Code: 11001110
Prior Symbol: 'r' Symbol: 'r' Code: 100110
Prior Symbol: 'r' Symbol: 's' Code: 0111
Prior Symbol: 'r' Symbol: 't' Code: 0011
Prior Symbol: 'r' Symbol: 'u' Code: 100000
Prior Symbol: 'r' Symbol: 'v' Code: 110011110
Prior Symbol: 'r' Symbol: 'y' Code: 01101
Prior Symbol: 's' Symbol: '27' Code: 10011100
Prior Symbol: 's' Symbol: ' ' Code: 0
Prior Symbol: 's' Symbol: ' ' Code: 100111100
Prior Symbol: 's' Symbol: ' ' Code: 100111101
Prior Symbol: 's' Symbol: ' ' Code: 111011
Prior Symbol: 's' Symbol: ' ' Code: 1000
Prior Symbol: 's' Symbol: ' ' Code: 11101011
Prior Symbol: 's' Symbol: 'a' Code: 110011
Prior Symbol: 's' Symbol: 'b' Code: 100111110
Prior Symbol: 's' Symbol: 'c' Code: 10010
Prior Symbol: 's' Symbol: 'e' Code: 1101
Prior Symbol: 's' Symbol: 'h' Code: 11000
Prior Symbol: 's' Symbol: 'i' Code: 11100

Prior Symbol: 's' Symbol: 'k' Code: 100111111
 Prior Symbol: 's' Symbol: 'l' Code: 1110100
 Prior Symbol: 's' Symbol: 'm' Code: 111010100
 Prior Symbol: 's' Symbol: 'n' Code: 111010101
 Prior Symbol: 's' Symbol: 'o' Code: 11110
 Prior Symbol: 's' Symbol: 'p' Code: 1001101
 Prior Symbol: 's' Symbol: 'q' Code: 11111
 Prior Symbol: 's' Symbol: 'r' Code: 101
 Prior Symbol: 's' Symbol: '27' Code: 110010
 Prior Symbol: 's' Symbol: 'w' Code: 10011101
 Prior Symbol: 's' Symbol: 'y' Code: 1001100
 Prior Symbol: 't' Symbol: '27' Code: 11000011
 Prior Symbol: 't' Symbol: '' Code: 111
 Prior Symbol: 't' Symbol: '"' Code: 11000100
 Prior Symbol: 't' Symbol: ',' Code: 0111100
 Prior Symbol: 't' Symbol: '.' Code: 01111110
 Prior Symbol: 't' Symbol: ':' Code: 01101
 Prior Symbol: 't' Symbol: ';' Code: 110000100
 Prior Symbol: 't' Symbol: '0' Code: 0100
 Prior Symbol: 't' Symbol: 'b' Code: 110000101
 Prior Symbol: 't' Symbol: 'c' Code: 11000101
 Prior Symbol: 't' Symbol: 'e' Code: 101
 Prior Symbol: 't' Symbol: 'h' Code: 00
 Prior Symbol: 't' Symbol: 'i' Code: 1101
 Prior Symbol: 't' Symbol: 'l' Code: 0111101
 Prior Symbol: 't' Symbol: 'm' Code: 01111111
 Prior Symbol: 't' Symbol: 'n' Code: 01111110
 Prior Symbol: 't' Symbol: 'o' Code: 100
 Prior Symbol: 't' Symbol: 'r' Code: 11001
 Prior Symbol: 't' Symbol: 's' Code: 0101
 Prior Symbol: 't' Symbol: 't' Code: 01100
 Prior Symbol: 't' Symbol: 'u' Code: 01110
 Prior Symbol: 't' Symbol: 'w' Code: 1100000
 Prior Symbol: 't' Symbol: 'y' Code: 1100011
 Prior Symbol: 'u' Symbol: '27' Code: 1001100
 Prior Symbol: 'u' Symbol: '' Code: 100000
 Prior Symbol: 'u' Symbol: 'a' Code: 100111
 Prior Symbol: 'u' Symbol: 'b' Code: 100001
 Prior Symbol: 'u' Symbol: 'c' Code: 10001

Prior Symbol: 'u' Symbol: 'd' Code: 11100
 Prior Symbol: 'u' Symbol: 'e' Code: 11101
 Prior Symbol: 'u' Symbol: 'g' Code: 11110
 Prior Symbol: 'u' Symbol: 'i' Code: 10010
 Prior Symbol: 'u' Symbol: 'k' Code: 1001101
