ETSI TS 101 225 V1.1.1 (2001-05)

Technical Specification

Digital Video Broadcasting (DVB); Home Local Network Specification based on IEEE 1394

Reference
DTS/JTC-DVB-85

Keywords
broadcasting, digital, DVB, TV, video

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

1 Scope

The present document standardizes the topology, physical interfaces and a complete stack of protocols for the Home Local Network (HLN). This includes the specification of the APIs that an application on an HLN device can use to access the services provided by this HLN device or any other HLN device, as well as a Java language binding for these APIs. This allows e.g. Java applications to be downloaded to a DVB Receiver and use the services from other HLN devices such as storage devices.

The HLN services and APIs allow any HLN device, to access (interactive) DVB services, Internet services, or other services even if that device itself is not directly connected to a Home Access Network (HAN) or to the corresponding Access Network that delivers these services to the home.

The HLN specification covers the possibility of connecting multiple DVB receivers or devices providing access to other services, to the HLN.

This HLN specification also allows for multiple applications to execute simultaneously on multiple HLN Devices and share, in a cooperative way, the HLN resources such as bandwidth or a DVB receiver.

The present document is based on the commercial requirements in TM1690 (rev 2) (see Bibliography), the role of the HLN in the total architecture of the In-Home Digital Network (IHDN) given in DVB-IHDN 012 (rev 6) (see Bibliography), as well as the technical requirements given in DVB-IHDN 012 (rev 6) (see Bibliography).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] IEC 60603-7: "Connectors for frequencies below 3 MHz for use with printed boards - Part 7: Detail specification for connectors, 8-way, including fixed and free connectors with common mating features, with assessed quality".

[2] IEC 61754-16: "Fibre optic connector interfaces - Part 16: Type PN connector family".

[3] IEEE 1394-1995: "IEEE Standard for a High Performance Serial Bus".

[4] P1394a (V4.0): "Draft Standard for a High Performance Serial Bus (Supplement)", September 15, 1999 (see note 1).

[5] P1394b (V0.9): "Draft Standard for a High Performance Serial Bus (Supplement), October 6, 1999 (see note 2).

[6] 1394 TRADE ASSOCIATION (Revision 0.96): "Power Specification Part 1: Cable Power Distribution", December 9, 1998 (see note 3).

[7] 1394 TRADE ASSOCIATION (Draft 0.90): "Power Specification Part 2: Suspend/Resume Implementation Guidelines", January 4, 1999 (see note 3).

[8] 1394 TRADE ASSOCIATION (Draft 0.72): "Power Specification Part 3: Power State Management", October 21, 1998 (see note 3).

[9] IEC 61883-1 (1998-02): "Consumer audio/video equipment – Digital interface – Part 1: General" (see note 4).

[10] IEC 61883-2 (1998-02): "Consumer audio/video equipment – Digital interface – Part 2: SD-DVCR data transmission" (see note 4).

[11] IEC 61883-3 (1998-02): "Consumer audio/video equipment – Digital interface – Part 3: HD-DVCR data transmission" (see note 2).

[12] IEC 61883-4 (1998-02): "Consumer audio/video equipment – Digital interface – Part 4: MPEG2-TS data transmission" (see note 4).

[13] IEC 61883-5 (1998-02): "Consumer audio/video equipment – Digital interface – Part 5: SDL-DVCR data transmission" (see note 4).

[14] ISO/IEC 13818-1: "Information technology - Generic coding of moving pictures and associated audio information: Systems".

[15] ETSI EN 300 468 (V1.3.1): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[16] "Specification of the Home Audio/Video Interoperability (HAVi) Architecture" (Version 1.0) January 2000 (see note 5).

[17] RFC 2734: "IPv4 over IEEE 1394" (see note 6).

[18] RFC 1918: "Address Allocation for Private Internets" (see note 7).

[19] RFC 1631: "The IP Network Address Translator (NAT)" (see note 1).

[20] RFC 2131: "Dynamic Host Configuration Protocol" (see note 1).

[21] RFC 1542: "Clarifications and Extensions for Bootstrap Protocol" (see note 1).

[22] IETF draft, DHCP for IEEE 1394, February 2000 (see note 7).

[23] DAVIC specification version 1.4 (see note 8).

[24] ETSI ETS 300 802: "Digital Video Broadcasting (DVB); Network-independent protocols for DVB interactive services".

[25] RFC 2663: "IP Network Address Translator (NAT) Terminology and Considerations" (see note 1).

[26] ETSI ETR 211: "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

[27] ETSI TR 101 226: "Digital Video Broadcasting (DVB); In-Home Digital Network (IHDN) guidelines".

[28] IEC 61753: "Fibre optic interconnecting devices and passive components - Performance standard".

[29] ISO/IEC 11801: "Information technology - Generic cabling for customer premises".

[30] ISO/IEC 8859: "Information technology - 8-bit single-byte coded graphic character sets".

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

NOTE 1: ftp://ftp.t10.org/1394/P1394a/Drafts/P1394a40.pdf

NOTE 2: http://www.zayante.com/p1394b/index.shtml

NOTE 3: http://www.1394ta.org

NOTE 4: http://www.iec.ch

NOTE 5: http://www.havi.org/techinfo/index.html

NOTE 6: http://www.ietf.org/rfc

NOTE 7: ftp://ftp.ietf.org/internet-drafts/draft-ietf-ip1394-dhcp-03.txt

NOTE 8: http://www.davic.org

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Home Access Network (HAN): As described in the IHDN reference model [2], the Home Access Network is a part of the IHDN. The HAN and its protocol stacks depend on the corresponding Access Network and on the home installations. HAN could e.g. be absent (e.g. when the Access Network such as a CATV is directly attached to a DVB receiver without the use of active elements such as protocol converters or line drivers, see note) or it could be a complete ATM or Ethernet access network.

NOTE: In this case the DVB receiver is a Residential Gateway.

Access Network: network providing DVB or other services to the home
The term Access Network designates network installations that do not belong to the consumer.

HLN Device: device attached to the HLN that is perceived as a box by the consumer and provides a specific functionality to the user
An HLN device is either an End Device or a Residential Gateway.

Residential Gateway (RG): specific type of HLN device that is connected to an (Home) Access Network and to the HLN
It executes specific functions to convey access to DVB or other services to End Devices or other Residential Gateways.

End Device: specific type of HLN device that is not directly connected to a (Home) Access Network

IEEE 1394 Domain: set of devices linked through one IEEE 1394 bus and having the same bus_id
Within one IEEE 1394 Domain the isochronous and asynchronous resources are shared.

DS-port: connector on an HLN devices as defined in clause 5.1
DS stands for "data-strobe", named after the electrical encoding of the signals on the IEEE 1394 cable.

DS-node: HLN device that has only DS-ports
A DS-node can have one or more DS-ports.

DS-link: point-to-point cable between two DS-ports

B-port: connector on an HLN device (as defined in clause5.2)

B-node: HLN device that has only B-ports
A B-node can have one or more B-ports.

B-link: point-to-point cable between two B-ports

Border-node: HLN device that has both DS-ports and B-ports

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming interface
ATM	Asynchronous Transfer Mode
CATV	Community Antenna TV
CMP	Connection Management Procedure (defined in [7])
DAVIC	Digital Audio Video Council
DCM	Device Control Module (defined in [14])
DHCP	Dynamic Host Configuration Protocol
DL	Datalink Layer

DS	Data-Strobe
DV	Digital Video format used by digital camcorders
DVCR	Digital Video Cassette Recorder
DVD	Digital Versatile Disk
D-VHS	Digital VHS
FAV	Full Audio-Video device (defined in [16])
FCM	Function Control Module (defined in [16])
FCP	Function Control Protocol
FTP	File Transfer Protocol
HAN	Home Access Network
HAVi	Home Audio Video interoperability
HDD	Hard-Disk drive
HLN	Home Local Network
HPCF	Hard Polymer Clad-Fibre
HTTP	Hyper Text Transfer Protocol
IAV	Intermediate Audio-Video device (defined in [16])
IDTV	Integrated Digital TV, i.e. a TV containing a STB
IHDN	In-Home Digital Network
IP	Internet Protocol
ISP	Internet Service Provider
MPEG2-pTS	MPEG2 partial Transport Stream (defined in [15])
MPEG2-TS	MPEG2 Transport Stream (defined in [14])
MPTS	Multi Program Transport Stream
NAT	Network Address Translation
PHY	Physical layer
POF	Plastic Optical Fibre
RC	Return Channel
RG	Residential Gateway
SPTS	Single Program Transport Stream
STB	Set Top Box
UI	User Interface
UTP5	Unshielded Twisted Pair (cat. 5)

4 General

4.1 HLN model

A Home Local Network (HLN) consists of several HLN devices connected through a physical network, see Figure 1.

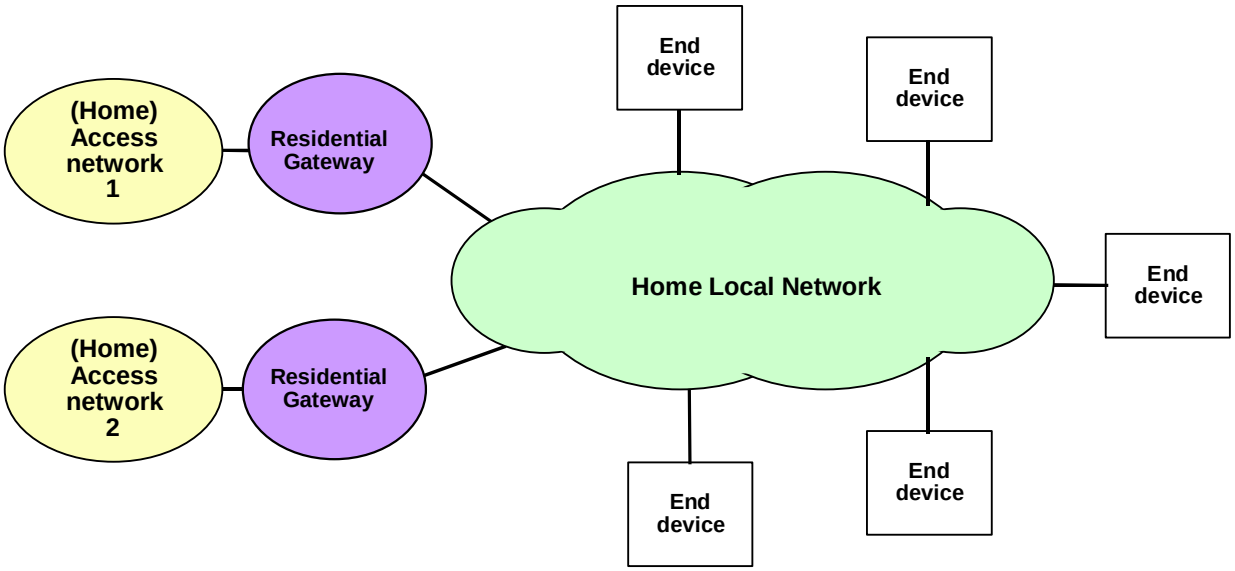

Figure 1: HLN Model

Residential Gateways are special HLN devices that forward information flows from the HLN into a (Home) Access Network or vice versa. For specific types of Access Networks and the services they deliver to the home, the way in which Residential Gateways operate needs to be specified. The present document specifies Residential Gateways for:

- DVB Access Networks
- ATM Home Access Networks
- Ethernet Home Access Networks
- Access Networks for IP based services (see note)

NOTE: This can be a DVB, ATM or Ethernet Access Network

4.2 HLN Topology

Logically the HLN consists of one IEEE 1394 Domain that behaves as a bus. Within one IEEE 1394 Domain, asynchronous and isochronous IEEE 1394 resources are shared.

The physical topology of the HLN is a tree structure consisting of DS-nodes and B-nodes and Border-nodes, connected via DS-links and B-links. The HLN can contain a mixture of DS-links (max. 4,5 m and 400 Mbit/s) and B-links (max. 50 m and 200 Mbit/s, see note) allowing the HLN to span multiple rooms in the home.

NOTE: Three independent users accessing two full DVB Transport Streams of around 41,25 Mbit/s and one partial Transport Stream of around ≈ 15 Mbit/s + additional bandwidth required for asynchronous traffic + additional bandwidth lost in IEEE 1394 overhead (gap-time, packet headers) requires with some performance margin an S200 IEEE 1394 bandwidth.

Loops in the topology will cause the HLN to fail. The existence of loops may be detected automatically but cannot be automatically healed.

A HLN device should contain at least one DS-port to accommodate existing 1394 devices. However as the HLN evolves, B-links and B-nodes may become the dominant technology.

HLN devices should be equipped with three or more ports to allow balanced tree structures to be built. At least one port is required. An HLN device with only one or two ports should be restricted to the case where the physical dimensions of the device prohibits the implementation of more ports. Typical examples of this case are battery operated or hand-held portable devices such as digital camcorders.

4.3 Future extensions (informative)

Once IEEE 1394 bridges have been standardized the HLN may be extended to several IEEE 1394 Domains interconnected by IEEE 1394 bridges. IEEE 1394 bridges prevent that all HLN devices have to share the same IEEE 1394 resources such as bandwidth. However in the near term this is not expected to create problems for the foreseen applications and the available bandwidth.

The present document does not address wireless links for the HLN. Work is ongoing outside DVB to standardize wireless interfaces. Once these are finished they can be integrated into the HLN specification.

4.4 HLN Example (informative)

A possible HLN configuration is given in Figure 2.

