INTERNATIONAL STANDARD

ISO/IEC 16500-5

First edition
1999-12-15

Information technology — Generic digital audio-visual systems —

Part 5: High and mid-layer protocols

Technologies de l'information — Systèmes audiovisuels numériques génériques —

Partie 5: Protocoles de la couche haute et moyenne

Reference number
ISO/IEC 16500-5:1999(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 16500 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 16500-5 was prepared by DAVIC (Digital Audio-Visual Council) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

ISO/IEC 16500 consists of the following parts, under the general title *Information technology — Generic digital audio-visual systems*:

— *Part 1: System reference models and scenarios*

— *Part 2: System dynamics, scenarios and protocol requirements*

— *Part 3: Contours: Technology domain*

— *Part 4: Lower-layer protocols and physical interfaces*

— *Part 5: High and mid-layer protocols*

— *Part 6: Information representation*

— *Part 7: Basic security tools*

— *Part 8: Management architecture and protocols*

— *Part 9: Usage information protocols*

Annexes A and B form a normative part of this part ISO/IEC 16500. Annex C is for information only.

Introduction

ISO/IEC 16500 defines the minimum tools and dynamic behavior required by digital audio-visual systems for end-to-end interoperability across countries, applications and services. To achieve this interoperability, it defines the technologies and information flows to be used within and between the major components of generic digital audio-visual systems. Interoperability between these components and between individual sub-systems is assured through specification of tools and specification of dynamic systems behavior at defined reference points. A reference point can comprise one or more logical (non-physical) information-transfer interfaces, and one or more physical signal-transfer interfaces. A logical interface is defined by a set of information flows and associated protocol stacks. A physical interface is an external interface and is fully defined by its physical and electrical characteristics. Accessible reference points are used to determine and demonstrate compliance of a digital audio-visual subsystem with this international standard.

A summary of each part follows.

ISO/IEC 16500-1 (DAVIC 1.3.1a Part 2) defines the normative digital audio-visual systems technical framework. It provides a vocabulary and a Systems Reference Model, which identifies specific functional blocks and information flows, interfaces and reference points.

ISO/IEC 16500-2 (DAVIC 1.3.1a Part 12) defines system dynamic behavior and physical scenarios. It details the locations of the control functional entities along with the normative protocols needed to support the systems behavior. It is structured as a set of protocol walk-throughs, or "*Application Notes*", that rehearse both the steady state and dynamic operation of the system at relevant reference points using specified protocols. Detailed dynamics are given for the following scenarios: video on demand, switched video broadcast, interactive broadcast, and internet access.

ISO/IEC 16500-3 (DAVIC 1.3.1a Part 14) provides the normative definition of DAVIC Technology Contours. These are strict sets of Applications, Functionalities and Technologies which allow compliance and conformance criteria to be easily specified and assessed. This part of ISO/IEC 16500 contains the full details of two contours. These are the Enhanced Digital Broadcast (EDB) and Interactive Digital Broadcast (IDB). ISO/IEC 16500-3 specifies required technologies and is a mandatory compliance document for contour implementations.

ISO/IEC 16500-4 (DAVIC 1.3.1a Part 8) defines the toolbox of technologies used for lower layer protocols and physical interfaces. The tools specified are those required to digitize signals and information in the Core Network and in the Access Network. Each tool is applicable at one or more of the reference points specified within the Delivery System. In addition a detailed specification is provided of the physical interfaces between the Network Interface Unit and the Set Top Unit and of the physical interfaces used to connect Set Top Boxes to various peripheral devices (digital video recorder, PC, printer). The physical Delivery System mechanisms included are copper pairs, coaxial cable, fiber, HFC, MMDS, LMDS, satellite and terrestrial broadcasting.

ISO/IEC 16500-5 (DAVIC 1.3.1a Part 7) defines the technologies used for high and mid-layer protocols for ISO/IEC 16500 digital audio-visual systems. In particular, this part defines the specific protocol stacks and requirements on protocols at specific interfaces for the content, control and management information flows.

ISO/IEC 16500-6 (DAVIC 1.3.1a Part 9) defines what the user will eventually see and hear and with what quality. It specifies the way in which monomedia and multimedia information types are coded and exchanged. This includes the definition of a virtual machine and a set of APIs to support interoperable exchange of program code. Interoperability of applications is achieved, without specifying the internal design of a set top unit, by a normative Reference Decoder Model which defines specific memory and behavior constraints for content decoding. Separate profiles are defined for different sets of multimedia components.