 Prior Symbol: 'u' Symbol: 'l' Code: 0100
 Prior Symbol: 'u' Symbol: 'm' Code: 111111
 Prior Symbol: 'u' Symbol: 'n' Code: 110
 Prior Symbol: 'u' Symbol: 'o' Code: 11111010
 Prior Symbol: 'u' Symbol: 'p' Code: 0101
 Prior Symbol: 'u' Symbol: 'r' Code: 00
 Prior Symbol: 'u' Symbol: 's' Code: 011
 Prior Symbol: 'u' Symbol: 't' Code: 101
 Prior Symbol: 'u' Symbol: 'v' Code: 11111011
 Prior Symbol: 'u' Symbol: 'y' Code: 1111100
 Prior Symbol: 'v' Symbol: '27' Code: 00010
 Prior Symbol: 'v' Symbol: 'a' Code: 001
 Prior Symbol: 'v' Symbol: 'e' Code: 1
 Prior Symbol: 'v' Symbol: 'i' Code: 01
 Prior Symbol: 'v' Symbol: 'o' Code: 0000
 Prior Symbol: 'v' Symbol: 's' Code: 000110
 Prior Symbol: 'v' Symbol: 'y' Code: 000111
 Prior Symbol: 'w' Symbol: '27' Code: 011101
 Prior Symbol: 'w' Symbol: '' Code: 001
 Prior Symbol: 'w' Symbol: 'l' Code: 011100
 Prior Symbol: 'w' Symbol: 'a' Code: 010
 Prior Symbol: 'w' Symbol: 'e' Code: 1110
 Prior Symbol: 'w' Symbol: 'h' Code: 000
 Prior Symbol: 'w' Symbol: 'i' Code: 10
 Prior Symbol: 'w' Symbol: 'l' Code: 011110
 Prior Symbol: 'w' Symbol: 'm' Code: 011111
 Prior Symbol: 'w' Symbol: 'n' Code: 11111
 Prior Symbol: 'w' Symbol: 'o' Code: 110
 Prior Symbol: 'w' Symbol: 'r' Code: 0110
 Prior Symbol: 'w' Symbol: 's' Code: 11110
 Prior Symbol: 'x' Symbol: '27' Code: 10
 Prior Symbol: 'x' Symbol: '' Code: 0110
 Prior Symbol: 'x' Symbol: 'y' Code: 0111
 Prior Symbol: 'x' Symbol: 'z' Code: 1100

Prior Symbol: 'x' Symbol: 'a' Code: 111
 Prior Symbol: 'x' Symbol: 'e' Code: 00
 Prior Symbol: 'x' Symbol: 'i' Code: 010
 Prior Symbol: 'x' Symbol: 'l' Code: 1101
 Prior Symbol: 'y' Symbol: '27' Code: 01010
 Prior Symbol: 'y' Symbol: '' Code: 1
 Prior Symbol: 'y' Symbol: '"' Code: 010010
 Prior Symbol: 'y' Symbol: ',' Code: 0001
 Prior Symbol: 'y' Symbol: ':' Code: 0111
 Prior Symbol: 'y' Symbol: ';' Code: 011001
 Prior Symbol: 'y' Symbol: '?' Code: 0100110
 Prior Symbol: 'y' Symbol: 'a' Code: 0100111
 Prior Symbol: 'y' Symbol: 'b' Code: 0110000
 Prior Symbol: 'y' Symbol: 'd' Code: 000001
 Prior Symbol: 'y' Symbol: 'e' Code: 0010
 Prior Symbol: 'y' Symbol: 'f' Code: 0110001
 Prior Symbol: 'y' Symbol: 'i' Code: 000010
 Prior Symbol: 'y' Symbol: 'l' Code: 01000
 Prior Symbol: 'y' Symbol: 'm' Code: 000000
 Prior Symbol: 'y' Symbol: 'n' Code: 01011
 Prior Symbol: 'y' Symbol: 'o' Code: 01101
 Prior Symbol: 'y' Symbol: 's' Code: 0011
 Prior Symbol: 'y' Symbol: 'w' Code: 000011
 Prior Symbol: 'z' Symbol: '27' Code: 100
 Prior Symbol: 'z' Symbol: '' Code: 1110
 Prior Symbol: 'z' Symbol: 'l' Code: 1111
 Prior Symbol: 'z' Symbol: 'a' Code: 000
 Prior Symbol: 'z' Symbol: 'e' Code: 001
 Prior Symbol: 'z' Symbol: 'i' Code: 110
 Prior Symbol: 'z' Symbol: 'l' Code: 010
 Prior Symbol: 'z' Symbol: 'o' Code: 101
 Prior Symbol: 'z' Symbol: 'z' Code: 011
