Doc. A/97
16 November 2004

ATSC Standard: Software Download Data Service

Advanced Television Systems Committee
1750 K Street, N.W.
Suite 1200
Washington, D.C. 20006
www.atsc.org

Table of Contents

Index of Tables and Figures

ATSC Standard: Software Download Data Service

1. SCOPE

1.1 Purpose

This document specifies a data service that may be used to download software to a terminal device using a [MPEG2] Transport Stream via an appropriate physical layer. This service may be used to effect updates or upgrades of firmware, operating system software, device driver software, native application software, middleware, and other types of software that reside in a terminal device.

This document specifies standard announcement, signaling, and encapsulation for the delivery of this download data service.

The content and format of the software download data is not defined by this Standard. The formats and interpretations of the software download payload are defined by each user of this Standard.

1.2 Application

The behavior and facilities of this Standard are intended to apply primarily to terrestrial television broadcast systems and receivers. In addition, the same behavior and facilities may be specified and/or applied to other transport systems (such as cable or satellite).

1.3 Organization

This specification is organized as follows:

- **Section 1** – Describes purpose, application and organization of this specification
- **Section 2** – Enumerates normative and informative references
- **Section 3** – Defines acronyms, terminology, and conventions
- **Section 4** – Provides an overview of the software download data service
- **Section 5** – Specifies forward channel transport of software download data service

This Standard makes use of certain notational devices to provide valuable informative and explanatory information in the context of normative and, occasionally, informative sections. These devices take the form of paragraphs labeled as *Example* or *Note*. In each of these cases, the material is to be considered informative in nature.

2. REFERENCES

This section defines the normative and informative references employed by this specification.

2.1 Normative References

The following documents contain provisions which, through reference in this specification, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All referenced documents are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent edition of the referenced document.

When a conflict exists between this specification and a referenced document, this specification takes precedence.

> *Note:* This specification uses a reference notation based on acronyms or convenient labels for identifying a reference (as opposed to using numbers).

[A90] ATSC Data Broadcast Standard, Sections 6, 7 & 13, ATSC Standard A/90, 26 July 2000, ATSC.

[A65] Program and System Information Protocol for Terrestrial Broadcast and Cable, Revision B, ATSC Standard A/65B, 18 March 2003, ATSC.

[UTF8] Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane., ISO/IEC 10646-1:2000.

[MPEG2] Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems, ISO/IEC 13818-1:2000, ISO.

2.2 Informative References

[CDS] HOST-POD Interface Standard, Section 8.15, ANSI/SCTE 28, SCTE.

[DVBD] Digital Video Broadcasting (DVB); Specification for System Software Update in DVB Systems, ETSI 102 006, ETSI.

[OUI] IEEE OUI and Company_id Assignments, http://standards.ieee.org/regauth/oui/index.shtml.

[PKCS7] PKCS #7: Cryptographic Message Syntax Standard, RSA Data Security, ftp://ftp.rsasecurity.com/pub/pkcs/ps/pkcs-7.ps. See also, IETF RFC 2315.

[SCTE23] Data-Over-Cable Systems 1.1, Baseline Privacy Plus Interface Specification, SCTE 23-2 2002, SCTE.

[SCTE28] HOST-POD Interface Standard, SCTE 28 2003, SCTE.

2.3 Reference Acquisition

ATSC Standards – Advanced Television Systems Committee (ATSC), 1750 K Street N.W., Suite 1200, Washington, DC 20006 USA; Phone: +1 202 872-9160; Fax: +1 202 872-9161; http://www.atsc.org/.

ISO Standards – International Organization for Standardization (ISO), 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland; Phone: +41 22 749 01 11; Fax: +41 22 733 34 30; http://www.iso.ch/.

SCTE Standards – Society of Cable Telecommunications Engineers, 140 Philips Road, Exton, PA 19341, USA; Phone: +1 610 363 6888; Fax: +1 610 363 5898; http://www.scte.org/.

ETSI Standards and Reports – 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex – France; Phone +33 4 92 94 42 00; Fax: +33 4 93 65 47 16; http://www.etsi.org.

3. DEFINITIONS

This section defines conformance keywords, acronyms and abbreviations, and terms as employed by this specification.

3.1 Conformance Keywords

As used in this document, the conformance keyword *shall* denotes a mandatory provision of the standard. The keyword *should* denotes a provision that is recommended but not mandatory. The keyword *may* denotes a feature whose presence does not preclude compliance, that may or may not be present at the option of the implementers.