ISO/IEC 16500-7 (DAVIC 1.3.1a Part 10) defines the interfaces and the security tools required for an ISO/IEC 16500 system implementing security profiles. These tools include security protocols which operate across one or both of the defined conditional access interfaces CA0 and CA1. The interface CA0 is to all security and conditional access functions, including the high speed descrambling functions. The interface CA1 is to a tamper resistant device used for low speed cryptographic processing. This cryptographic processing function is implemented in a smart card.

ISO/IEC 16500-8 (DAVIC 1.3.1a Part 6) specifies the information model used for managing ISO/IEC 16500 systems. In particular, this part defines the managed object classes and their associated characteristics for managing the access network and service-related data in the Delivery System. Where these definitions are taken from existing standards, full reference to the required standards is provided. Otherwise a full description is integrated in the text of this part. Usage-related information model is defined in ISO/IEC 16500-9.

ISO/IEC 16500-9 (DAVIC 1.3.1a Part 11) specifies the interface requirements and defines the formats for the collection of usage data used for billing, and other business-related operations such as customer profile maintenance. It also specifies the protocols for the transfer of Usage Information into and out of the ISO/IEC 16500 digital audio-visual system. In summary, flows of audio, video and audio-visual works are monitored at defined usage data collection elements (e.g., servers, elements of the Delivery System, set-top boxes). Information concerning these flows is then collected, processed and passed to external systems such as billing or a rights administration society via a standardised usage data transfer interface.

Additional Information

ISO/IEC TR 16501 is an accompanying Technical Report. Further architectural and conformance information is provided in other non-normative parts of DAVIC 1.3.1a (1999). A summary of these documents is included here for information.

ISO/IEC TR 16501 (DAVIC 1.3.1a Part 1) provides a detailed listing of the functionalities required by users and providers of digital audio-visual applications and systems. It introduces the concept of a contour and defines the IDB (Interactive Digital Broadcast) and EDB (Enhanced Digital Broadcast) functionality requirements which are used to define the normative contour technology toolsets provided in ISO/IEC 16500-3.

DAVIC 1.3.1a Parts 3, 4 and 5 are DAVIC technical reports. They provide additional architectural and other information for the server, the delivery-system, and the Service Consumer systems respectively. Part 3 defines how to load an application, once created, onto a server and gives information and guidance on the protocols transmitted from the set-top user to the server, and those used to control the set-up and execution of a selected application. Part 4 provides an overview of Delivery Systems and describes instances of specific DAVIC networked service architectures. These include physical and wireless networks. Non-networked delivery (e.g., local storage physical media like discs, tapes and CD-ROMs) are not specified. Part 5 provides a Service Consumer systems architecture and a description of the DAVIC Set Top reference points defined elsewhere in the normative parts of the specification.

DAVIC 1.3.1a Part 13 is a DAVIC technical report, which provides guidelines on how to validate the systems, technology tools and protocols through conformance and/or interoperability testing.

Information technology — Generic digital audio-visual systems — Part 5: High and mid-layer protocols

1. Scope

This part of ISO/IEC 16500 covers the high and mid layer protocols for DAVIC systems. It defines a set of protocol components ("tools") which are referenced by ISO/IEC 16500-2 ("System Dynamics, Scenarios, and Protocol Requirements").

In particular this part of ISO/IEC 16500 describes the protocol stacks required for the support of the DAVIC flows, i.e., S1 through S5. The specific DAVIC options for each of the protocols are specified as well as the different optional protocols stacks applicable. Requirements on protocols at specific interfaces are also considered and included in the specification.

2. Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 16500. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 16500 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau (TSB) maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation H.220.0 (1996) | ISO/IEC 13818-1:1996, *Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems*
  (Note: known as MPEG-2).