 Prior Symbol: '}' Symbol: '27' Code: 1
 Prior Symbol: '|' Symbol: '27' Code: 1
 Prior Symbol: '|}' Symbol: '27' Code: 1
 Prior Symbol: '~' Symbol: '27' Code: 1
 Prior Symbol: '127' Symbol: '27' Code: 1

Table F.7 English-language Program Description Decode Table

0	1	76	1	152	2	228	5	304	155	380	3	456	36
1	0	77	240	153	242	229	208	305	155	381	4	457	37
2	1	78	1	154	2	230	6	306	155	382	5	458	38
3	44	79	242	155	252	231	6	307	155	383	207	459	39
4	1	80	1	156	3	232	6	308	155	384	6	460	40
5	46	81	248	157	8	233	52	309	155	385	201	461	41
6	1	82	1	158	3	234	6	310	155	386	249	462	42
7	48	83	250	159	16	235	96	311	155	387	234	463	244
8	1	84	1	160	3	236	6	312	155	388	235	464	43
9	50	85	252	161	26	237	134	313	155	389	245	465	44
10	1	86	1	162	3	238	6	314	155	390	246	466	45
11	52	87	254	163	40	239	146	315	155	391	7	467	46
12	1	88	2	164	3	240	6	316	155	392	8	468	47
13	54	89	0	165	42	241	170	317	155	393	9	469	225
14	1	90	2	166	3	242	6	318	155	394	178	470	48
15	56	91	4	167	52	243	184	319	155	395	197	471	49
16	1	92	2	168	3	244	6	320	155	396	198	472	50
17	58	93	22	169	74	245	220	321	155	397	177	473	51
18	1	94	2	170	3	246	6	322	155	398	10	474	52
19	60	95	32	171	90	247	236	323	155	399	238	475	53
20	1	96	2	172	3	248	6	324	155	400	203	476	54
21	62	97	34	173	94	249	238	325	155	401	11	477	55
22	1	98	2	174	3	250	6	326	155	402	212	478	155
23	64	99	44	175	100	251	240	327	155	403	12	479	155
24	1	100	2	176	3	252	6	328	155	404	196	480	3
25	66	101	50	177	110	253	242	329	155	405	200	481	4
26	1	102	2	178	3	254	6	330	155	406	210	482	128
27	68	103	56	179	112	255	244	331	155	407	13	483	174
28	1	104	2	180	3	256	20	332	155	408	14	484	200
29	70	105	60	181	114	257	21	333	155	409	15	485	212
30	1	106	2	182	3	258	155	334	155	410	199	486	1
31	72	107	64	183	116	259	214	335	155	411	202	487	2
32	1	108	2	184	3	260	201	336	155	412	206	488	155
33	74	109	68	185	118	261	207	337	155	413	208	489	160
34	1	110	2	186	3	262	215	338	155	414	215	490	155
35	76	111	70	187	120	263	199	339	155	415	16	491	155
36	1	112	2	188	3	264	1	340	155	416	194	492	155
37	78	113	74	189	122	265	162	341	155	417	17	493	155
38	1	114	2	190	3	266	206	342	155	418	204	494	155
39	80	115	76	191	124	267	203	343	155	419	236	495	155
40	1	116	2	192	3	268	2	344	155	420	229	496	155
41	82	117	84	193	126	269	3	345	155	421	231	497	155
42	1	118	2	194	3	270	197	346	155	422	18	498	2
43	84	119	86	195	128	271	204	347	155	423	205	499	243
44	1	120	2	196	3	272	198	348	155	424	19	500	160
45	86	121	88	197	180	273	200	349	155	425	20	501	244
46	1	122	2	198	3	274	4	350	155	426	195	502	155
47	88	123	90	199	206	275	196	351	155	427	21	503	1
48	1	124	2	200	3	276	5	352	155	428	22	504	155
49	90	125	92	201	240	277	194	353	155	429	23	505	155
50	1	126	2	202	4	278	6	354	155	430	237	506	172
51	92	127	94	203	26	279	195	355	155	431	24	507	155