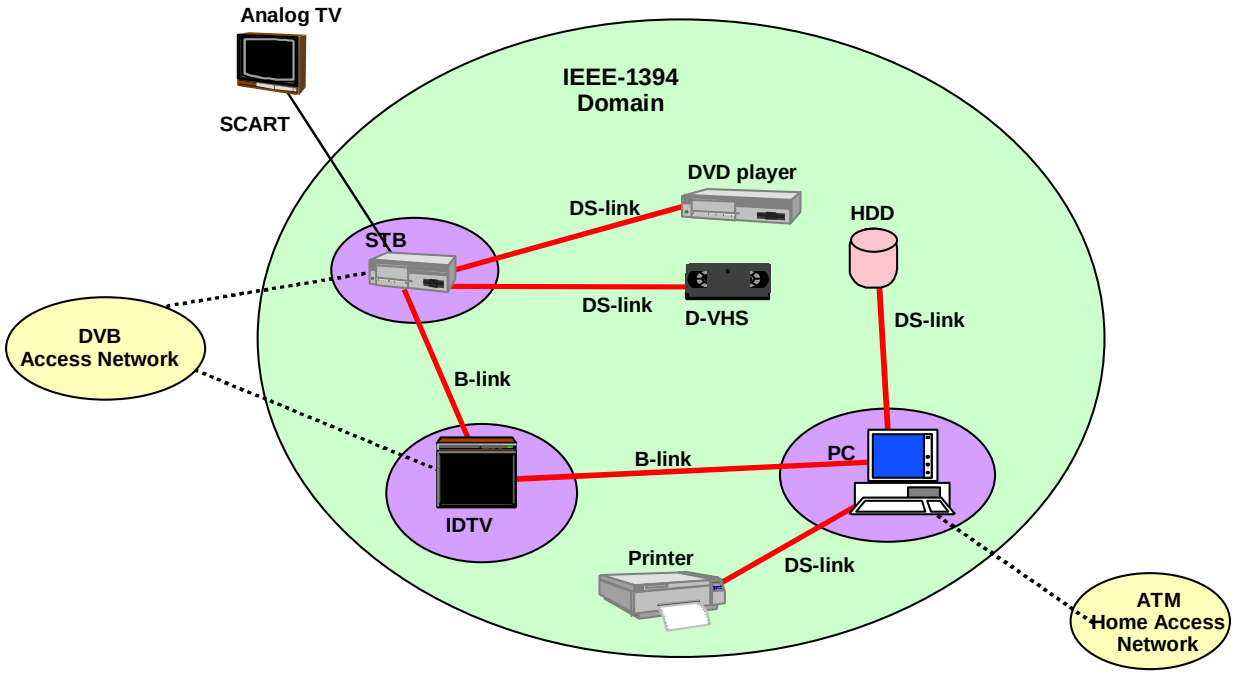

Figure 2: HLN Example

The PC is connected to an ATM specific Home Access Network. The Set Top Box (STB) and the Integrated Digital TV (IDTV) are connected directly to a DVB Access Network (e.g. cable or satellite) without the use of a HAN. In this example they are connected to the same DVB Access Network but this does not need to be the case (e.g. the IDTV could be connected to a DVB cable network, while the STB is connected to a DVB satellite network). In the example the STB, IDTV and PC are all Residential Gateways.

Other configurations are also possible.

5 General HLN specifications

5.1 Media and Connectors

5.1.1 DS-ports and DS-links

DS-ports and DS-links can be used to connect devices within 4,5 m and using bit rates up to 400 Mbit/s. Two types of DS-links and DS-ports are defined: a 4-pin and a 6-pin type. These two types are intended for different applications, either with or without power-provision or power-consumption over the DS-link.

An HLN device shall be equipped with 4-pin DS-ports unless it is a primary- or alternate-power provider or a power consumer as defined in [6], [7] and [8]. If an HLN device is a primary- or alternate-power-provider or a power consumer, it shall be equipped with 6-pin DS-ports. An HLN device shall implement only 4-pin DS-ports or only 6-pin DS-ports.

5.1.2 DS-ports and DS-links

The 4-pin DS-ports and DS-links shall comply to the specifications in section 5 of [4].

5.1.3 DS-ports and DS-links

The 6-pin DS-ports and DS-links shall comply to the specifications in section 4 of [3].

5.1.4 Power over DS-links

IEEE 1394 defines four power classes:

- Primary Power Provider
- Alternate Power Provider
- Power Consumer
- Self-Powered Device

HLN devices with 4-pin DS-ports are Self-Powered devices. HLN devices with 6-pin DS-ports are either Primary Power Provider, Alternate Power Provider or Power Consumer.

HLN devices shall follow the specifications corresponding to their power class as specified in [6], [7] and [8].

5.1.5 B-ports and B-links

B-links and B-ports comply to a sub-set of the P1394b specification [5] and are capable of transporting data-streams up to 200 Mbit/s over a distance up to 50 m (see note).

NOTE: 50 m is the maximum length for the reasons put forth under clause 5.2.2. 100 m variants are under specification for HPCF and UTP5.

5.1.6 B-ports

For each port B-port, an HLN device may choose a possibly different entry from Table 1. The B-port shall comply to the specifications listed in that entry. The grey field is not covered.

Table 1: B-ports

	S100	S200
UTP-5	8 pin "jack" as per IEC 60603-7 [1] and section 10 in [5]	
POF	PN connector (see note) Mechanical std.:IEC 61754-16 [2]; perf std. IEC 61753-AA [28]	PN connector (see note) mechanical std.: IEC 61754-16 [2]; perf std. IEC 61753-AA [28]
HPCF	PN connector (see note) Mechanical std.: IEC 61754-16 [2]; perf std. IEC 61753-AA [28]	PN connector (see note) mechanical std.: IEC 61754-16 [2]; perf std. IEC 61753-AA [28]
NOTE: This is a widely known connector for dual-wired optical media. See also section 9 in [5].		

5.1.7 B-links

For each B-port an HLN device shall comply to the specification of the B-link listed in the entry in Table 2 that corresponds to the entry that was chosen for the B-port in Table 1. The grey field is not covered by the present document.

Table 2: B-links

	S100	S200
UTP-5	ISO/IEC 11801 [29]; Max 50 m see [5], section 10	
POF	Core = 1000 µm step-index, Multi-mode max 50 m see [5], section 9	Core = 1 000 µm step-index, Multi-mode max 50 m see [5], section 9
HPCF	Core = 225 µm Graded-index, Multi-mode Max 50 m see [5], section 9	Core = 225 µm graded-index, Multi-mode max 50 m see [5], section 9

5.2 Physical Layer

5.2.1 DS-ports

The physical layer of a DS-port shall comply to section 4 of [3], where applicable supplemented by [4].

5.2.2 B-ports

The physical layer of a B-port shall comply to the P1394b specification [5], with the following restrictions:

1) The physical layer attached to a long-range connector shall comply to the time-constants values defined in Table15-8 in section 15-6 of [5] for the **root-contention-resolution algorithm**, which are compatible with long-range cables that do not exceed 50 m.

2) The length of a single B-link shall not exceed 50 m.

3) The following sections of [5] are applicable:

Table 3: Applicable B-port physical layer characteristics

Section	Title	Remarks
7	Jitter measurement	
9	Plastic Optical Fibre	Mandatory when POF or HPCF connectors are used
10	Cat 5 UTP physical medium dependent layer	Mandatory when UTP5 connectors are used
11	Beta mode port specification	
12	Connection management	Mandatory
13	PHY-LNK interface	Mandatory if a link layer is implemented, otherwise not applicable
14	PHY register map	Mandatory
15	Data routing, arbitration, and control	Mandatory Since S200 is the max. HLN speed, the beta-only packet format may be ignored. Same remark as for section 16
16	C-code	Informative. Significant simplifications are to be expected if one uses the model of simple repeaters (see note) with no link-layer interface
NOTE: See TR 101 226 [27].		

4) The following sections of [5] are **not** applicable:

Table 4: Inapplicable B-port physical layer characteristics

Section	Title	Remarks
5	Beta and Bilingual Cable Media Attachment	Covers the new (short-range) bilingual connectors and cables
6	Electrical specifications (for short-haul copper)	Deals mainly with the possibility to use beta signalling over short range copper
8	Glass Optical Fibre	

5) Sections 1, 2, 3 and 4 of [5] are informative and contain general information on how to read the standard as well as an introduction and its improvements (and enhancements) over [3] and [4].

5.2.3 Power Specification

An HLN device should ensure that all its DS- and B-ports are powered even if the device is in a low-power or stand-by mode. Users should be warned that connections can be lost in case of power-off or power-unplug.

5.3 Datalink Layer

The Data Link layer is identical for all ports of an HLN device.

The Datalink Layer of IEEE 1394 supports different types of traffic with different behaviour for latency, guaranteed delivery, etc. Which types an HLN device needs to support is determined by other parts of the present document.

5.3.1 Support for Isochronous Streams

When implemented by an HLN device, isochronous data transport in the HLN shall comply to the IEEE 1394 specifications in [3], [4] and [5].

When isochronous data consist of DV it shall comply to the IEC 61883 specifications in [9], [10], [11], [13]. When isochronous data consist of an MPEG2-TS or a MPEG2-pTS as defined in [14] and [15], it shall comply to the IEC 61883 specifications in [9] and [12].

5.3.2 Support for Asynchronous Data and Asynchronous Streams

When implemented by an HLN device, asynchronous data transport and asynchronous streams in the HLN shall comply to the IEEE 1394 specifications [3], [4] and [5].

5.4 Support for IP-based Protocols

There are two possible ways to deal with IP in the HLN: either IP is carried transparently over the HLN, or it is terminated in the Residential Gateway. In the first case the HLN devices have to be IP-capable since they must implement all the applicable protocols of the IP stack. In the second case, HLN devices do not need to be IP-capable (but will support IP-based protocols like ftp, http, etc.).

5.4.1 Transparent IP over the HLN

When IP is carried transparently over the HLN, it shall comply to the IETF specifications [17]. An IP-capable HLN device can now perform either Intranet (i.e. some IP dialogue with another IP-capable HLN device), or Internet (e.g. browse the web or receive e-mail), or even both.

5.4.1.1 Intranet over the HLN

This is outside the scope of the present document.

5.4.1.2 Internet across the HLN

In this case an IP capable HLN device wants to communicate with other IP devices over a HAN or access network via a Residential Gateway connected to that network. There are two possible options for the HLN device to get an IP address:

1) It is given an IP address by some mechanism (not DHCP), such as manually entered or via a smart card. The device continues to use this address until it is changed by a similar mechanism.

2) The use of DHCP as defined in [20]. In this case, the Residential Gateway shall provide one of the mechanisms as described in clause 6.1.

The implementation of both mechanisms is recommended for an IP-capable HLN device.

5.4.2 IP Termination in the residential gateway

IP may also be terminated in the HLN device connected to the IP Access Network (the residential gateway). In that case the IP-based protocols such as FTP, HTTP and POP3 need be transported over the HLN using the HLN protocol stacks. This is described in clause 6.2.3.

5.5 HLN Protocol Stacks and APIs

The definition of S-flows is according to DAVIC [23] and referred to by DVB SIS [24]. In DVB these S-flows flow between applications running on the DVB service provider and the DVB receiver. In the context of the HLN the DVB receiver is a Residential Gateway (RG). It is the task of the Residential Gateway to forward these flows over the HLN to End Devices or other Residential Gateways that do not have direct access to the DVB services. Note that the same kind of flows may also exist within the boundaries of the HLN flowing where the source is an HLN Device.

The protocol stacks as described below shall be used to transport the S1, S2 and S4 flows. The S3 and S5 flows are not applicable to the HLN.

- S1 flow is a unidirectional flow from a source of content (may be a broadcast service provider or a HLN device) to an HLN device. S1 is a HLN user plane flow.

- S2 is a bi-directional control flow between two applications. Both applications may run on (possibly different) HLN devices. One may run on an HLN device while the other may run on a service provider server. S2 is a HLN user plane flow.

- S4 flow provides network connection set up services inside the HLN and on the (Home) Access Network to the applications. S4 is a HLN control plane flow. For certain Access Networks, the S4 flow is absent.

5.5.1 HLN S1 protocol stack

The S1 stack is given in Figure 3. It complies to the isochronous parts of IEEE 1394 Datalink layer as described in annex A. The format specific application layer is not within the scope of the present document and is added for information purposes only.

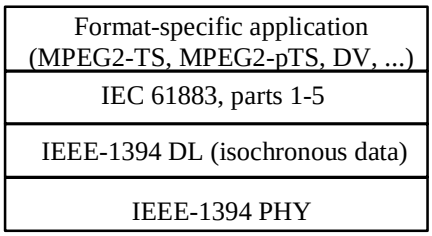

Figure 3: HLN S1 protocol stack

5.5.2 HLN S2 protocol stack

The S2 stack is given in Figure 4. It complies to asynchronous parts of IEEE 1394 as described in annex A, the FCP protocol as defined in IEC 61883 part 1 [9], and the HAVi protocols and APIs [16]. The application layer is not within the scope of the present document and is added for information purposes only.

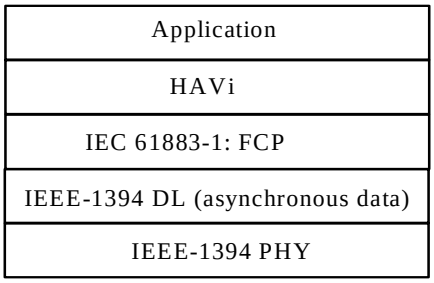

Figure 4: HLN S2 protocol stack

5.5.3 HLN S4 protocol stack

The S4 stack is given in Figure 5. It complies to asynchronous parts of IEEE 1394 as defined in annex A, the FCP and CMP protocols as defined in IEC 61883 part 1 [9], and the HAVi protocols and APIs [16]. The application layer is not within the scope of the present document and is added for information purposes only.

HAVi provides the S4 functionality through the Stream Manager, DCM and general FCM APIs on top of the FCP protocol. The Stream Manager uses the CMP protocol to manage isochronous connections over IEEE 1394.

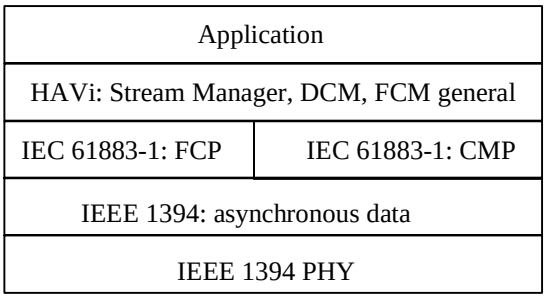

Figure 5 : HLN S4 protocol stack

6 Residential Gateway specifications

A Residential Gateway is one specific type of HLN device. On one hand it is connected to the HLN, on the other hand it is connected to one or more (Home) Access Networks. A Residential Gateway may be integrated in a Set Top Box and thus provide audio and video rendering facilities. A Residential Gateway may also be without any decoders, just in charge of protocol transportation or conversion.

The Residential Gateway aims at providing the service providers and the users with tools (APIs, protocol stacks, ...) that allow a service coming from an Access Network to be selected and forwarded (and therefore received) anywhere in the HLN. Different kinds of Access Networks are envisaged:

- It may be based on a broadcast delivery network (satellite, CATV, SMATV, ... as already specified by DVB).
- It may be a HAN.
- It may be an IP network providing audio and video delivery using new IP-based protocols such as RTP, RTCP and RSVP. IP multicast networks may also convey a wide variety of video and audio formats, where the MPEG2-TS is only one among others. IP networks also provide non-real-time services via protocols such as FTP and HTTP.