3.2 Acronyms and Abbreviations

bslbf – bit serial leftmost bit first

uimsbf – unsigned integer most significant bit first

OUI – Organizationally Unique Identifier

POD – Point Of Deployment

SDO – Standards Development Organization

3.3 Terms

carousel – A download scenario where the modules are repeated. See [A90].

download scenario – The collection of DSM-CC control and data messages with the same DownloadID value. See [A90].

software download data service – A data service as defined by this specification.

module – The payload data that is delivered to the receiver. See [A90].

group – A collection of messages and modules. See [A90].

3.4 Data 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).

4. OVERVIEW AND ARCHITECTURE

The entirety of this section and its subsections is informative.

4.1 Application Scenarios

In order to better understand the design rationale, outlined here are several application scenarios.

Scenario 1: A carousel broadcasts modules targeting a single device.

- Typically used in an environment where there is no aggregator and a single manufacturer is creating a carousel to support only one hardware/software model.
- The carousel targets a single hardware/software version at a time.
- Additional hardware/software versions would be supported by terminating the carousel and restarting with new announcement, signaling, and data.
- This is the simplest model.

Scenario 2: A carousel broadcasts modules targeting multiple devices.

- Typically used in an environment where there is an aggregator supporting multiple manufacturers or where a single manufacturer would like to support multiple hardware/software versions on a single carousel.
- The carousel targets multiple hardware/software versions at a time.

- Announcements are a critical part of the functionality of this scenario.

Scenario 3: Multiple carousels.

- Typically used in an environment where there is a combination of aggregators and/or individual manufacturers all creating carousels for a single transport.
- Each virtual channel may contain carousel(s) of Scenario 1 or Scenario 2.
- Multiple Virtual Channels containing carousels for software downloads may exist on a single transport.

  Note: Scenario 1 is a simplification of 2, and Scenario 2 is a simplification of Scenario 3.

4.2 Architecture

The software download data service is based on the "2-layer carousel scenario" of [A90], including the DSI, DII and DDB messages as describe in Section 7 of [A90]. Top level signaling is accomplished via a new VCT [A65] service_type. Announcement is accomplished via schedule information added to the DSI. Thus, the DSI is used for both signaling and announcement; and the DII is used for some parts of the module signaling.

For more information on this underlying design and architecture, please see [A90], Section 7, and [DVBD].

The application scenarios described above are supported by this design as follows:

- Scenario 1 is a single carousel in a single Virtual Channel and contains only a single Group. Multiple downloads are managed serially through complete version changes of all DSI and DII messages.
- Scenario 2 is also a single carousel in a single Virtual Channel, but it makes use of multiple Groups. Multiple downloads are managed concurrently through changes to the DSI and DII messages as needed.
- Scenario 3 is a combination of scenarios 1 and/or 2 in multiple Virtual Channels. Multiple downloads are managed as in those scenarios on a per Channel basis.

5. FORWARD CHANNEL TRANSPORT

This section specifies the transport of a software download data service in the forward (broadcast) channel in the terms of the following mechanisms:

- Announcement
- Signaling
- Encapsulation

Signaling is the indication of what is being carried in the transport "now". Announcement is the signaling of future times for the download. Encapsulation is the binding of the download payload to the transport.

The software download data service shall comply with the 2-layer carousel scenario as defined in [A90], Section 7, except as constrained and extended by this Standard.

5.1 Compatibility Descriptor

Both announcement and signaling of a software download data service shall use the Compatibility Descriptor as defined in [A90] Section 6, (also known as the [A90] groupCompatibility) with the following constraints applied:

1) The descriptorCount field shall be 0x0001 or greater.
2) The descriptorType field of the first descriptor shall be 0x01 (system hardware).
3) The specifierType field of the first descriptor shall be 0x01 (IEEE OUI).
4) The specifierData field of the first descriptor shall be an OUI value assigned by IEEE to the manufacturer of the terminal device(s) to which the software download is intended to be applied. See [OUI].
5) The descriptorType field of the second descriptor, when present, shall be 0x02 (system software) and constraints (3) and (4) above shall apply.

The model and version fields are required, and their meaning should be defined by the manufacturer identified by the OUI value in the specifierData field. Manufacturers may define and use the subDescriptor field per [A90].

5.2 Signaling

A software download data service shall be signaled in the ATSC Transport using a Virtual Channel with a VCT service_type of 0x05.

A Virtual Channel with this service_type value shall have exactly one Program Element of stream_type 0x0B, and may have one Program Element of stream_type 0x95. The Program Element of stream_type 0x0B shall contain a "DSM-CC carousel scenario" as more fully defined here and in [A90]. The Program Element shall contain the minimum DSI, DII, and DDB messages as further defined in the encapsulation Section 5.3 below.