52	1	128	2	204	4	280	210	356	155	432	25	508	155
53	94	129	96	205	88	281	7	357	155	433	242	509	155
54	1	130	2	206	4	282	211	358	155	434	26	510	155
55	96	131	98	207	110	283	8	359	155	435	211	511	155
56	1	132	2	208	4	284	202	360	155	436	27	512	1
57	98	133	118	209	142	285	212	361	155	437	28	513	160
58	1	134	2	210	4	286	9	362	56	438	228	514	155
59	100	135	132	211	172	287	205	363	57	439	29	515	162
60	1	136	2	212	4	288	208	364	173	440	193	516	7
61	102	137	148	213	216	289	10	365	175	441	227	517	8
62	1	138	2	214	4	290	193	366	183	442	30	518	226
63	104	139	162	215	224	291	11	367	218	443	233	519	228
64	1	140	2	216	4	292	12	368	168	444	240	520	229
65	106	141	178	217	244	293	13	369	179	445	226	521	230
66	1	142	2	218	5	294	14	370	181	446	247	522	160
67	222	143	186	219	36	295	15	371	1	447	31	523	242
68	1	144	2	220	5	296	16	372	2	448	243	524	225
69	224	145	200	221	64	297	17	373	155	449	230	525	1
70	1	146	2	222	5	298	18	374	180	450	32	526	2
71	234	147	210	223	118	299	19	375	241	451	33	527	243
72	1	148	2	224	5	300	155	376	162	452	34	528	227
73	236	149	222	225	174	301	155	377	213	453	232	529	3
74	1	150	2	226	5	302	155	378	214	454	239	530	4
75	238	151	234	227	206	303	155	379	217	455	35	531	5

532	155	610	8	688	155	766	155	844	229	922	231	1000	10
533	6	611	9	689	5	767	227	845	247	923	8	1001	239
534	4	612	230	690	2	768	239	846	214	924	9	1002	225
535	128	613	245	691	3	769	1	847	225	925	228	1003	232
536	202	614	243	692	229	770	245	848	155	926	10	1004	11
537	211	615	244	693	236	771	229	849	233	927	227	1005	12
538	162	616	155	694	155	772	2	850	242	928	11	1006	13
539	1	617	228	695	239	773	3	851	1	929	237	1007	14
540	155	618	1	696	1	774	233	852	2	930	12	1008	19
541	2	619	237	697	242	775	4	853	3	931	243	1009	20
542	3	620	2	698	5	776	229	854	4	932	13	1010	167
543	160	621	3	699	6	777	3	855	239	933	14	1011	187
544	155	622	4	700	245	778	155	856	5	934	15	1012	230
545	160	623	242	701	239	779	233	857	6	935	236	1013	237
546	3	624	5	702	155	780	1	858	174	936	16	1014	247
547	4	625	6	703	236	781	225	859	1	937	244	1015	231
548	155	626	236	704	233	782	239	860	155	938	17	1016	246
549	183	627	238	705	1	783	2	861	238	939	18	1017	1
550	244	628	7	706	225	784	3	862	233	940	242	1018	2
551	160	629	160	707	242	785	4	863	2	941	160	1019	155
552	176	630	5	708	2	786	167	864	229	942	19	1020	238
553	243	631	6	709	229	787	238	865	155	943	20	1021	3
554	1	632	155	710	3	788	236	866	160	944	21	1022	4
555	2	633	236	711	4	789	242	867	1	945	238	1023	236
556	185	634	245	712	3	790	243	868	3	946	22	1024	5
557	2	635	1	713	4	791	1	869	4	947	23	1025	245
558	184	636	2	714	155	792	155	870	155	948	11	1026	6
559	155	637	225	715	229	793	2	871	232	949	12	1027	172
560	160	638	239	716	233	794	225	872	229	950	228	1028	228
561	1	639	229	717	245	795	6	873	225	951	243	1029	249