All these different topologies are abstracted and masked to the HLN devices by the Residential Gateway, so that End Devices or other Residential Gateways can use a common way to access the services while being unaware of the different Access Networks and their specific protocols stacks (with the exception of the IP Access Network, in the case other HLN devices are IP capable as described in clause 5.4.1).

No specific Residential Gateways for certain Access Networks are mandated for the HLN. However, a Residential Gateway for an Access Network described in this clause shall comply to the corresponding specifications in this clause.

6.1 Involvement of the Residential Gateway in IP address allocation

If the Residential Gateway connects to an IP access network and if it allows other HLN devices to access this network, it shall provide support for the mechanisms to allocate IP addresses to those of them which are IP-capable.

As already described in clause 5.4.1.2, two means are proposed to the HLN device to get an IP address:

- either by configuration (i.e. a fixed IP address),
- or via DHCP as defined in [20].

Therefore a Residential Gateway as described above, shall at least provide one of the following mechanisms:

- a DHCP server
- a BOOTP relay agent

as described in the clauses below together with the case when fixed IP addresses are allocated to the HLN devices.

6.1.1 Policy for a managed IP address domain

This aspect is covered by TR 101 226 [27].

6.1.2 DHCP server in the Residential Gateway

If DHCP, as defined in [20], is used in the Residential Gateway to connect IP-capable HLN devices to the Internet, each IP-capable HLN device is assigned an IP address by a DHCP server in the Residential Gateway, using the protocol defined in [22]. All the IP addresses that are assigned to the HLN devices shall be part of only one of the following address spaces:

- The private address space of the HLN [18]. In this case, this address space is completely decoupled from the address space on the Access Network. In this situation, a NAT as described in [19], or a NAPT as described in [25] has to be used to interconnect IP-capable HLN devices to the Internet. The Residential Gateway shall then perform address translation or address and port translation according to [19] and [25] between its Access Network side IP addresses and the IP addresses of the IP-capable HLN devices.

- The address range that, in some way, is provided to the DHCP server by the ISP. These addresses can either be globally unique or private within the domain of the ISP (see note).

NOTE: In order to enable intranet, these IP addresses should be part of the same IP (sub)net.

One or more IP addresses are also assigned by the ISP or access network to the Residential Gateway at the Access Network side. The Residential Gateway also behaves as an IP router to convey IP traffic between the IP Access Network and the HLN.

6.1.3 BOOTP relay in the Residential Gateway

If a BOOTP relay agent, as described in [21], is used in the Residential Gateway to connect IP-capable HLN devices to the Internet, each IP-capable HLN device is assigned a globally unique IP address, or an IP address from the private address space of the ISP, by a remote DHCP server over the IP Access Network via the BOOTP relay agent of the Residential Gateway.

DHCP is also used to assign one or more IP addresses to the Residential Gateway. The Residential Gateway also behaves as an IP router to convey IP traffic between the IP Access Network and the HLN.

A BOOTP relay agent makes sense in the Residential Gateway only if there is a DHCP server in the IP Access Network.

Furthermore it is advised that, in case that the IP addresses are assigned from the private address space of the ISP, also a NAT as described in [19], or a NAPT as described in [25] will be used to interconnect Residential Gateways, and IP-capable HLN devices to the Internet. This NAT or NAPT should be located outside the Residential Gateway.

6.1.4 Fixed IP addresses

The IP-capable HLN devices may be assigned fixed IP addresses.

The fixed IP addresses of the HLN devices can be provided by the ISP or chosen from the private address space of the HLN, according to [18]. In the situation that the fixed IP addresses of the HLN devices are part of the private address space of the HLN, the Residential Gateway shall be equipped with a NA(P)T as described in [19] and [25].

The fixed IP addresses of the Access Network side of the Residential Gateway are provided by the ISP.

One or more IP addresses are also assigned by the ISP or access network to the Residential Gateway at the Access Network side. The Residential Gateway also behaves as an IP router to convey IP traffic between the IP Access Network and the HLN.

6.2 Protocol conversion

Protocol conversion is defined for three different Access Networks and for each possible S-flow.

6.2.1 A Residential Gateway to DVB broadcast Access Networks

6.2.1.1 S1 flow

Audio, video and data are transmitted over the DVB Access Network in MPEG2-MPTS as defined in [14]. The physical layer and the modulation depend on the DVB Access Network (cable or satellite). The Residential Gateway receives the MPEG2-MPTS and shall perform a partial demultiplexing to build an MPEG2-pTS as defined in [15] to transmit it over the HLN S1 stack as defined in clause 5.5.1.

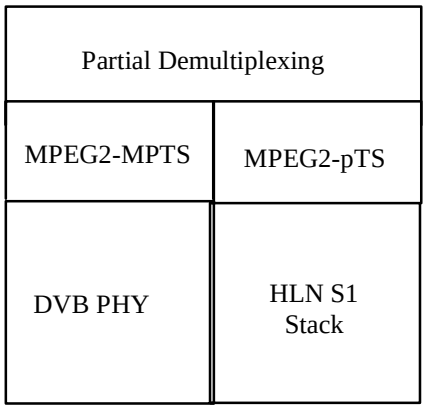

Figure 6: S1 protocol conversion for a DVB Residential Gateway

6.2.1.2 S2 flow

For the S2 flow the Residential Gateway operates in the application layer (i.e. on the top of the layers in Figure 7) as defined in clause 6.3.

The HLN S2 stack is defined in clause 5.5.2.

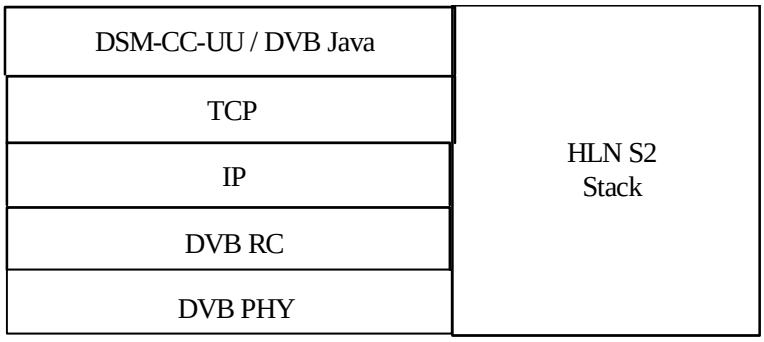

Figure 7: S2 protocol conversion for a DVB Residential Gateway

6.2.1.3 S4 flow

There usually is an S2 dialogue between an HLN device and the Residential Gateway in order to select a service of the DVB Access Network. When used, this S2 dialogue shall be as defined in clause 6.3.

There is no S4 dialogue between the Residential Gateway and the DVB Access Network. All the services are broadcasted to the Residential Gateway which just has to filter part of them using a tuner and a demultiplexer.

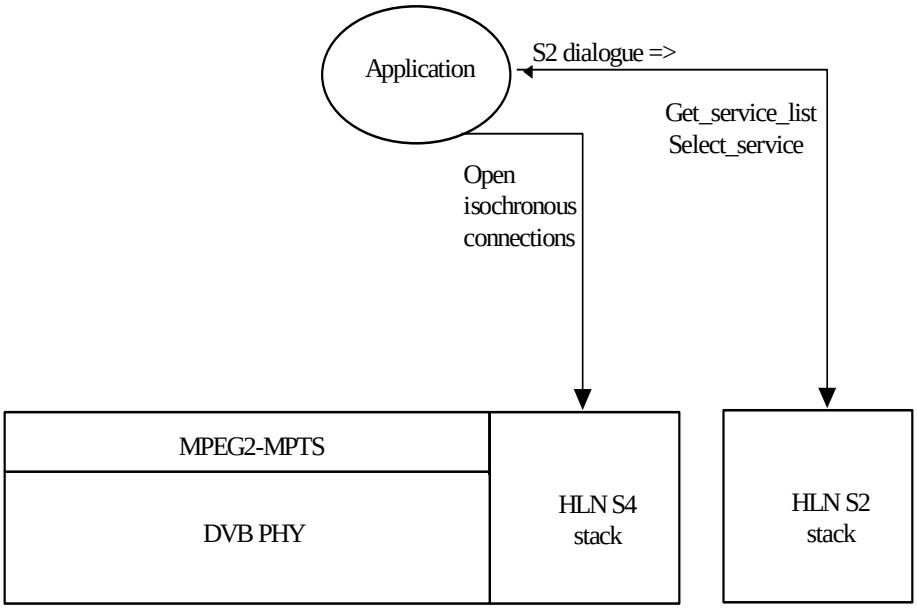

Figure 8: S4 protocol "conversion" for a DVB Residential Gateway

6.2.2 A Residential Gateway to Home Access Networks

The protocol conversions between a RG and a HAN are dealt with in the HAN specifications.

6.2.3 A Residential Gateway to IP Access Networks

When a Residential Gateway connects to an IP Access Network, it is:

1) Either transparent to this IP access network, in the sense that it forwards IP to the IP-capable HLN devices. This is described in clauses 5.4.1 and 6.1.

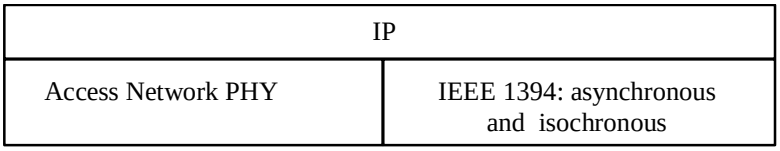

Figure 9: transparent IP access protocol conversion

Remarks for Figure 9:

- in case of NAPT, the RG also needs to operate on TCP level
- for implementing the protocol, the RG may also do static routing

2) Or it converts the IP Access Network protocols to enable non IP-capable devices (i.e. devices that do not have an IP stack) to access a number of IP services.

   This protocol conversion consists of tunnelling the IP-based protocols (i.e. carrying them transparently inside the transport protocol of the HLN). The Web FCM as defined by HAVi [16] shall be used for this purpose.

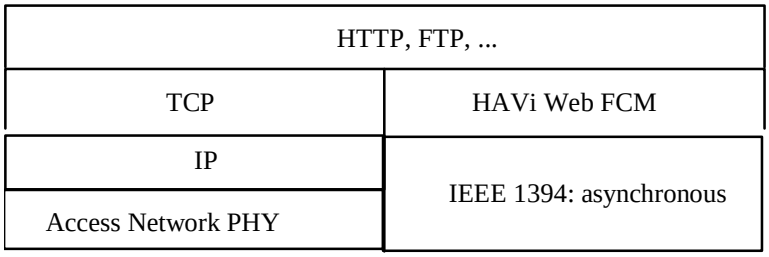

Figure 10: IP non real time protocol conversion

6.3 HAVi Functionality

As defined in clause 6.2, a Residential Gateway takes part in S2 flow protocol conversion using the HAVi protocols and APIs [16]. This clause defines the HAVi functionality of the Residential Gateway.

6.3.1 General

A Residential Gateway shall be an IAV or FAV node as defined by HAVi [16] and shall comply to the conformance requirements defined by HAVi for these device categories.

A Residential Gateway shall implement a DCM to represent itself on the HLN to other HLN devices.

Annex A describes a mapping of relevant HAVi data types and formats for a Residential Gateway. Also Residential Gateway specific extensions to HAVi APIs are specified.

6.3.2 Service Selection

A Residential Gateway shall implement the "Content Icon List" APIs for a DCM as defined by HAVi [16] to represent the selectable services on the Access Network to let other HLN devices select services from it.

The Residential Gateway shall implement the Tuner FCM as defined by HAVi [16] to allow other HLN devices to select services from its lists

6.3.3 Modem access

If the Residential Gateway uses a modem for the DVB Return Channel and wants to allow other HLN devices access to the modem, it shall implement the Modem FCM as defined by HAVi [16].

6.3.4 Internet access

If the Residential Gateway connects to the Internet and wants to allow other HLN devices access to the internet, it shall implement transparent internet access and the Web FCM as defined by HAVi [16].

6.3.5 User Interface (UI)

If the Residential Gateway has a display facility, it shall implement the DDI Controller functionality as defined by HAVi [16] to be able to display a UI for other HLN devices.

A Residential Gateway with a display facility that is an FAV, it shall also implement the HAVi Level 2 User Interface APIs as defined by HAVi [16].

Annex A (normative): DVB specification for the HAVi Tuner FCM

Within the HAVi specification (see [16]), a special Tuner FCM is defined. This FCM has been defined to be applicable to a variety of tuners – from tuners used for analogue radio and television to digital tuners such as those for ATSC, DVB and DSS. This clause specifies how to implement the HAVi tuner FCM for DVB services.

The clause is written assuming the reader is familiar with the following standards [16] (HAVi), [15] and [26] (DVB-SI), [9], [10], [11], [12], and [13] (IEC 61883) and [14] (MPEG-2 systems).

For the DVB-RG, the HAVi tuner FCM has been extended resulting in a "DVB Tuner FCM". The DVB Tuner FCM consists of the HAVi Tuner FCM and several extensions.

The HAVi Tuner FCM is registered as a normal Tuner FCM in the HAVi registry. In addition to the mandatory registry entry that every FCM shall register with (see [16]), the extensions have independent registry entries. Each such registry entry shall have at least the registry attributes vendor_id and software_element_type. The vendor_id identifies this extension as a DVB extension and shall hold the DVB vendor_id (TBD, see note). The software_element_type indicates the extension type and shall hold the corresponding DVB defined value (see clause A.5.1). No other registry attributes as defined by [16] are required.

NOTE: DVB has to request a vendor id from the IEEE Organisationally Unique Identifiers (OUI) and Company identifiers, http://standards.ieee.org/regauth/oui/index.html.