Each DII message shall contain information for only one manufacturer.

Optionally, multiple Virtual Channels of service_type 0x05 may be used in a single Transport Stream.

As constrained by [A90], the downloadId field is constant on any one Program Element.

5.3 Encapsulation

Each manufacturer's data shall be encapsulated as one or more modules signaled in a unique DII message as defined by [A90] Section 7.

The organization and content of the module(s) that comprise the data associated with a single manufacturer are not defined by this specification, but are expected to be defined by each implementer.

5.3.1 DownloadInfoIndication (DII)

The DownloadInfoIndication (DII) messages of [A90] Section 7 are transmitted for each manufacturer, as identified by each related groupCompatibility structure in the DSI. The DII fields shall be as described in [A90]. In addition, they are constrained as follows:

1) The moduleInfoByte field shall be set to the descriptorStructure defined in Section 5.3.1.1, and may contain one or more descriptors. It may contain the moduleInfoDescriptor and scheduleDescriptor. When more than one module is being signaled, it shall contain the moduleInfoDescriptor.

2) The privateDataByte field shall be set to the descriptorStructure defined in Section 5.3.1.1 and may contain one or more descriptors.

As required by [A90], the moduleId field is unique across all DII messages within any one Program Element (and thus Virtual Channel containing this data service).

5.3.1.1 descriptorStructure

The descriptorStructure is a generic descriptor loop used to contain either module or group-specific descriptors. It is defined in Table 5.1 below.

Table 5.1 descriptorStructure

Syntax	No. of Bits	Format
descriptorStructure() {		
for (i = 0; i < N; i++) {		
descriptor()	8	uimsbf
}		
}		

descriptor – this field may contain zero or more MPEG-2 compliant descriptors as further described in this Standard.

5.3.1.2 ModuleInfoDescriptor

The moduleInfoDescriptor is used to define module-specific information. The syntax of the descriptor shall be as defined in Table 5.2.

Table 5.2 ModuleInfoDescriptor

Syntax	No. of Bits	Format
moduleInfoDescriptor() {		
descriptorTag	8	uimsbf
descriptorLength	8	uimsbf
encoding	8	uimsbf
nameLength	8	uimsbf
for (i = 0; i<nameLength; i++) {		
nameByte	8	uimsbf
signatureType	8	uimsbf
signatureLength	8	uimsbf
for (i = 0; i < signatureLength; i++) {		
signatureByte	8	uimsbf
privateModuleLength	8	uimsbf
for (i = 0; i < privateModuleLength; i++) {		
privateModuleByte	8	uimsbf
}		
}		

descriptorTag – This field shall be set to 0xB7.

descriptorLength – This field shall be set to the length in bytes in this descriptor following this field.

encoding – This field shall be set to the type of encoding used for the module. The values are defined in Table 5.3 below.

nameLength – This field shall be set to the length, in bytes, of the nameByte field.

nameByte – This field shall be set to the string name of the module encoded as UTF-8 [UTF8].

signatureType – The type of module signature shall be set as defined in Table 5.4 below.

signatureLength – This field shall be set to the length, in bytes, of the signatureByte field.

signatureByte – The signature bytes shall be set as defined by the signatureType value.

privateModuleLength – This field shall be set to the length, in bytes, of the privateModuleByte.

privateModuleByte – This field may be used by the manufacturer defined in the groupCompatibility field to provide more module-specific information.

Table 5.3 Encoding Values

Encoding Value	Definition
0x00	None
0x01-0xFF	ATSC Reserved[1]

Table 5.4 signatureType Values

signatureType Value	Definition
0x00	None
0x01-0xFF	ATSC Reserved[2]

1 Any SDO that participates in the ATSC code point management process may request a value in this range.

2 Any SDO that participates in the ATSC code point management process may request a value in this range.

5.3.2 DownloadDataBock (DDB)

The DownloadDataBlock (DDB) shall be used to convey the blocks of the modules and shall conform to [A90]. Each module shall be encoded within the data bytes field as defined by the encoding field of the moduleInfoDescriptor of Section 5.3.1.2.

5.3.3 DownloadServerInitiate (DSI)

The DownloadServerInitiate (DSI) message shall be used to signal one or more DII messages and shall be present even if there is only one DII needed. That is, this shall be a 2-layer carousel design at all times.