562	174	640	233	718	225	796	155	874	239	952	155	1030	242
563	2	641	242	719	1	797	232	875	1	953	174	1031	7
564	182	642	3	720	239	798	233	876	233	954	226	1032	8
565	155	643	4	721	2	799	1	877	2	955	1	1033	9
566	1	644	6	722	4	800	242	878	155	956	2	1034	174
567	160	645	7	723	5	801	236	879	155	957	3	1035	10
568	160	646	155	724	160	802	2	880	155	958	236	1036	239
569	1	647	233	725	201	803	239	881	239	959	160	1037	11
570	155	648	249	726	243	804	3	882	155	960	4	1038	225
571	176	649	242	727	155	805	229	883	155	961	233	1039	243
572	174	650	245	728	174	806	4	884	155	962	242	1040	12
573	1	651	1	729	242	807	5	885	155	963	245	1041	233
574	155	652	2	730	1	808	155	886	155	964	5	1042	13
575	160	653	3	731	2	809	155	887	155	965	249	1043	14
576	174	654	236	732	3	810	3	888	155	966	225	1044	15
577	1	655	239	733	238	811	4	889	155	967	6	1045	16
578	160	656	225	734	239	812	155	890	155	968	239	1046	229
579	155	657	4	735	5	813	174	891	155	969	7	1047	17
580	155	658	232	736	155	814	1	892	155	970	229	1048	18
581	155	659	5	737	174	815	233	893	155	971	8	1049	160
582	155	660	5	738	233	816	2	894	155	972	9	1050	29
583	1	661	6	739	229	817	225	895	155	973	10	1051	30
584	172	662	249	740	1	818	229	896	24	974	15	1052	169
585	174	663	242	741	245	819	239	897	25	975	16	1053	232
586	155	664	245	742	2	820	9	898	232	976	241	1054	245
587	155	665	155	743	225	821	10	899	239	977	174	1055	155
588	2	666	229	744	3	822	246	900	248	978	196	1056	1
589	3	667	239	745	4	823	249	901	155	979	249	1057	173
590	155	668	1	746	229	824	1	902	167	980	172	1058	187
591	160	669	2	747	3	825	174	903	247	981	1	1059	235
592	181	670	233	748	225	826	227	904	250	982	227	1060	250
593	182	671	225	749	233	827	233	905	1	983	2	1061	2
594	184	672	3	750	242	828	245	906	2	984	155	1062	167
595	1	673	4	751	155	829	155	907	3	985	242	1063	230
596	155	674	6	752	1	830	229	908	4	986	3	1064	226
597	160	675	7	753	2	831	239	909	229	987	4	1065	231
598	155	676	225	754	3	832	2	910	174	988	160	1066	3
599	160	677	233	755	4	833	3	911	5	989	236	1067	4
600	155	678	238	756	155	834	225	912	230	990	245	1068	5
601	155	679	246	757	233	835	4	913	226	991	5	1069	6
602	155	680	228	758	245	836	232	914	6	992	6	1070	233
603	155	681	236	759	1	837	5	915	246	993	233	1071	248
604	155	682	243	760	229	838	6	916	235	994	7	1072	7
605	155	683	1	761	2	839	244	917	245	995	235	1073	172
606	155	684	2	762	239	840	7	918	233	996	8	1074	239
607	160	685	242	763	225	841	8	919	7	997	244	1075	240
608	155	686	3	764	225	842	232	920	240	998	9	1076	8
609	155	687	4	765	5	843	7	921	249	999	229	1077	237

1078	246	1156	225	1234	15	1312	18	1390	19	1468	4	1546	226
1079	249	1157	8	1235	16	1313	19	1391	20	1469	239	1547	235
1080	9	1158	9	1236	17	1314	20	1392	228	1470	5	1548	237
1081	247	1159	232	1237	238	1315	21	1393	21	1471	233	1549	238
1082	10	1160	10	1238	18	1316	12	1394	22	1472	6	1550	155