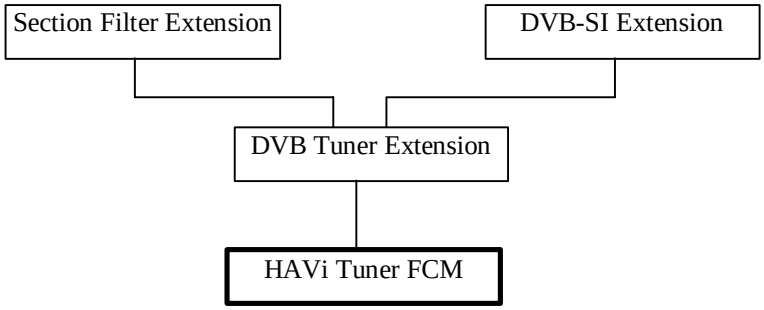

Figure 11: DVB Tuner FCM components

The different parts of the DVB Tuner FCM are:

- **HAVi Tuner FCM** (see clause A.1), a DVB interpretation of the functionality provided by the basic HAVi tuner FCM. This interpretation provides basic SI access and selection of services, and components.
- **DVB Tuner Extension** (see clause A.1), an extension used to link the section filter and DVB SI functionality to the Tuner FCM. The extension provides some general purpose methods (like selection of MPEG2-TS-source without streaming over the IHDN) and acts as access point for DVB-Tuner functionality.
- **DVB-SI Extension** (see clause A.2), an extension to the HAVi tuner FCM allowing access to most SI information and that provides an MHP SI API like interface.
- **Section Filter Extension** (see clause A.3), this extension provides basic section filtering functionality in the incoming transport stream. This functionality allows access to the not commonly used SI information and supports implementing the DVB-SI section filtering API.

To understand the operation and purpose of each extension, it is important to see the extensions in relation to the streaming infrastructure of the RG. The implementation of the section filter extension and DVB-SI extension work on the incoming transport stream, not on the partial transport stream.

For a DVB-RG, the HAVi Tuner FCM and the DVB Tuner Extension are mandatory. An application can locate a HAVi Tuner FCM with DVB defined functionality by locating the registered DVB Tuner Extension. For each registered DVB Tuner Extension, the corresponding HAVi Tuner FCM shall also be registered. The DVB Tuner Extension also provides methods to locate the (optional) Section Filter and DVB-SI extensions.

The HAVi Tuner FCM is the access point to the functionality of the software element. All the HAVi defined functionality is accessed using this base software element, e.g. all resource management of the FCM.

A.1 Mappings for HAVi Tuner FCM

A.1.1 Introduction

This clause specifies how to implement the HAVi tuner FCM for DVB services. Only the basic tuner FCM is discussed, implying that no new methods or functionality is added. This results in a DVB specific HAVi tuner FCM that is completely compliant with the HAVi tuner FCM.

The main reasons why a DVB specific version of the tuner FCM is made are:

- The functionality of the tuner can be enhanced by targeting the tuner more at the intended services. For instance, by defining a DVB specific locator format, elements from the DVB compliant MPEG-2 transport stream can be selected based on the DVB identifiers. This implies that more elements can be chosen on a standardized way than currently is possible.
- The return values of HAVi tuner FCM calls can be defined a standardized way, using the DVB standardized items.
- The set of stream types supported by a DVB tuner FCM and the mapping of DVB streams onto HAVi streams is defined.

Besides the reasons mentioned above, the tuner FCM is also used as the base software element for the DVB specific extensions discussed in the other clauses.

A.1.2 Common definitions

A.1.2.1 Service locator definition

In HAVi, the `ServiceLocator` is defined as octet string of 256 bytes. A DVB `ServiceLocator` occupies 8-15 bytes, depending on the type. The following format for a DVB locator is defined:

Table 5

Type \ octet	0	1	2-3	4-5	6-7	8-9	10	11-12	13-14
Transport Stream	1	0	or_nw_id	nw_id	ts_id	-	-	-	-
Transport Stream	1	1	or_nw_id	-	ts_id	-	-	-	-
DVB-Service	1	10	or_nw_id	nw_id	ts_id	service_id	-	-	-
DVB-Service	1	11	or_nw_id	nw_id	-	service_id	-	-	-
DVB-Service	1	12	or_nw_id	-	ts_id	service_id	-	-	-
DVB-Service	1	13	or_nw_id	-	-	service_id	-	-	-
DVB-Component	1	20	or_nw_id	nw_id	ts_id	service_id	comp_tag	-	-
DVB-Component	1	21	or_nw_id	nw_id	-	service_id	comp_tag	-	-
DVB-Component	1	22	or_nw_id	-	ts_id	service_id	comp_tag	-	-
DVB-Component	1	23	or_nw_id	-	-	service_id	comp_tag	-	-
DVB-Network	1	40	-	nw_id	-	-	-	-	-
DVB-Bouquet	1	50	-	-	-	-	-	bouq_id	-
DVB-Event	1	60	or_nw_id	nw_id	ts_id	service_id	-	-	event_id
DVB-Event	1	61	or_nw_id	nw_id	-	service_id	-	-	event_id
DVB-Event	1	62	or_nw_id	-	ts_id	service_id	-	-	event_id
DVB-Event	1	63	or_nw_id	-	-	service_id	-	-	event_id

In Table 5, the first octet, octet 0, is used to define this locator as a DVB locator. The second octet, octet 1, indicates the DVB locator type (decimal). For each different type, some room is reserved for future expansion. The other fields represent different DVB identifiers: or_nw_id corresponds to the original_network_id value, ts_id to the transport_stream_id value, and nw_id to the network_id value. Furthermore, service_id corresponds to the service_id value, and comp_tag to the component_tag. The fields bouq_id corresponds to the bouquet_id and event_id with the event_id value. All values are defined in [14] and [15]. The sign '-' indicates the value of this field is not defined.

A.1.2.2 HAVi <-> DVB character sets

The DVB-SI strings are coded using different character coding tables (see [15]). The following text is a quotation from [15] explaining the process:

"Text fields can optionally start with non-spacing, non-displayed data which specifies the alternative character table to be used for the remainder of the text item. The selection of character table is indicated as follows:

- *if the first byte of the text field has a value in the range "0x20" to "0xFF" then this and all subsequent bytes in the text item are coded using the default character coding table (table 00 - Latin alphabet) of figure A.1;*
- *if the first byte of the text field has a value in the range "0x01" to "0x05" then the remaining bytes in the text item are coded in accordance with character coding tables 01 to 05 respectively, which are given in figures A.2 to A.6 respectively;*
- *if the first byte of the text field has a value "0x10" then the following two bytes carry a 16-bit value (uimsbf) N to indicate that the remaining data of the text field is coded using the character code table specified by ISO Standard 8859 [5], Parts 1 to 9;*
- *if the first byte of the text field has a value "0x11" then the remaining bytes in the text item are coded in pairs in accordance with the Basic Multilingual Plane of ISO/IEC 10646-1 [8].*

Values for the first byte of "0x00", "0x06" to "0x0F", and "0x12" to "0x1F" are reserved for future use."

The HAVi specification uses Unicode to code the strings. The DVB defined streams have to be transcoded from ASCII to Unicode. Furthermore, DVB has defined special emphasis and carriage return codes that also have to be mapped to the HAVi Unicode strings.

This conversion is easy for the first 128 characters. In this character space the Unicode and ASCII DVB-SI characters overlap. The conversion of the other characters has to be done using a table (see for instance www.unicode.org).

The emphasis marks are removed from the string.

A.1.2.3 Multiple languages

The HAVi "service"-lists provide only one entry for each element. This entry, a "service", contains a name and a language indicator. The name is the name of this "service" (MPEG2-TS, service or component). The indicated language is the language of the "service", not the language of the name of the "service". The manufacturer has the option of indicating an empty string when no language is, or can be associated with the "service".

In DVB-SI it is possible to provide separate names and descriptions, each in a different language. When creating the "service"-lists, the manufacturer has to select one of the available languages. Probably the DCM containing the tuner FCM has a list of preferred languages and will select the name or description accordingly.

The language indicated by a service is to be defined by the manufacturer. When a "service" represents a component, this will be the language of the component. For multi-lingual DVB-services this can be either a preferred language or empty indication. Transport streams have no associated language.

A.1.2.4 Shortening DVB strings

The HAVi specification has limited sizes for all strings. For most entries, the strings defined in DVB-SI have a larger maximum length. For example, the theoretical maximal length of a service name is 254 bytes. Within HAVi this string has to be mapped on the 32 character string of the `Service` structure.

Whenever the string defined within DVB is longer than the corresponding size in HAVi, only the first characters of the DVB string shall be used.

A.1.3 Stream types, connections and transport types

The HAVi specification contains a mechanism to create connections between devices and check whether the devices can communicate with each other. In this system, the information is identified using a stream type and is transported according to a transport type.

A.1.3.1 Stream types

Within the HAVi specifications, stream types are used to indicate the information type and bandwidth requirements. The Tuner FCM uses stream types in two ways:

- Creating connections between Tuner FCM and some destination FCM (e.g. a VCR).
- Identifying the type of a Service.

For streaming information over 1394, the mandatory stream type for DVB tuners is Multiplex/DVB_PTS. This stream type indicates the DVB defined partial transport stream (see [15] and [26]). Some tuner FCMs will also support the transmission of complete transport streams over the IHDN. The stream type of these streams is Multiplex/DVB (see note).

NOTE: Both Multiplex/DVB and Multiplex/DVB_PTS are compliant to the MPEG-2 Transport Stream format [14] so both can map on the stream type Multiplex/MPEG2_TS.

When stream types are used to identify a service, the maximum bitrate field will indicate the expected maximum bitrate for this service/component.

The mapping between MPEG defined stream types and the HAVi defined stream types for service components is shown below:

Table 6: Mapping of MPEG stream type to HAVi stream types

MPEG-2 stream type (PMT)		HAVi
Value	**Description**	**stream type**
0x01	ISO/IEC 11172-2 Video	DigitalVideo/MPEG1
0x02	ITU-T Rec. H. 262 | ISO/IEC 13818-2 Video or ISO/IEC 11172-2 constrained parameter video stream	DigitalVideo/MPEG2_MP_ML || DigitalVideo/MPEG2_MP_HL
0x03	ISO/IEC 11172-3 Audio	DigitalAudio/MPEG1_LayerII || MPEG1_LayerIII
0x04	ISO/IEC 13818-3 Audio	DigitalAudio/MPEG2
0x05	ITU-T Rec. H.222.0| ISO/IEC 13818-1 [14] private_sections	DataStream/DVB
0x06	ITU-T Rec. H.222.0| ISO/IEC 13818-1 PES packets containing private data	DataStream/DVB

Optionally, the manufacturer may replace the DataStream/DVB stream type with the DVB stream type. Within HAVi, a stream type is defined by the `StreamTypeId`. This field consists of the combination of a vendor id and a 16 bit data field:

```
struct StreamTypeId {
   VendorId    vendorId;
   ushort      typeNo;
};
```

In a DVB stream type, the vendor_id field contains the DVB vendor_id. The `typeNo` field holds the stream content and content type identifiers from the DVB component descriptor (see table 16 in [15]). The stream content occupies the most significant byte, the content type the least significant byte.

A.1.3.2 Transport types

Every implementation of a DVB Tuner FCM shall have at least one (DCM) plug capable of sending the TransmissionFormat IEC 61883-MPEG (see [9], [10], [11], [12], and [13]).

A.1.4 Multiplex creation

The application can add and remove elements from the transport stream using the `Tuner::SelectService` call. The result of this call is the addition or removal of specific elements from the transport stream. In this clause, the option replace is interpreted as removal of all components and addition of the requested component.

The information sent over 1394 will be a complete MPEG-2 transport stream. This can be a complete DVB multiplex or a special so-called partial Transport Stream (MPEG2-pTS) (see [15] and [26]).

The MHP tuner API requires the selection of the transport stream source of a tuner without selecting any components to be streamed over the IHDN. A utilization of this functionality is selecting a transport stream to filter some MPEG-2 sections using the section filter functionality provided in clause A.4. This functionality is provided in the DVB Tuner Extension discussed in clause A.2. As this action will not provide any services or components in the MPEG2-pTS, such a selection shall be treated as a Remove on the current transport stream with no new selections.

The next table indicates the result of selecting and removing different elements from the transport stream:

Table 7: Partial transport stream changes as a result of Tuner::SelectService

Locator type	Remove/add	result
Transport Stream	Add	Only possible as replacement of an empty MPEG2-TS. All services will be added to the MPEG2-pTS.
	Remove	All information from this MPEG2-TS is removed.
Service (note 1)	Add	Add PMT, all packets containing PCR value from PCR_PID and all data on the PIDs of all components of this service.
	Remove	Remove PMT, all packets containing the PCR from the PCR_PID and all data on the PIDs of all components of this service (note 2).
Component	Add	Add component to the PMT of the accompanying service; add all packets containing a PCR value from PCR_PID and all data on the PID of this component.
	Remove	Remove the component. All packets on the PID of this component are removed and the component is removed from the PMT of the accompanying service. When no components are left in the service, the PCR packets and the PMT are also removed (note 2).

NOTE 1: Adding a service can only be done if the service belongs to the same transport stream, as a tuner can only be tuned to one transport stream at a time.
NOTE 2: The components or PCR are only removed when no other service present in this partial transport stream refers to them.

The SI in the resulting transport stream depends on the chosen stream type (MPEG2-pTS or MPEG2-TS) and is defined by DVB. The ECMs, of the components added to the multiplex, will also be send.

When creating a partial transport stream, the PSI information (PAT and PMT) is changed to reflect only those parts of the MPEG2-TS actual present. The DVB-SI extension allows access to the complete PMT as present in the incoming transport stream. To prevent inconsistencies between these two PMTs, the PID of the components in the partial transport stream shall be the same as in the incoming transport stream.

A.1.5 Resource usage and Bandwidth allocation

A.1.5.1 Primary and secondary users

Within HAVi, resources like a tuner are governed by a resource mechanism. This allows an application to claim and release a tuner resource.

Access to a resource can happen on three different levels: unclaimed, secondary resource user and primary resource user. At any time, there can be at most one primary user and zero or more secondary users. When the tuner resource is unclaimed, all resource-guarded methods are open for use by any application.

A primary user has access to all resource-guarded methods. The secondary resource user has access to the resource-guarded methods but with limited functionality. The actions of a secondary resource owner may not hamper the primary resource user or other secondary resource users.

Three different possibilities can be distinguished regarding primary and secondary users within a DVB tuner FCM:

1) Only a primary user is allowed.

2) Secondary users are allowed, but all users (primary and secondary) use different output plugs.

3) Secondary users are allowed, and different users can use the same output plug.

In the primary-user-only case, the primary user has access to all resource-guarded methods, without any restriction in functionality. The DVB tuner FCM shall use the normal HAVi mechanisms (`FCM::GetReservationStatus` and `FCM::Reserve`) to indicate that only a primary user is allowed.