The DSI fields are as defined and constrained by [A90]. Additionally:

1) The groupCompatibility field shall be populated with the constrained Compatibility Descriptor structure defined in Section 5.1.
2) The groupInfoByte shall be set to the descriptorStructure.
3) For each group, either there shall be a DII signaled, or there shall be a scheduleDescriptor.
4) For each schedule period signaled in the scheduleDescriptor, a DII shall be emitted during that period.

The complete DSI message shall be transmitted no less than once every 60 seconds.

5.4 Announcement

Announcement is the schedule information about module transmissions occurring in the future. A software download data service may be announced using the scheduleDescriptor and, when used, shall be placed either in the DSI descriptorStructure descriptor loop to signal group schedules, or the DII descriptorStructure descriptor loop to signal module schedules. The scheduleDescriptor shall be defined as in Table 5.5 below.

Table 5.5 Schedule Descriptor

Syntax	No. of Bits	Format
scheduleDescriptor() {		
descriptorTag	8	uimsbf
descriptorLength	8	uimsbf
for (i = 0; i < N; i++) {		
startTime	32	uimsbf
reserved	12	uimsbf
lengthInSeconds	20	uimsbf
}		
}		

descriptorTag – Shall be set to 0xBA.

descriptorLength – Shall be set to the length in bytes in this descriptor following this length field.

startTime – This field shall be set to the start time of the download window period for either the group or the module. This is the beginning of a period that may contain the download defined by the groupDescriptor and other fields and descriptors. The field shall be set as per the EIT start_time field as defined in A65, Section 6.5.

reserved – Reserved and shall be set to 0xFFF.

lengthInSeconds – This field shall be set to the duration, in seconds, of the download window period. The field shall be set as per the EIT length_in_seconds field as defined in [A65], Section 6.5.

Note: The startTime and lengthInSeconds fields define a “download period” during which the download describe by this descriptor will occur. This window may not be precise but is assured to bound the time one or more sets of DDB messages will be emitted. This is different from the DII field, tCDownloadScenario, which bounds the entire DII-signaled period.

5.5 Buffer Model

This section defines the Transport System Target Decoder (T-STD) model for the download service.

The buffer model block diagram is shown in Figure 5.1 below. Refer to [MPEG2] Section 2.4.2 for context, and refer to Figure 5.2. The transport packet bytes containing DSM-CC (MPEG) sections shall not be transferred to TB_{sys}, but to TB_{data} as more fully described below. All other behavior of the transport buffer (TB_{data}) shall be as defined in [MPEG2] for other transport buffers.

t(i) is as defined in [MPEG2]. TB_{data} is the transport buffer for the Program Element containing the DSM-CC sections of the download carousel transport packets. As provided in [MPEG2], the transport buffer size (TBS_{data}) is 512 bytes. RX_{data} is the rate bytes are moved from the transport buffer to the smoothing buffer, SB_{data}; SBS_{data} is the size of the smoothing buffer; and R_{data} is the rate bytes are moved out of the smoothing buffer. All four of these parameters are defined further below depending on the presence of descriptors in the PMT.

The [MPEG2] maximum_bitrate_descriptor may be emitted to indicate RX_{data}. When present it shall be placed in the ES_info descriptor loop of the PMT Program Element containing the DSM_CC sections of the download carousel, and RX_{data} shall be set to the value of the maximum_bitrate field. When not present, RX_{data} shall be defined to be equal to 1.20 x R_{data}.

The [MPEG2] smoothing_buffer_descriptor shall be emitted and placed in the ES_info descriptor loop of the PMT Program Element containing the DSM-CC sections of the download carousel. R_{data} shall be set to the value of the sb_leak_rate field, and SBS_{data} shall be set to the value of the sb_size field.

In summary:

- t(i) is the ith byte of the transport stream input (if the PID matches) to TB_{data}
- TB_{data} is the transport buffer for the download stream
- TBS_{data} is the size of TB_{data} and is set to 512
- RX_{data} is the leak rate from the transport buffer into the smoothing buffer and may be set by the maximum_bitrate_descriptor
- SB_{data} is the smoothing buffer
- SBS_{data} is the size of SB_{data} and is set by the smoothing_buffer_descriptor
- R_{data} is the leak rate out of the smoothing buffer and is set by the smoothing_buffer_descriptor

The smoothing buffer, SB_{data}, shall not overflow, but may underflow.

Note: This T-STD is compatible with the [A90] T_STD for asynchronous services and the [DVBD] decoder model. It is restated here due to construction constraints in [A90] with respect to the VCT service_type.