1083	11	1161	11	1239	19	1317	13	1395	23	1473	232	1551	247
1084	174	1162	229	1240	3	1318	167	1396	160	1474	160	1552	1
1085	12	1163	12	1241	239	1319	187	1397	24	1475	225	1553	2
1086	227	1164	160	1242	155	1320	155	1398	26	1476	236	1554	3
1087	13	1165	13	1243	225	1321	1	1399	27	1477	7	1555	187
1088	229	1166	13	1244	229	1322	249	1400	194	1478	242	1556	249
1089	244	1167	14	1245	245	1323	174	1401	155	1479	8	1557	240
1090	14	1168	167	1246	1	1324	226	1402	173	1480	229	1558	4
1091	15	1169	172	1247	2	1325	2	1403	172	1481	9	1559	5
1092	228	1170	243	1248	8	1326	237	1404	248	1482	10	1560	236
1093	16	1171	173	1249	9	1327	243	1405	1	1483	11	1561	6
1094	236	1172	1	1250	236	1328	3	1406	174	1484	12	1562	7
1095	17	1173	2	1251	249	1329	245	1407	2	1485	13	1563	8
1096	225	1174	155	1252	167	1330	239	1408	3	1486	155	1564	245
1097	18	1175	249	1253	238	1331	240	1409	229	1487	245	1565	225
1098	19	1176	245	1254	1	1332	4	1410	231	1488	25	1566	9
1099	20	1177	174	1255	172	1333	5	1411	232	1489	26	1567	172
1100	21	1178	3	1256	155	1334	233	1412	249	1490	169	1568	227
1101	22	1179	238	1257	174	1335	6	1413	233	1491	187	1569	10
1102	238	1180	4	1258	2	1336	7	1414	235	1492	246	1570	232
1103	243	1181	242	1259	3	1337	8	1415	4	1493	230	1571	11
1104	23	1182	5	1260	4	1338	9	1416	227	1494	1	1572	233
1105	24	1183	6	1261	243	1339	160	1417	225	1495	155	1573	12
1106	242	1184	244	1262	5	1340	225	1418	5	1496	173	1574	239
1107	160	1185	7	1263	233	1341	229	1419	246	1497	226	1575	243
1108	25	1186	8	1264	6	1342	10	1420	6	1498	240	1576	174
1109	26	1187	9	1265	160	1343	11	1421	228	1499	2	1577	13
1110	27	1188	239	1266	7	1344	25	1422	7	1500	167	1578	14
1111	28	1189	225	1267	229	1345	26	1423	226	1501	3	1579	229
1112	9	1190	160	1268	22	1346	173	1424	240	1502	4	1580	15
1113	10	1191	10	1269	23	1347	187	1425	8	1503	5	1581	16
1114	174	1192	233	1270	167	1348	226	1426	9	1504	245	1582	17
1115	155	1193	11	1271	173	1349	234	1427	243	1505	227	1583	244
1116	236	1194	12	1272	238	1350	237	1428	244	1506	172	1584	18
1117	1	1195	229	1273	227	1351	242	1429	247	1507	231	1585	19
1118	245	1196	20	1274	235	1352	250	1430	239	1508	242	1586	20
1119	2	1197	21	1275	242	1353	230	1431	10	1509	6	1587	21
1120	244	1198	172	1276	155	1354	236	1432	11	1510	235	1588	20
1121	230	1199	226	1277	226	1355	1	1433	12	1511	7	1589	21
1122	3	1200	248	1278	1	1356	2	1434	13	1512	236	1590	187
1123	225	1201	155	1279	2	1357	3	1435	236	1513	237	1591	226
1124	229	1202	174	1280	245	1358	155	1436	14	1514	238	1592	173
1125	233	1203	250	1281	3	1359	245	1437	15	1515	249	1593	237
1126	4	1204	1	1282	244	1360	4	1438	16	1516	8	1594	1
1127	242	1205	235	1283	172	1361	167	1439	245	1517	174	1595	155
1128	239	1206	2	1284	4	1362	246	1440	237	1518	9	1596	167
1129	5	1207	160	1285	5	1363	249	1441	17	1519	10	1597	227
1130	6	1208	3	1286	230	1364	5	1442	230	1520	228	1598	172