In the case that secondary users are allowed, but all users use different output plugs of the DVB tuner FCM, the following actions are allowed by the various users (see Table 8).

Table 8

Action	allowed by
Change transport stream selection	P
Replace current services by another service	PS (note 1)
Append components and services within the current MPEG2-TS	PS (note 1)
Remove components and services (from the current MPEG2-TS) added by the same resource user	PS (note 1)
Remove components and services (from current MPEG2-TS) added by other resource users	P (note 2)
Retrieve the complete list of the currently selected "services"	All
NOTE 1: This action is only allowed for the plug that the respective user is using.	
NOTE 2: This is only allowed in an implicit way, in the case that the primary resource user changes the transport stream.	

Table 8: Allowed actions in the case that secondary users are allowed, but all users use different output plugs. In this table, P stands for Primary and S for Secondary resource user.

If a `Tuner::SelectService` is called by a user, indicating a plug that is in use by another user, the DVB tuner FCM shall return the `Tuner::ENOT_COMPAT` error.

In the situation that secondary users are allowed, and that different users can use the same output plug of the DVB tuner FCM, the actions mentioned in Table 9 are allowed.

Table 9

Action	Allowed by
Change transport stream selection	P
Replace current services by another service	PS (note 1)
Append components and services within the current MPEG2-TS	PS
Remove components and services (from the current MPEG2-TS) added by the same resource user	PS (note 1)
Remove components and services (from the current MPEG2-TS) added by other resource users	P (note 2)
Retrieve the complete list of the currently selected "services"	All
NOTE 1: A service or component can be dropped only when all users requesting that information and using the same plug, indicate the information is no longer required.	
NOTE 2: This is only allowed in an implicit way, in the case that the primary resource user changes the transport stream.	

Table 9: Allowed actions in the case that secondary users are allowed and that different users can use the same output plug. In this table, P stands for Primary, S for secondary resource user.

A.1.5.2 Bandwidth allocation

When streaming information from the tuner to another device, a certain amount of bandwidth needs to be claimed on 1394. Within HAVi, the various mechanisms to allocate bandwidth are described.

In the situation of static bandwidth allocation and two (or more) resource users that use the same output plug, changing the contents of the output stream may result in insufficient bandwidth on the corresponding channel (even after issuing the `BandwidthRequirementChanged` event, after which the stream manager may attempt to claim more bandwidth). In the situation that the resource user that tried to change the stream contents is the primary one, the tuner will remove stream components from secondary users on the same plug until the stream contents desired by the primary resource user fit in the available bandwidth. If this is not possible, the DVB tuner FCM shall post a `ConnectionChanged` event, with reason `BandwidthAdaptationFailure.`

In the case, that bandwidth allocation is dynamic (see [16], section 5.9.5.3.2), and the currently selected services and possibly any new services require more bandwidth than is currently reserved, the system will attempt to claim more bandwidth. When no new bandwidth can be claimed, a mechanism similar to the static bandwidth allocation will be executed.

A.1.6 Service Lists and provided information

A.1.6.1 Definition of the different Lists

The Tuner FCM provides one or more lists of the available "services". Within the context of the Tuner FCM, a "service" "*encompasses both multiplexes of broadcast programs and components of broadcast programs*" (see [16]).

Three different "service"-lists are used, `USER_ASSOCIATED_NAMES`, `PROVIDER_ASSOCIATED_NAMES` and `TUNER_ASSOCIATED_NAMES`. The DVB interpretation of the Tuner FCM only relates to the lists of the type `PROVIDER_ASSOCIATED_NAMES` because they are defined by the provider of the information. The origin of the names in the list is the DVB-SI information. The other two list types are free to be used by the manufacturer and user.

All entries in any of the DVB defined "service"-lists shall contain "services" actually accessible to the tuner (including scrambled services). The tuner has to check each list entry in the SI to find out if this is possible.

The DVB tuner can provide the following types of `PROVIDER_ASSOCIATED_NAMES` service lists:

- **Service list.** This list contains all accessible DVB-services (services found in SDT-actual of all transport streams accessible by the tuner).
- **Transport stream list.** This list contains the different transport streams accessible by the tuner.
- **Bouquet lists.** A list for a bouquet defined in a transport stream accessible by the tuner, holding each DVB-service in that bouquet. Each such DVB-service in the list should be accessible by the tuner.
- **Network lists.** This list contains per network a list with the to the tuner accessible transport streams of that network.

Each DVB-Tuner shall implement a Service list. A list of transport streams is mandatory if the tuner supports redirecting of complete transport streams. The use of bouquet lists and network lists is optional.

The order of the lists returned by the method `GetServiceListInfo` shall be:

List index	List type
1	Service list
2	Transport stream list
3 - n	Bouquet lists (see note)
n+1 - m	Network lists
m+1 - ?	Possible other lists (e.g. user defined lists)
NOTE: When Transport Stream lists are not supported, the first bouquet list is list 2.	

In clause A.2 some utility methods are defined to retrieve the list indexes for bouquet-lists and network-lists.

A.1.6.2 Definition of the service List

The definition of the `ServiceListInfo` struct for the Service List is:

`title`	not defined
`type`	`PROVIDER_ASSOCIATED_NAMES`
`NumEntries`	Conform HAVi definition
`sizeHint`	Conform HAVi definition
`userOrdered`	False

The `Service` structure for the "services" from the Service List is defined in the following way:

`Locator`	See clause A.1.2.1, type DVB Service (numbers 10-13)
`Type`	Multiplex/DVB_PTS (see clause A.1.3.1)
`Language`	See clause A.1.2.3
`Name`	The service_name of the service as defined in the service_descriptor or multilangual_service_descriptor of the SDT.

A.1.6.3 Definition of the transport stream list

This list defines the following settings for the `ServiceListInfo` struct:

`Title`	not defined
`Type`	`PROVIDER_ASSOCIATED_NAMES`
`NumEntries`	Conform HAVi definition
`SizeHint`	Conform HAVi definition
`UserOrdered`	False

The `Service` structure for the "services" from the Transport stream list is defined in the following way:

`Locator`	See clause A.1.2.1, type Transport Stream (number 0 and 1)
`Type`	Multiplex/DVB, see clause A.1.3.1
`Language`	Empty
`Name`	Not defined

Each tuner has only one transport stream list.

A.1.6.4 Definition of the bouquet lists

These lists define the following settings for the `ServiceListInfo` struct:

`Title`	The bouquet name as defined in the BAT.
`Type`	`PROVIDER_ASSOCIATED_NAMES`
`NumEntries`	Conform HAVi definition
`SizeHint`	Conform HAVi definition
`userOrdered`	False

These lists provide a list of DVB-services in the corresponding bouquet and uses the same definition for the `Service` struct as the defined by the service list (see clause A.1.6.2). Each tuner can have several bouquet lists, one for each bouquet it has access to.

A.1.6.5 Definition of the network lists

These lists define the following settings for the `ServiceListInfo` struct:

`Title`	The network name as defined in the NIT.
`Type`	`PROVIDER_ASSOCIATED_NAMES`
`NumEntries`	Conform HAVi definition
`SizeHint`	Conform HAVi definition
`userOrdered`	False

These lists provide, per DVB network, a list of transport streams in that network, and use the same settings for the `Service` struct as the one defined by the transport stream list (see clause A.1.6.3).

A.1.6.6 GetServiceComponents and DVB ServiceLocators

The HAVi tuner specification provides some basic functionality to access the components of the "services". This functionality can be used to browse the DVB-SI tree, i.e. go from network to transport streams to services. Furthermore, it is also possible to access the different events associated with a "service".

The HAVi specification uses a special object called `ServiceLocator` to uniquely identify a "service". By defining a DVB specific format for this, an application can access specific "services" directly without using the "service"-lists. Furthermore, such a specific locator format provides the possibility to access other elements from transport streams than real services/service components.

When GetServiceComponents is called on a network, and the Tuner FCM supports networks, the result is a list of ServiceLocator indicating the transport streams of the network. When GetServiceComponents is called on a bouquet, and the Tuner FCM supports bouquets, the list of the services of this bouquet is returned. When GetServiceComponents is called on a transport stream, the result is a list of ServiceLocator indicating the DVB-services in the transport stream. When GetServiceComponents is called on a DVB-service, a list of the components of this DVB-service is returned.

The GetServiceEvents method shall always be supported for the current and next event. Support for other events is optional. This means that the Tuner capabilities GET_SERVICE_EVENTS_CURRENT_NEXT and GET_SERVICE_COMPONENTS are always supported.

In the situation that the GetServiceEvents method is called on a transport stream, bouquet, or network the DVB tuner FCM shall return the TUNER::EUNAVAILABLE error code.

A.1.6.6.1 Service struct when used as component

The Service structure when representing a component is defined in the following way:

Locator	See clause A.1.2.1, type DVB-Component (numbers 20 – 23)
Type	Multiplex/DVB, DigitalVideo/…, DigitalAudio/…, see clause A.1.3.1
Language	When the component represents audio, teletext, RBG or another language related data type the language of the component may be indicated. Otherwise, this field shall be empty.
Name	The description of the component as indicated in the component description or a manufacturer defined name.

A.1.6.6.2 Service Events

This list defines the following settings for the ServiceEvent struct:

type	When representing a service holding audio or video this shall be PROGRAM_EVENT. When only data streams are presents, DATA_EVENT is chosen. In any other case, UNKNOWN_EVENT shall be indicated.
EventName	The event_name of the event as defined in the short_event_descriptor or multilangual_event_descriptor of the EIT.
StartTime	The start time as indicated in the EIT, converted to the local time of the device.
StopTime	The stop time of the event calculated from the StartTime and the duration (as indicated in the EIT), converted to the local time of the device.
EventDescription	The description as found in the short_event_descriptor or multilingual_event_descriptor without emphasis marks or line feeds.

A.1.6.7 Representation of NVOD services

In DVB, a NVOD service is represented by a set of services described in one or more SDTs. One of the services of an NVOD service holds the SI information representing the information being broadcast. The other services hold special descriptors, indicating this base service. The added information provided by these additional services is the start and end-time of this specific instance of the service. The basic HAVi Tuner FCM shall indicate each NVOD service with the information provided by the base service in the service list and possibly in a bouquet list. So in the case of an NVOD service holding 8 time shifted services, all 8 will be represented using the information from the base NVOD service.

To provide a coherent interface to the user, the implementation of the DVB-RG should present these services in the time shifted order.

A.2 DVB-Tuner Extension

The FCM extension described in this clause is the central point of the DVB-Tuner FCM extensions and a DVB residential gateway shall always provide a DVB-Tuner extension for each HAVi Tuner FCM.

It provides the link to the actual HAVi Tuner FCM and to the section filter and DVB-SI extensions.

Furthermore, some basic utility methods and a method allowing the selecting of the transport stream source of a tuner without selecting any components to be streamed over the IHDN are provided. This functionality is required by the MHP tuner API. A utilization of this functionality is selecting a transport stream to filter some MPEG-2 sections using the section filter functionality provided in clause A.4.

A.2.1 Registry attributes

Registry Attribute Name	IDL Type	Value
ATT_VENDOR_ID	VendorId	DVB vendor id
ATT_SE_TYPE	SoftwareElementType	0x8000 0000

A DVB-Tuner Extension FCM shall register with at least these two attributes.

A.2.2 Services

Service	Comm Type	Locality	Access	Resv Prot
DvbTunerExt::GetBaseFcmSeid	M	global	all	
DvbTunerExt::GetDvbSiSeid	M	global	all	
DvbTunerExt::GetSfSeid	M	global	all	
DvbTunerExt::SelectSourceTs	M	global	all	Yes
DvbTunerExt::GetBouquetServiceLists	M	global	all	
DvbTunerExt::GetNetworkServiceLists	M	global	all	

NOTE: Grey tinted methods are optional.

A.2.3 API definition

DvbTunerExt::GetBaseFcmSeid

Prototype

```
Status DvbTunerExt::GetBaseFcmSeid(
      out SEID baseFcmSeid)
```

Parameters

- `baseFcmSeid` – the SEID of the Tuner FCM this is an extension on.

Description

This method returns the software element identifier of the HAVi Tuner FCM extended by the services of clause A.2.2.

DvbTunerExt::GetDvbSiSeid

Prototype

```
Status DvbTunerExt::GetDvbSiSeid(
      out SEID dvbSiSeid)
```

Parameters

- `dvbSiSeid` – the SEID of the SI extension of this Tuner FCM.

Description

This method returns the software element identifier of the SI extension of the DVB Tuner described by this extension.

Error codes

- `ENOTIMPLEMENTED` – the tuner does not implement a DVB-SI extension.

DvbTunerExt::GetDvbSfSeid

Prototype

```
Status DvbTunerExt::GetDvbSfSeid(
      out SEID dvbSfSeid)
```

Parameters

- `dvbSfSeid` – the SEID of the section filtering extension of this Tuner FCM.

Description

This method returns the software element identifier of the section filter extension of the DVB Tuner described by this extension.

Error codes

- `ENOTIMPLEMENTED` – the tuner does not implement a DVB-SI extension.

DvbTunerExt::SelectSourceTs

Prototype

```
Status DvbTunerExt::SelectSourceTs(
      in ServiceLocator ts )
```

Parameters

- `ts` – the ServiceLocator representing the transport stream that the tuner, connected to the HAVi Tuner FCM, is tuned to.

Description

This method changes the MPEG2-TS selection of the HAVi Tuner FCM for this DVB Tuner FCM Extension. Selecting an MPEG2-TS will not generate any streaming on the output plugs. Only the MPEG2-TS source is selected as a preparation for SI filtering and/or section filtering. The method allows the Section Filter and DVB-SI Extension to access different transport streams without streaming information over the HLN.

Selecting a transport stream will stop the streaming of any MPEG2-TS components of aMPEG2-TS.

This method is mandatory for all DVB Tuner FCMs that also implement the Section Filter or DVB-SI Extension.

Error codes

- `ENOTIMPLEMENTED` – the SelectSourceTs method is not supported by the DVB Tuner FCM.
- `DvbTunerExt::ELOCATOR` – if the Tuner FCM cannot resolve the locator.

DvbTunerExt::GetBouquetServiceLists

Prototype

```
Status DvbTunerExt::GetBouquetServiceLists(
      out ushort startListNumber,
      out ushort endListNumber)
```

Parameters

- `startListNumber` – the list number of the first bouquet service list.
- `endListNumber` – the list number of the last bouquet service list.