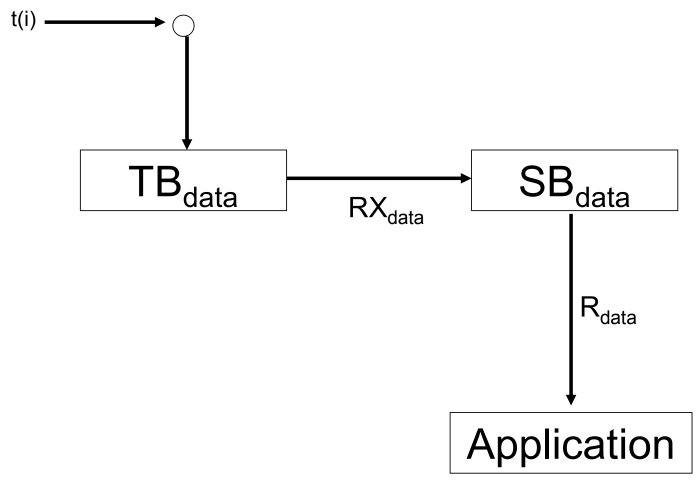

Figure 5.1 T-STD model for software download.

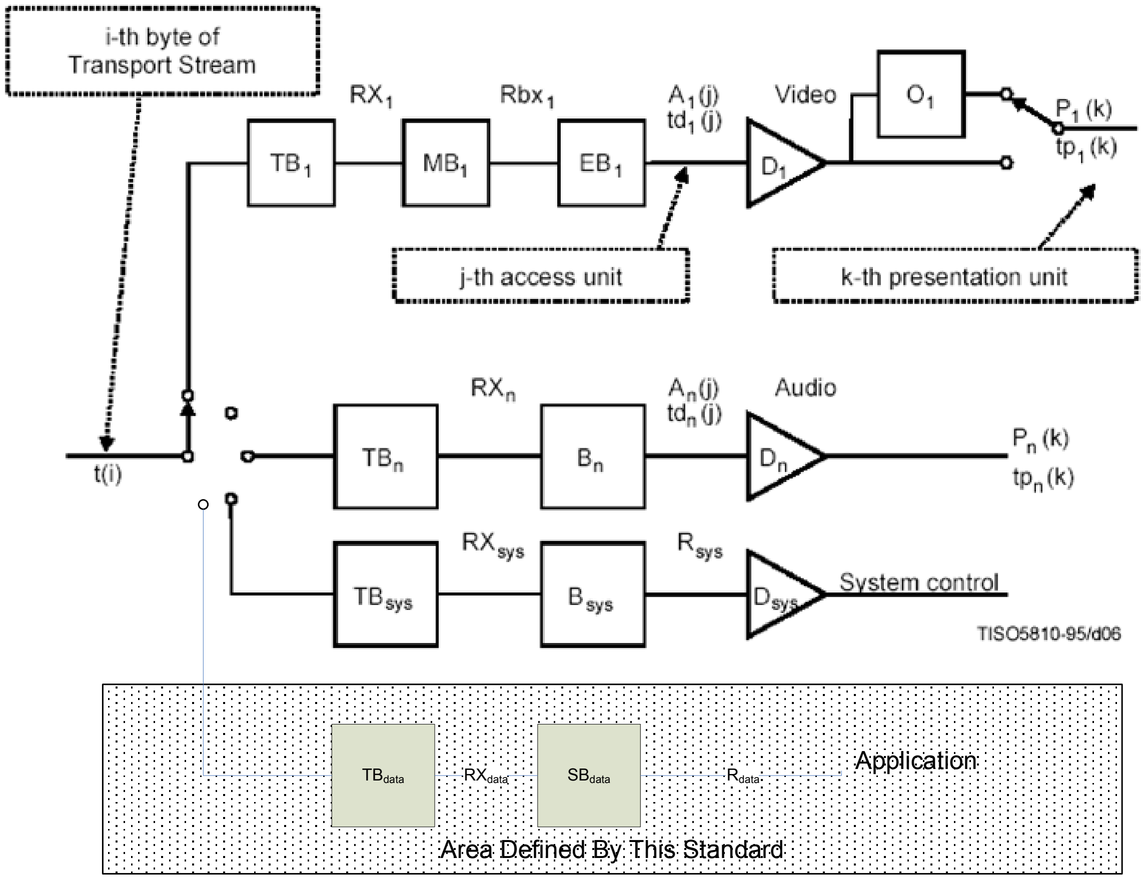

Figure 5.2 T-STD context (informative).