1131	7	1209	4	1287	237	1365	6	1443	160	1521	11	1599	236
1132	160	1210	240	1288	246	1366	235	1444	18	1522	12	1600	238
1133	8	1211	5	1289	6	1367	239	1445	242	1523	244	1601	2
1134	14	1212	6	1290	174	1368	7	1446	19	1524	13	1602	247
1135	15	1213	230	1291	240	1369	8	1447	20	1525	243	1603	3
1136	173	1214	246	1292	7	1370	9	1448	21	1526	14	1604	4
1137	231	1215	7	1293	8	1371	10	1449	238	1527	15	1605	249
1138	155	1216	228	1294	243	1372	172	1450	22	1528	16	1606	5
1139	167	1217	237	1295	9	1373	11	1451	23	1529	225	1607	6
1140	249	1218	231	1296	10	1374	12	1452	24	1530	239	1608	7
1141	1	1219	8	1297	228	1375	227	1453	25	1531	17	1609	8
1142	236	1220	225	1298	11	1376	174	1454	14	1532	233	1610	244
1143	2	1221	239	1299	12	1377	13	1455	15	1533	18	1611	174
1144	172	1222	242	1300	249	1378	238	1456	173	1534	19	1612	245
1145	242	1223	9	1301	13	1379	233	1457	237	1535	229	1613	9
1146	3	1224	10	1302	239	1380	14	1458	249	1536	20	1614	10
1147	174	1225	11	1303	14	1381	225	1459	155	1537	160	1615	242
1148	243	1226	236	1304	225	1382	15	1460	174	1538	21	1616	225
1149	245	1227	12	1305	15	1383	243	1461	1	1539	22	1617	243
1150	4	1228	229	1306	16	1384	16	1462	243	1540	23	1618	11
1151	5	1229	227	1307	233	1385	17	1463	2	1541	24	1619	12
1152	239	1230	13	1308	236	1386	244	1464	3	1542	160	1620	13
1153	6	1231	244	1309	17	1387	18	1465	245	1543	22	1621	233
1154	7	1232	14	1310	160	1388	231	1466	244	1544	162	1622	14
1155	233	1233	243	1311	229	1389	229	1467	240	1545	167	1623	15

1624	239	1647	227	1670	5	1693	3	1716	229	1739	238	1762	236
1625	229	1648	233	1671	229	1694	229	1717	3	1740	6	1763	250
1626	16	1649	5	1672	243	1695	4	1718	155	1741	239	1764	155
1627	160	1650	228	1673	249	1696	232	1719	4	1742	7	1765	239
1628	232	1651	229	1674	155	1697	160	1720	17	1743	172	1766	233
1629	17	1652	231	1675	1	1698	225	1721	160	1744	229	1767	1
1630	18	1653	6	1676	239	1699	5	1722	191	1745	243	1768	2
1631	19	1654	236	1677	2	1700	239	1723	225	1746	8	1769	3
1632	17	1655	240	1678	3	1701	6	1724	226	1747	9	1770	4
1633	18	1656	7	1679	225	1702	7	1725	230	1748	10	1771	5
1634	239	1657	8	1680	4	1703	8	1726	237	1749	174	1772	155
1635	246	1658	9	1681	233	1704	233	1727	228	1750	11	1773	155
1636	155	1659	10	1682	10	1705	9	1728	233	1751	12	1774	155
1637	235	1660	11	1683	11	1706	5	1729	247	1752	13	1775	155
1638	249	1661	243	1684	174	1707	6	1730	167	1753	14	1776	155
1639	1	1662	12	1685	155	1708	160	1731	1	1754	15	1777	155
1640	160	1663	244	1686	236	1709	172	1732	2	1755	16	1778	155
1641	226	1664	238	1687	237	1710	173	1733	187	1756	6	1779	155
1642	2	1665	13	1688	1	1711	244	1734	3	1757	7	1780	155
1643	225	1666	242	1689	2	1712	233	1735	4	1758	160	1781	155
1644	3	1667	14	1690	243	1713	1	1736	236	1759	174		
1645	237	1668	15	1691	238	1714	2	1737	5	1760	225		
1646	4	1669	16	1692	242	1715	225	1738	155	1761	229		