Description

This method returns the start and end list number of the service lists representing bouquets known to the tuner FCM (`Tuner::GetServiceList` and `Tuner::GetServiceListInfo`).

Error codes

- `ENOTIMPLEMENTED` – the HAVi Tuner FCM does not support bouquet lists
- `DvbTunerExt::ELIST` – no bouquets are known to the tuner.

DvbTunerExt::GetNetworkServiceLists

Prototype

```
Status DvbTunerExt::GetNetworkServiceLists(
      out ushort startListNumber,
      out ushort endListNumber)
```

Parameters

- `startListNumber` – the list number of the first network service list.
- `endListNumber` – the list number of the last network service list.

Description

This method returns the start and end list number of the service lists representing networks known to the tuner FCM (`Tuner::GetServiceList` and `Tuner::GetServiceListInfo`).

Error codes

- `ENOTIMPLEMENTED` – the HAVi Tuner FCM does not support network lists
- `DvbTunerExt::ELIST` – no networks are known to the tuner.

A.3 DVB-SI Extension

This Tuner FCM extension describe the additional methods to access the DVB-SI information. The information provided by these methods is intended to be used by EPGs and to support networked MHP applications. Because of this requirement, the structure and data elements are similar to the SI API present in the MHP specification (the org.dvb.si package).

The application can request lists of service locators indicating the different SI elements (service, bouquet, network, event) present in the network. Using dedicated methods, the DVB-SI specific information can be accessed.

The DVB-SI specific information for each SI element is expressed in structures, one for each SI element. These structures contain basic information When the application requests descriptor access, it has the opportunity to indicate the descriptor tags of the required descriptors. These descriptors are then extracted and added to the provided information. The descriptors shall be presented in the same order as they were present in the original SI table. The API does not support all DVB-SI tables, more specific, the TSDT, TOT, TDT and the transport stream sub-loops of the BAT and NIT are not supported. When the application requests information from these tables, the section filter API should be used.

Some of the calls of this API require information to be filtered from the incoming stream. This can take up to a 30 s. When the calls to these methods are handled in a synchronous manner, such behaviour could block message processing for an unacceptable time. This is why the application should use the asynchronous version of the HAVi calls in this set-up.

A.3.1 Registry attributes

Registry Attribute Name	IDL Type	Value
`ATT_VENDOR_ID`	`VendorId`	`DVB vendor id`
`ATT_SE_TYPE`	`SoftwareElementType`	`0x8000 0001`

A DVB Tuner supporting the DVB-SI Extension shall register with at least these two attributes.

A.3.2 Services

Service	Comm Type	Locality	Access	Resv Prot
DvbSiExt::GetBaseFcmSeid	M	global	all	
DvbSiExt::RetrieveNetworks	M	global	all	
DvbSiExt::RetrieveActualNetwork	M	global	all	
DvbSiExt::RetrieveBouquets	M	global	all	
DvbSiExt::RetrieveActualTransportStream	M	global	all	
DvbSiExt::RetrieveServices	M	global	all	
DvbSiExt::RetrieveActualServices	M	global	all	
DvbSiExt::RetrievePresentEvent	M	global	all	
DvbSiExt::RetrieveFollowingEvent	M	global	all	
DvbSiExt::RetrieveScheduledEvents	M	global	all	
DvbSiExt::RetrieveDvbNetworkInfo	M	global	all	
DvbSiExt::RetrieveDvbBouquetInfo	M	global	all	
DvbSiExt::RetrieveDvbServiceInfo	M	global	all	
DvbSiExt::RetrieveDvbProgramInfo	M	global	all	
DvbSiExt::RetrieveDvbEventInfo	M	global	all	
DvbSiExt::MonitorStart	M	global	all	
DvbSiExt::MonitorStop	M	global	all	
<Client>::MonitorNotification	MB	global	all	

A.3.3 Data structures

DvbDataOrigin

```
struct DvbDataOrigin {
   boolean        fromActual;
   boolean        fromCache;
   DateTime       updateTime;
   ServiceLocator sourceTs;
};
```

Description

Element	Description
`fromActual`	`True` when the information was filtered from an "actual" table or from a table with no "actual/other" distinction. False otherwise.
`fromCache`	`True` whether the information is returned from cache.
`updateTime`	The time (local time) when the information was last updated.
`sourceTs`	A `serviceLocator` representing the transport stream (type 0, including network id) the information was filtered from.

This structure gives information on the origin of a DVB data element.

DvbDescriptor

```
struct DvbDescriptor {
      octet               descriptorTag
      octet               descriptorLength
      sequence<octet,254> descriptorData;
};
```

Description

Element	Description
descriptorTag	The descriptor tag of this descriptor.
descriptorLength	The descriptor length of this descriptor.
descriptorData	The data of this descriptor (everything after the length field).

This structure represents a descriptor as present in DVB and MPEG.

DvbString

```
typedef sequence<octet, 254> DvbString;
```

Description

Within DVB strings are coded using a DVB specific method (see [15]). This structure represents such a string. The SI access part of the Tuner FCM provides this information to allow the most generic access to the DVB information.

DvbContentNibbles

```
typedef struct DvbContentNibble
{  octet contentNibble;
   octet userNibble;
};
```

Description

The content nibbles related to an event.

DvbRetrieveMode

```
enum DvbRetrieveMode {
   FROM_CACHE_ONLY,
   FROM_CACHE_OR_STREAM,
   FROM_STREAM_ONLY
};
```

Description

Retrieve mode parameter of the retrieve methods of the Tuner FCM.

Element	Description
FROM_CACHE_ONLY	When FROM_CACHE_ONLY mode is specified, the data will be retrieved only if it is in the cache. Otherwise, DvbSiExt::ENOT_IN_CACHE will be returned. No stream access is done in this case.
FROM_CACHE_OR_STREAM	When FROM_CACHE_OR_STREAM mode is specified, the data will be retrieved from cache if it is present in the cache, otherwise it will be retrieved from the stream.
FROM_STREAM_ONLY	When FROM_STREAM_ONLY mode is specified, the data will be retrieved directly from the stream and no cache access is tried first. This mode is meaningful if the application knows that the information is not in the cache or that the information in the cache is no longer valid. This for instance the case if the application has got the notification of the existence of an updated version through the notification mechanism.

DvbRunningStatus

```
enum DvbRunningStatus   {
   UNDEFINED,               // =0
   NOT_RUNNING,             // =1
   STARTS_IN_A_FEW_SECONDS, // =2
   RUNNING,                 // =3
   PAUSING,                 // =4
};
```

Description

The running status of a service or event as defined by DVB.

DvbEsInfo

```
struct DvbEsInfo {
   short                    componentTag;
   ushort                   elementaryPid;
   ushort                   streamType;
   sequence<DvbDescriptor>  esDescriptorList;
};
```

Description

Description of an Elementary Stream.

Element	Description
componentTag	The component tag of this elementary stream. If no component tag is indicated, -2 is returned.
elementaryPid	The PID of this elementary stream (clause A.1.4).
streamType	The MPEG2 stream type of this elementary stream as indicated in the PMT.
esDescriptorList	The list of indicated ES descriptors. The descriptors in the descriptor list shall be presented in the same order as they were present in the original SI table.

DvbMonitorResult

```
enum DvbMonitorResult {
   ELEMENT_CHANGED,
   ELEMENT_KILLED
};
```

Description

Element	Description
ELEMENT_CHANGED	The table containing the monitored SI element has changed.
ELEMENT_KILLED	The monitored SI element is no longer present and monitoring of this element has stopped.

A.3.4 API definition

DvbSiExt::GetBaseFcmSeid

Prototype

```
Status DvbSiExt::GetBaseFcmSeid(
      out SEID baseFcmSeid)
```

Parameters

- `baseFcmSeid` – the SEID of the HAVi Tuner FCM this is an extension on.

Description

This method returns the software element identifier of the HAVi Tuner FCM extended by the services of clause A.3.2.

DvbSiExt::RetrieveNetworks

Prototype

```
Status DvbSiExt::RetrieveNetworks(
      in    DvbRetrieveMode           retrieveMode,
      out   sequence<ServiceLocator>  networkList )
```

Parameters

- `retrieveMode` - the retrieve mode to be used for this request.
- `networkList` – the list of networks or the requested network. The safe parameter size limit is 200 entries.

Description

This method provides `ServiceLocators` for DVB networks. Each returned `ServiceLocator` represents a DVB-Network.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveActualNetwork

Prototype

```
Status DvbSiExt::RetrieveActualNetwork(
      in    DvbRetrieveMode    retrieveMode,
      out   ServiceLocator     serviceLocator )
```

Parameters

- `retrieveMode` - the retrieve mode to be used for this request.
- `serviceLocator` – the ServiceLocator of the actual network.

Description

Retrieve information associated with the actual network. The actual network is the network carrying the transport stream currently selected by the network interface connected to this SI database. The returned `ServiceLocator` will represent a DVB-Network.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveActualTransportStream

Prototype

```
Status DvbSiExt::RetrieveActualTransportStream(
   in    DvbRetrieveMode  retrieveMode,
   out   ServiceLocator   actualTs )
```

Parameters

- `retrieveMode` - the retrieve mode to be used for this request.
- `actualTs` – a service locator representing the transport stream currently selected by the Tuner FCM.

Description

Retrieve the current transport stream selection. The returned `ServiceLocator` represents a DVB-Transport stream of locator type 0.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveBouquets

Prototype

```
Status DvbSiExt::RetrieveBouquets(
   in    DvbRetrieveMode           retrieveMode,
   out   sequence<ServiceLocator>  bouquetList )
```

Parameters

- `retrieveMode` - the retrieve mode to be used for this request.
- `bouquetList` – the list of bouquets requested. The safe parameter size limit is 1 000 entries.

Description

Retrieve the list of bouquets. Each returned `ServiceLocator` represents a DVB-Bouquet.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveServices

Prototype

```
Status DvbSiExt::RetrieveServices(
   in    int                       originalNetworkId,
   in    int                       transportStreamId,
   in    int                       serviceId,
   in    DvbRetrieveMode           retrieveMode
   out   sequence<ServiceLocator>  serviceList )
```

Parameters

- `originalNetworkId` – the original network id of the requested services, -1 indicates "any".
- `transportStreamId` – the transport stream id of the requested services, -1 indicates "any".
- `serviceId` – the service id of the requested services, -1 indicates "any".
- `retrieveMode` - the retrieve mode to be used for this request.
- `serviceLocatorList` – the list of requested services. The safe parameter size limit of this sequence is 3000.

Description

Retrieve list of services. The required services can be specified by their identification (-1 means "any"). The returned `ServiceLocator`s shall contain DVB-Services of type 10 or 12.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveActualServices

Prototype

```
Status DvbSiExt::RetrieveActualServices(
   in    DvbRetrieveMode           retrieveMode,
   out   sequence<ServiceLocator>  serviceList )
```

Parameters

- `retrieveMode` - the retrieve mode to be used for this request.
- `serviceList` – the ServiceLocators of the services in the transport stream currently selected by the Tuner FCM. The safe parameter size limit of this sequence is 1 000.

Description

This method provides the `ServiceLocators` of the services of the actual transport stream. The returned `ServiceLocators` shall contain DVB-Services of type 10 or 12.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrievePresentEvent

Prototype

```
Status DvbSiExt::RetrievePresentEvent(
   in       ServiceLocator     service,
   in       DvbRetrieveMode    retrieveMode,
   out      ServiceLocator     presentEvent )
```

Parameters

- `service` – the ServiceLocator of the service the present event is requested from.
- `retrieveMode` - the retrieve mode to be used for this request.
- `presentEvent` – the ServiceLocator of the present event of the indicated service.

Description

This method provides the ServiceLocator of the present event of the indicated service. The returned ServiceLocator shall represent a DVB-Event (type 60 or 62).

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.

- DvbSiExt::ETABLE_NOT_FOUND - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveFollowingEvent

Prototype

```
Status DvbSiExt::RetrieveFollowingEvent(
   in       ServiceLocator     service,
   in       DvbRetrieveMode    retrieveMode,
   out      ServiceLocator     followingEvent )
```

Parameters

- service – the ServiceLocator of the service the following event is requested from.
- retrieveMode - the retrieve mode to be used for this request.
- presentEvent – the ServiceLocator of the following event of the indicated service.

Description

This method provides the ServiceLocator of the following event of the indicated service. The returned ServiceLocator shall represent a DVB-Event (type 60 or 62).

Error codes

- ERESOURCE_LIMIT - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- DvbSiExt::ENOT_IN_CACHE - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- DvbSiExt::ENOT_IN_TABLE - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- DvbSiExt::ETABLE_NOT_FOUND - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveScheduledEvents

Prototype

```
Status DvbSiExt::RetrieveScheduledEvents(
   in    ServiceLocator            service,
   in    DateTime                  startOfPeriod,
   in    DateTime                  endOfPeriod,
   in    DvbRetrieveMode           retrieveMode,
   out   sequence<ServiceLocator>  scheduledEvents )
```

Parameters

- service – the ServiceLocator of the service the events are requested from.
- startOfPeriod – the start of the period the events are requested for (local time).
- endOfPeriod – the end of the period the events are requested for (local time).
- retrieveMode - the retrieve mode to be used for this request.
- scheduledEvents – the ServiceLocator of the events within the indicated timeframe. The safe parameter size limit for this sequence is 3000.

Description

This method provides the ServiceLocators of the events of this indicated service within the time frame startOfPeriod - endOfPeriod. The events are presented in the order they are present in the EIT-schedule. The returned ServiceLocator shall represent a DVB-Event (type 60 or 62).

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::EPERIOD` – incorrect period specification.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveDvbNetworkInfo

Prototype

```
Status DvbSiExt::RetrieveDvbNetworkInfo(
   in    ServiceLocator            network,
   in    octet[3]                  preferredLanguage,
   in    sequence<octet>           descriptorHints,
   in    DvbRetrieveMode           retrieveMode,
   out   DvbString                 networkName,
   out   octet[3]                  language,
   out   int                       serviceListNumber,
   out   sequence<ServiceLocator>  transportStreamList,
   out   sequence<DvbDescriptor>   descriptorList,
   out   DvbDataOrigin             dataOrigin )
```

Parameters

- `network` – the locator whose information is requested (locator type 40).
- preferredLanguage - this field contains an ISO-639 three-character code representing the preferred language in which the network name should be returned (under condition it is available in the SI tables). Value 0x000000 means that the network name is not requested in any particular language. If a particular language is requested, and if the available languages do not correspond to it, it is up to the tuner to select a language, for instance on the basis of a preferred language set in the tuner.
- `descriptorHints` – a list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If this list is empty, the application is not interested in descriptors. All non-applicable tag values are ignored.
- `retrieveMode` - the retrieve mode to be used for this request.
- networkName - the name of the network. The name is extracted from the network_name_descriptor or optionally from the multilingual_network_name_descriptor. If none of these descriptors are present an empty sequence is returned in the structure.
- language - this field contains an ISO-639 three-character code representing the language in which the network name is provided. This parameter has no meaning if the networkName parameter is empty.
- serviceListIndex - the number of the service list representing this network. (The listNumber field of the Tuner::GetServiceList method.) When the HAVi Tuner FCM does not support service lists representing networks, the field has the value -1.
- transportStreamList - the list of locators representing the transport streams present in this network. All locators shall represent a DVB transport stream locator type 0. The safe parameter limit for this method is 1 000.
- descriptorList - the list of requested descriptors. The descriptors in the descriptor list shall be presented in the same order as they were present in the original SI table. The safe parameter limit for this method is 100.

- dataOrigin - the data origin parameters of this call.

Description

This method accesses some detailed SI specific information regarding the DVB-Network.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ELOCATOR` – The locator could not be resolved.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveDvbBouquetInfo

Prototype

```
Status DvbSiExt::RetrieveDvbBouquetInfo(
   in    ServiceLocator            locator,
   in    octet[3]                  preferredLanguage,
   in    sequence<octet>           descriptorHints,
   in    DvbRetrieveMode           retrieveMode,
   out   DvbString                 bouquetName,
   out   octet[3]                  language,
   out   ServiceLocator            locator,
   out   int                       serviceListIndex,
   out   sequence<ServiceLocator>  bouquetServiceList,
   out   sequence<DvbDescriptor>   descriptorList,
   out   DvbDataOrigin             dataOrigin )
```

Parameters

- `locator` – the locator whose information is requested (locator type 50).
- preferredLanguage - this field contains an ISO-639 three-character code representing the language in which the bouquet name should be returned (under condition it is available in the SI tables). Value 0x000000 means that the bouquet name is not requested in any particular language. If a particular language is requested, and if the available languages do not correspond to it, it is up to the tuner to select a language, for instance on the basis of a preferred language set in the tuner.
  `descriptorHints` – a list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If this list is empty, the application is not interested in descriptors. All non applicable tag values are ignored.
- `retrieveMode` - the retrieve mode to be used for this request.
- bouquetName - the name of the bouquet. The name is extracted from the bouquet_name_descriptor or optionally from the multilingual_bouquet_name_descriptor. When this information is not available an empty sequence is returned.
- language - this field contains an ISO-639 three character code representing the language of the bouquet name. This parameter has no meaning if the bouquetName parameter is empty.
- serviceListIndex - the number of the service list representing this bouquet (the listNumber field of the Tuner::GetServiceList method). When the HAVi Tuner FCM does not support service lists representing bouquets, the field has the value -1.
- bouquetServiceList - the list of locators representing the services of this bouquet. All locators shall represent a DVB service, locator type 12. The safe parameter limit for this method is 100.
- descriptorList - the list of requested descriptors. The descriptors in the descriptor list shall be presented in the same order as they were present in the original SI table. The safe parameter limit for this method is 100.

- dataOrigin - the data origin parameters of this call.

Description

This method accesses some detailed SI specific information regarding the DVB-Bouquet.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ELOCATOR` – The locator could not be resolved.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveDvbServiceInfo

Prototype

```
Status DvbSiExt::RetrieveDvbServiceInfo(
   in    ServiceLocator          locator,
   in    octet[3]                preferredLanguage,
   in    sequence<octet>         descriptorHints,
   in    DvbRetrieveMode         retrieveMode,
   out   DvbString               serviceName,
   out   DvbString               providerName,
   out   octet[3]                language,
   out   ushort                  serviceType,
   out   boolean                 hasPresentFollowing,
   out   boolean                 hasSchedule,
   out   boolean                 freeCaMode,
   out   DvbRunningStatus        runningStatus,
   out   sequence<DvbDescriptor> descriptorList,
   out   DvbDataOrigin           dataOrigin )
```

Parameters

- `locator` – the locator whose information is requested (locator type 10-13).
- preferredLanguage - this field contains an ISO-639 three-character code representing the language in which the service name and provider name should be returned (under condition it is available in the SI tables). Value 0x000000 means that this information is not requested in any particular language. If a particular language is requested, and if the available languages do not correspond to it, it is up to the tuner to select a language, for instance on the basis of a preferred language set in the tuner. `descriptorHints` – a list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If this list is empty, the application is not interested in descriptors. All none applicable tag values are ignored.
- `retrieveMode` - the retrieve mode to be used for this request.
- serviceName - the name of the service represented by this service. The name is extracted from the service_descriptor or optionally from the multilingual_service_name_descriptor. If this descriptor is not present an empty sequence is returned in the structure.
- providerName - the name of the service provider of this service. The name is extracted from the service_descriptor or optionally from the multilingual_service_name_descriptor. If this descriptor is not present an empty sequence is returned in the structure.
- language - this field contains an ISO-639 three character code representing the language of the service and service provider name. This parameter has no meaning if serviceName and providerName parameters are empty.

- serviceType - the service type of this service as coded in the SDT.
- hasPresentFollowing - this field is true when the SDT of this service indicates that this service holds a EIT-present/following table.
- hasSchedule - this field is true when the SDT of this service holds a EIT-schedule table.
- freeCaMode - this field is true when the SDT of this service indicates that none of the components of this service is scrambled.
- runningStatus - the running status of this service as coded in the SDT.
- descriptorList - the list of requested descriptors. The descriptors in the descriptor list shall be presented in the same order as they were present in the original SI table. The safe parameter limit for this method is 100.
- dataOrigin - the data origin parameters of this call.

Description

This method provides detailed information regarding DVB-services.

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ELOCATOR` – The locator could not be resolved.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveDvbProgramInfo

Prototype

```
Status DvbSiExt::RetrieveDvbProgramInfo(
   in    ServiceLocator          locator,
   in    sequence<octet>         programDescriptorHints,
   in    sequence<octet>         esDescriptorHints,
   in    DvbRetrieveMode         retrieveMode,
   out   uint                    pcrPid,
   out   sequence<DvbDescriptor> programDescriptorList,
   out   sequence<DvbEsInfo>     esList,
   out   DvbDataOrigin           dataOrigin )
```

Parameters

- `locator` – the locator whose information is requested (locator type 10-13).
- `programDescriptorHints` – a list of hints for descriptors (identified by their tags) the application is interested in from the main descriptor loop of the PMT. If the array contains -1 as its one and only element, the application is interested in all descriptors. If this list is empty, the application is not interested in descriptors. All non applicable tag values are ignored.
- `programDescriptorHints` – a list of hints for descriptors (identified by their tags) the application is interested in from the elementary stream descriptor loops of the PMT. If the array contains -1 as its one and only element, the application is interested in all descriptors. If this list is empty, the application is not interested in descriptors. All non applicable tag values are ignored.
- `retrieveMode` - the retrieve mode to be used for this request.
- pcrPid - The PID of the transport stream packets holding the PCR of this program.
- descriptorList - the list of requested descriptors from the main descriptor loop. The descriptors in the descriptor list shall be presented in the same order as they were present in the original SI table.

- esList - the list of elementary stream descriptions. The safe parameter list size of this sequence is 100.
- dataOrigin - the data origin parameters of this call.

Description

This method provide detailed information regarding DVB-services. The total amount of information returned to the application can never exceed 1024 bytes (= maximal length of a PMT).

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ELOCATOR` – The locator could not be resolved.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::RetrieveDvbEventInfo

Prototype

```
Status DvbSiExt::RetrieveDvbEventInfo(
   in    ServiceLocator               locator,
   in    octet[3]                     preferredLanguage,
   in    sequence<octet>              descriptorHints,
   in    DvbRetrieveMode              retrieveMode,
   out   DvbString                    eventName,
   out   DvbString                    shortDescription,
   out   octet<3>                     language,
   out   DateTime                     startTime,
   out   ulong                        duration,
   out   boolean                      freeCaMode,
   out   ushort                       parentalRating,
   out   DvbRunningStatus             runningStatus,
   out   sequence<DvbContentNibble>   contentNibbles,
   out   sequence<DvbDescriptor>      descriptorList,
   out   DvbDataOrigin                dataOrigin )
```

Description

This method provides the DVB-SI information related to DVB networks described in the EIT schedule or present following.

Parameters

- `locator` – the locator whose information is requested.
- preferredLanguage - this field contains an ISO-639 three-character code representing the language in which the event is described and named (under condition it is available in the SI tables). Value 0x000000 means that the event name and description are not requested in any particular language. If a particular language is requested, and if the available languages do not correspond to it, it is up to the tuner to select a language, for instance on the basis of a preferred language set in the tuner. `descriptorHints` – a list of hints for descriptors (identified by their tags) the application is interested in. If the array contains -1 as its one and only element, the application is interested in all descriptors. If this list is empty, the application is not interested in descriptors. All non applicable tag values are ignored.
- `retrieveMode` - the retrieve mode to be used for this request.
- `eventName` – the name of the indicated event. The name is extracted from a short_event_descriptor. When this information is not available an empty sequence is returned..

- `shortDescription` – this method returns the short description of this event. The name is extracted from a short_event_descriptor. When this information is not available an empty sequence is returned.
- language - this field contains an ISO-639 three character code representing the language of the event name and short description. This parameter has no meaning if `eventName` and `shortDescription` parameters are empty.
- `startTime` – the start time of this event (local time).
- `duration` – the duration of this event in seconds.
- `freeCaMode` – if this field is true, the EIT indicates none of the components of this event are scrambled.
- `parentalRating` – the parental rating of this event in this country. This information is extracted from the parental rating descriptor. If no such descriptor is present, 0 is returned.
- `runningStatus` – the running status of this service as coded in the EIT.
- `contentNibbles` – the content nibbles related to the event. This information is extracted from the content_descriptor. If this descriptor is not present an empty list is returned. The safe parameter limit for this method is 128.
- `descriptorList` – the list of requested descriptors. The descriptors in the descriptor list shall be presented in the same order as they were present in the original SI table. The safe parameter limit for this method is 100.
- `dataOrigin` – the data origin parameters of this call.

Description

Access the DVB information of the indicated event (preference is given to the EIT present following).

Error codes

- `ERESOURCE_LIMIT` - the resources needed for retrieving the data are not available. This error is also returned when the information needs to be filtered and no transport stream is selected by the tuner.
- `DvbSiExt::ELOCATOR` – The locator could not be resolved.
- `DvbSiExt::ENOT_IN_CACHE` - the request was made with the FROM_CACHE_ONLY mode and the requested data is not present in the cache.
- `DvbSiExt::ENOT_IN_TABLE` - the DVB-SI table where the information about the requested information should be located has been retrieved but the requested information is not present in it. The reason may be that the object corresponding to the requested identifiers does not exist.
- `DvbSiExt::ETABLE_NOT_FOUND` - the DVB-SI table that should contain the requested information could not be retrieved. The reason may be that the requested table is not broadcasted in the transport stream currently associated with the Tuner FCM.

DvbSiExt::MonitorStart

Prototype

```
Status DvbSiExt::MonitorStart(
      in ServiceLocator   locator,
      in OperationCode    opCode)
```

Parameters

- `locator` – the locator whose information should be monitored.
- `opCode` – the `OperationCode` provided by the client. This value will be placed in the operation code of the notification message that is send to the client.

Description

Indicates the application requires to be notified of changes in the table holding the SI element represented by the ServiceLocator. A locator indicating a network results in monitoring the NIT, a locator representing a bouquet results in monitoring a BAT, a locator representing a transport stream results in monitoring an SDT and a locator representing a service results in monitoring of the EIT and possibly the RST.

The notification will be send when the residential gateway notices a change in the monitored table.

The subscription ends when the application unsubscribes (`MonitorStop`) explicitly or when the application is no longer reachable.

Error codes

- `ERESOURCE_LIMIT` – if the FCM was unable to allocate resources to register this indication listener.
- `DvbSiExt::ELOCATOR` – if the locator is not suitable or could not be resolved.

DvbSiExt::MonitorStop

Prototype

```
Status DvbSiExt::MonitorStop(
      in ServiceLocator locator)
```

Parameters

- `locator` – the locator that should no longer be monitored.

Description

Unsubscribes the caller as a software element that is interested in changes of the SI element indicated by the MonitorStart call. The FCM will stop sending notifications to the caller.

Error codes

- `DvbSiExt::ENO_NOT` – if no notification with this ID has been registered for the caller.

<Client>::MonitorNotification

Prototype

```
Status <Client>::MonitorNotification(
      in ServiceLocator     serviceLocator,
      in DvbMonitorResult   monitorResult )
```

Parameters

- `locator` – the locator whose info changed.
- `monitorResult` – indication of the notified change.

Description

During the subscription, "message back" is sent to the subscriber, each time a change occurs for the specified locator.

A.4 Section Filtering Extension

This clause describes the section filter extension for the DVB-Tuner FCM. This extension allows an application to filter MPEG-2 private sections (see [14]) from the incoming MPEG-2 transport stream.

A.4.1 Registry attributes

Registry Attribute Name	IDL Type	Value
ATT_VENDOR_ID	VendorId	DVB vendor id
ATT_SE_TYPE	SoftwareElementType	0x8000 0002

A DVB Tuner supporting the Section Filter Extension shall register with at least these two attributes.

A.4.2 Services

Service	Comm Type	Locality	Access	Resv Prot
DvbSfExt::GetBaseFcmSeid	M	Global	All	
DvbSfExt::ReserveSF	M	Global	All	
DvbSfExt::ReleaseSF	M	Global	All	
DvbSfExt:: StartFiltering	M	Global	All	
DvbSfExt:: StopFiltering	M	Global	All	
<Client>::SFNotification	MB	Global	All	
SFAvailable	E	Global	Tuner (all)	

A.4.3 Data structures

Section

```
Typedef sequence<octet, 4096> Section
```

Description

SectionFilterType

```
enum SectionFilterType {
        SIMPLE_SECTION_FILTER,
        CONTINUOUS_SECTION_FILTER,
        TABLE_SECTION_FILTER
};
```

Description

There are three types of section filters: Simple section filters, continuous section filters and table section filters. Simple section filters filter one section, according to the specified section filtering parameters. Continuous section filters filter all sections according to the specified filter parameters, until the section filter is stopped by the client application, or revoked for some other reason. If the residential gateway runs out of buffer space (as specified in the filter parameters), a message will be sent to the client. After the buffer problems have been solved, the section filter continues filtering. Table section filters filter all sections according to the specified filter parameters, until a complete table has been filtered, the table section filter has been stopped by the client or the table section filter has been revoked for some other reason.

If the client application receives a buffer overflow message from a section filter, this implies that the section that caused the overflow is lost. Furthermore, sections that have been filtered after the one causing the overflow may also be lost.

SectionFilterParameters

```
struct SectionFilterParameters {
      SectionFilterType     filterType;
      uint                  filtergranularity;
      serviceLocator        TSIdentifier;
      ushort                PID;
      uchar                 table_id;
      uint                  offset;
      sequence<octet, 20>   posFilterDef;
      sequence<octet, 20>   posFilterMask;
      sequence<octet, 20>   negFilterDef;
      sequence<octet, 20>   negFilterMask;
      uint                  bufferSize;
      long                  timeout;
};
```

Description

This structure specifies the section filter parameters for the three types of section filters. The filterType parameter specifies the type of the section filter. The filter granularity parameter specifies which of the following parameters is specified, i.e. on which filter parameters filtering will be performed:

Filter granularity	Filter parameters
1	PID
2	PID, table_id
3	PID, table_id, PosFilterDef, PosFilterMask
4	PID, table_id, Offset, PosFilterDef, PosFilterMask
5	PID, table_id, PosFilterDef, PosFilterMask, NegFilterDef, NegFilterMask
6	PID, table_id, Offset, PosFilterDef, PosFilterMask, NegFilterDef, NegFilterMask

`TSIdentifier` identifies the transport stream that this section filter should work on.

`PID` identifies the Packet Identifier from which the sections will be filtered.

`Table_id` specifies the table_id of the sections that will be filtered.

`posFilterDef` defines the value to match in the section. `posFilterMask` defines which bits in the section are to be compared against the value specified in the `posFilterDef` parameter, according to: `PosFilterDef` & `PosFilterMask` = Section & `PosFilterMask`.

The Negative Filter also uses mask and value parameters, but the Negative Filter triggers when: `NegFilterDef` & `NegFilterMask` ≠ Section & `NegFilterMask`.

In the situation that no offset is specified (filter granularity 3 and 5), the filtermasks (positive and negative) are applied at the section, starting from byte 3.

`offset` defines the offset within the section which the first byte of the `posFilterDef`, `posFilterMask`, `NegFilterDef` and `NegFilterMask` arrays is intended to match. The offset must be less than 31 as described in DAVIC [23] part 10, section 11.5.3. The offset must be equal to or greater than 3.

The `bufferSize` specifies the size of the section filter buffer in bytes (typically larger than 8 kB). If, at a certain moment, the buffer overflows, a notification is sent to the application that uses the section filter, indicating that a buffer overflow has occurred. The reception of this buffer overflow notification indicates to the client application that the section that caused the overflow is lost. Furthermore, sections following the one that caused the overflow may also be lost.

In the situation that the buffer size is smaller than the section size, each time a section is filtered and the buffer overflows, a notification is sent to the client. The client can conclude that the buffer size is too small because of the fact that this notification will contain an empty section. The client then has to stop the section filter, change the buffer size and start the section filter again.

In the case that the buffer size is larger than the section size, overflow will occur when the section filtered last does not fit into the buffer anymore. In that situation, section filtering will be stopped until at least one section has been transferred to the client.

The `timeout` value sets the time-out for this section filter, in milliseconds. When the time-out happens, the client application will be called, with a `SF_TIME_OUT` SectionFilterResult. Furthermore, the section filter stops. A SimpleSectionFilter will time out if no sections arrive within the specified period. A TableSectionFilter will time out if the complete table does not arrive within the specified time. A ContinousSectionFilter will time out if the specified time has elapsed since the arrival of the last section being successfully filtered. Setting a timeout of 0 ms indicates that no time-out will be used.

SectionFilterResult

```
enum SectionFilterResult {
      SF_REVOKED,
      SF_STOPPED,
      SF_TIME_OUT
      SF_TS_CHANGED,
      SF_SECTION_FILTERED,
      SF_BUFFER_OVERFLOW,
      SF_TABLE_COMPLETE,
      SF_TABLE_VERSION_CHANGED
};
```

Description

This structure describes the possible notification that can be sent back to the application. `SF_REVOKED` implies that, for some reason, the section filter resource has been revoked. One of the reasons that a filter is revoked can be that the secondary resource owner looses the filter because a primary resource owner reserved a section filter and no free section filters are available. `SF_STOPPED` implies that the section filter has been stopped, due to the calling of the `DvbSfExt::StopFiltering` method. `SF_TIME_OUT` indicates that a time out has occurred in the section filter (see also `SectionFilterParameters`). `SF_TS_CHANGED` indicates that the transport stream has changed (and consequently, the section filter has been stopped). `SF_SECTION_FILTERED` implies that a section has been filtered. For a SimpleSectionFilter, this also implies that the section filter has stopped section filtering. `SF_BUFFER_OVERFLOW` indicates that a buffer overflow has occurred. `SF_TABLE_COMPLETE` indicates that the table section filter has completed the filtering of a complete table. `SF_TABLE_VERSION_CHANGED` indicates that, during the filtering of a table by a table section filter, the version of the table has changed. After such a version change has been detected, the TableSectionFilter continues filtering on the old version number.

FilteringResult

```
struct FilteringResult {
      SectionFilterResult result;
      sequence<Section> sections;
};
```

Description

This structure describes the complete result that a section filter can send back to the application. The result of the section filter is indicated by the result field. The section(s) is/are present in the sections field.

A.4.4 API definition

DvbSfExt::GetBaseFcmSeid

Prototype

```
Status DvbSfExt::GetBaseFcmSeid(
      out SEID baseFcmSeid)
```

Parameters

- `baseFcmSeid` – the SEID of the FCM this is an extension on.

Description

This method returns the software element identifier of the HAVi Tuner FCM extended by the services of clause A.4.2.

DvbSfExt::ReserveSf

Prototype

```
Status DvbSfExt::ReserveSf(
      in    OperationCode   OpCode,
      out   uint            SfId
      )
```

Parameters

- `OpCode` – operation code of the client call back function.
- `SfId` – identifier of the section filter.

Description

This method reserves a section filter from the available section filters. Before a client can use a section filter, first a section filter has to be reserved.

Error codes

- `DvbSfExt::EUNAVAILABLE` – no section filter was available.

DvbSfExt::ReleaseSf

Prototype

```
Status DvbSfExt::ReleaseSf(
      in uint  SfId )
```

Parameters

- `SfId` – identifier of the section filter.

Description

This method releases the section filter, indicated by `SfId`. Releasing implies that the client is no longer able to use this section filter and that the tuner can allocate this section filter to another client at the moment that such a client tries to reserve a section filter.

Filter will be automatically released if their clients cannot be reached anymore (detectable via watch-on).

Error codes

- `DvbSfExt::ENO_NOT` – This SfId was not allocated to the client application, or was unknown to the tuner.

DvbSfExt::StartFiltering

Prototype

```
Status DvbSfExt::StartFiltering(
      in    uint                      SfId
      in    SectionFilterParameters   SfParams
      out      uint                   FilterOperationId
   )
```

Parameters

- `SfId` – identifier of the section filter.
- `SfParams` – section filter parameters.
- filterType –type of filtering action applied.
- `FilterOperationId` – an identifier to this specific filtering operation.

Description

This method starts the section filter, indicated by `SfId.` After calling this method, section filtering starts, given the specified parameters (`SfParams`) and according to the required `filterType`. The method returns an identifier (`FilterOperationId`) that identifies the filter action started with this method call.

Error codes

- `EINVALID_PARAMETER` - Error in SF parameters
- `DvbSfExt::ENO_NOT` – This SfId was not allocated to the client application, or was unknown to the tuner.
- `DvbSfExt::EMEM_NOT_AVAILABLE` - the required amount of memory for filtering is not available.
- `DvbSfExt::ELOCATOR` – if the Tuner FCM cannot resolve the locator, i.e. the tuner was not tuned to the requested transport stream.
- `DvbSfExt::EALREADY_STARTED – The section filter has already been started.`

DvbSfExt::StopFiltering

Prototype

```
Status DvbSfExt::Stopfiltering(
      in uint           SfId
)
```

Parameters

- `SfId` – identifier of the section filter.

Description

This method stops the section filter, indicated by `sfid.` After calling this method, the tuner will sent back a notification to the calling client at the moment the actual section filtering has been stopped.

Error codes

- `DvbSfExt::ENO_NOT` – this SfId was not allocated to the client application, or was unknown to the tuner.
- `DvbSfExt::EINVALID_OPERATION_ID – the FilteringOperationId is unknown to the tuner.`
- `DvbSfExt::EINVALID_ID` – the SfId is invalid.

<Client>::SfNotification

Prototype

```
Status <Client>::SfNotification(
      in uint               SfId,
      in uint               FilteringOperationId
      in FilteringResult    SfResult)
```

Parameters

- `SfId` – identifier of the section filter.
- `SfResult` – the result of the section filter. The safe parameter size limit regarding the amount of data carried in the sections part of SfResult is equal to the specified buffersize in the corresponding SectionFilterParameters. The actual size of `SfResult` is determined by the Tuner FCM taken into account its capacity.
- `FilteringOperationId` – an identifier to this filter operation (if applicable).

Description

During the subscription, a "message back" is send to the subscriber, each time a `SfResult` is available for the specified section filter using the operation code the subscriber has specified via `DvbSfExt::StartFiltering`.

A.4.5 Events

6.7.5.1.1 SfAvailable

Prototype

```
void SfAvailable()
```

Parameters

Description

This event notifies its subscribers that a section filter has become available for HLN devices/applications.

A.5 HAVi constants for DVB extensions

This clause defines the different constants that are required by the previous interface specification.

A.5.1 Software element types

This is extension on section 11.3 of the HAVi specification [16].

In combination with the DVB VendorId, the following software element types are additionally defined:

DVB Software Element Type	ATT_SE_TYPE Value	Trusted	System Element
DVB_TUNER_EXT	0x8000 0000	yes	no
DVB_SI_EXT	0x8000 0001	yes	no
SECTION_FILTER_EXT	0x8000 0002	yes	no

A.5.2 API codes

This is extension on section 11.5 of the HAVi specification [16].

In combination with the DVB VendorId, the API codes are additionally defined:

DVB API Name	API Code
DvbTunerExtension	0x8000
DvbSiExtension	0x8001
DvbSfExtension	0x8002

A.5.3 Operation codes

This is extension on section 11.6 of the HAVi specification [16].

In combination with the DVB VendorId, the following Operation codes are defined:

DVB Message	API Code	Operation ID
DvbTunerExt::GetBaseFcmSeid	0x8000	0x00
DvbTunerExt::GetSfSeid	0x8000	0x01
DvbTunerExt::GetDvbSiSeid	0x8000	0x02
DvbTunerExt::SelectSourceTs	0x8000	0x03
DvbTunerExt::GetBouquetServiceLists	0x8000	0x04
DvbTunerExt::GetNetworkServiceLists	0x8000	0x05
DvbSiExt::GetBaseFcmSeid	0x8001	0x00
DvbSiExt::RetrieveNetworks	0x8001	0x01
DvbSiExt::RetrieveActualNetwork	0x8001	0x02
DvbSiExt::RetrieveBouquets	0x8001	0x03
DvbSiExt::RetrieveServices	0x8001	0x04
DvbSiExt::RetrieveActualServices	0x8001	0x05
DvbSiExt::RetrievePresentEvent	0x8001	0x06
DvbSiExt::RetrieveFollowing	0x8001	0x07
DvbSiExt::RetrieveScheduledEvents	0x8001	0x08
DvbSiExt::RetrieveDvbNetworkInfo	0x8001	0x09
DvbSiExt::RetrieveDvbBouquetInfo	0x8001	0x0A
DvbSiExt::RetrieveDvbServiceInfo	0x8001	0x0B
DvbSiExt::RetrieveDvbProgramInfo	0x8001	0x0C
DvbSiExt::RetrieveDvbEventInfo	0x8001	0x0D
DvbSiExt::MonitorStart	0x8001	0x0E
DvbSiExt::MonitorStop	0x8001	0x0F
DvbSfExt::GetBaseFcmSeid	0x8002	0x00
DvbSfExt::ReserveSf	0x8002	0x01
DvbSfExt::ReleaseSf	0x8002	0x02
DvbSfExt::StartFiltering	0x8002	0x03
DvbSfExt::StopFiltering	0x8002	0x04

A.5.4 Error codes

This is extension on section 11.7 of the HAVi specification [16].

In combination with the DVB VendorId, the following error codes are defined:

DVB Error Name	API Code	Error Code
DvbTunerExt::ELOCATOR	0x8000	0x0000
DvbTunerExt::ELIST	0x8000	0x0001
DvbSiExt::ELOCATOR	0x8001	0x0000
DvbSiExt::ENOT_IN_CACHE	0x8001	0x0001
DvbSiExt::ETABLE_NOT_FOUND	0x8001	0x0002
DvbSiExt::ENOT_IN_TABLE	0x8001	0x0003
DvbSiExt::EUNAVAILABLE	0x8001	0x0004
DvbSiExt::ENO_NOT	0x8001	0x0005
DvbSfExt::EUNAVAILABLE	0x8002	0x0000
DvbSfExt::ENO_NOT	0x8002	0x0001
DvbSfExt::ELOCATOR	0x8002	0x0002
DvbSfExt::EMEM_NOT_AVAILABLE	0x8002	0x0003
DvbSfExt::EALREADY_STARTED	0x8002	0x0004
DvbSfExt::EINVALID_OPERATION_ID	0x8002	0x0005
DvbSfExt::EINVALID_ID	0x8002	0x0006

A.5.5 DVB event types

This is an extension on section 11.9 of the HAVi specification [16]. The following table lists the base event number for all event types used in combination with the DVB vendor id.

DVB Event Type	Posted By	Distribution	Base Event Number
SfAvailable	DvbSfExt	global	0x0001

Annex B (informative): Bibliography

TM1690 (rev 2): "User and Market Requirements for In-Home Digital Networks".

DVB-IHDN 012 (rev 6): "In-Home Digital Network".

History

Document history		
V1.1.1	May 2001	Publication