  ISO/IEC 13818-1/Amendment 1:1997, *Registration procedure for "copyright identifier".*

  ISO/IEC 13818-1/Amendment 2:1997, *Registration procedure for "format identifier".*

  ISO/IEC 13818-1/Amendment 3:1998, *Private data identifier.*
- ITU-T Recommendation H.262 (1996) | ISO/IEC 13818-2:1996, *Information technology – Generic coding of moving pictures and associated audio information: Video* (Note: known as MPEG-2).
- ITU-T Recommendation X.215 (1995) | ISO/IEC 8326:1996, *Information technology — Open Systems Interconnection — Session service definition.*
- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition.*
- ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element.*
- ITU-T Recommendation X.225 (1995) | ISO/IEC 8327-1:1996, *Information Technology – Open Systems Interconnection – Connection-oriented session protocol: Protocol specification.*
- ITU-T Recommendation X.226 (1994) | ISO/IEC 8823-1:1994, *Information Technology – Open Systems Interconnection – Connection-oriented protocol: Protocol specification.*
- ITU-T Recommendation X.233 (1993) | ISO/IEC 8473-1:1994, *Information technology – Protocol for providing the connectionless-mode network service: Protocol specification.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- ITU-T (CCITT) Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1).*

  ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1).*

- ITU-T (CCITT) Recommendation X.209 (1988), *Specification of Basic Encoding rules for abstract syntax notation one (ASN.1).*

  ISO/IEC 8825:1990, *Information Technology – Open Systems Interconnection – ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) for Abstract Syntax Notation One (ASN.1).*
- ITU-T Recommendation X.218 (1995), *Reliable transfer: model and service definition.*

  ISO/IEC 9066-1:1989, *Information processing systems – Text communication – Reliable transfer: Model and service definition.*
- ITU-T (CCITT) Recommendation X.219 (1988), *Remote operations: Model, notation and service definition.*

  ISO/IEC 9072-1:1989, *Information processing systems – Text communication – Remote Operations: Model, notation and service definition.*
- ITU-T Recommendation X.224 (1993), *Protocol for providing the OSI connection-mode transport service*.

  ISO/IEC 8073:1992, *Information technology – Telecommunications and information exchange between systems – Open Systems Interconnection – Protocol for providing the connection-mode transport service.*
- ITU-T (CCITT) Recommendation X.229 (1988), *Remote operations: Protocol specification.*

  ISO/IEC 9072-2:1989, *Information processing systems – Text communication – Remote Operations: Protocol specification.*

2.3 IEC, ISO, and ISO/IEC Standards

- IEC 61883-1:1997, *Consumer audio/video equipment – Digital Interface. – Part 1: General.*
- IEC 61883-4:1997, *Consumer audio/video equipment – Digital Interface – Part 4: MPEG-2 TS data transmission.*
- ISO/IEC 11172-2:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbits/s – Part 2: Video.* (Note: also known as MPEG-1).
- ISO/IEC 11172-3:1993, *Information technology Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbits/s – Part 3: Audio.* (Note: also known as MPEG-1).
- ISO/IEC 13818-3:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio*. (MPEG-2).
- ISO/IEC 13818-6:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSM-CC.*
- ISO/IEC 14750, Information technology – Open distributed processing - *Interface Definition Language.*
- ISO/IEC 7776:1995, *Information technology – Telecommunications and information exchange between systems – High-level data link control procedures – Description of the X.25 LAPB-compatible DTE data link procedures.*
- ISO/IEC 8802-2:1994, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control.*
- ISO/IEC 8802-3:1993, *Information technology – Local and metropolitan area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.*

2.4 ITU-T Recommendations

- ITU-T Recommendation E.164 (R/1996) *Numbering plan for the ISDN era*
- ITU-T Recommendation G.774 (1992), *Synchronous digital hierarchy management information model*
- ITU-T I.150 (R/1995), *B-ISDN asynchronous transfer mode functional characteristics*
- ITU-T Recommendation I.361 (R/1996), *B-ISDN ATM layer specification*
- ITU-T Recommendation I.363 (R/1994), *B-ISDN ATM adaptation layer (AAL) specification*

- ITU-T Recommendation I.363.5 (1996), *B-ISDN ATM Adaptation Layer Type 5 (AAL5) specification*
- ITU-T Recommendation I.610 (R/1995), *B-ISDN operation and maintenance principles and functions*
- ITU-T Recommendation M.3100 (1995), *Generic network information model*
- ITU-T Recommendation Q.822 (1994), *Stage 1, Stage 2, Stage 3 description for the Q3 interface - Performance management*
- ITU-T Recommendation Q.2110 (1994), *B-ISDN ATM Adaptation Layer – Service Specific Connection Oriented Protocol (SSCOP)*
- ITU-T Recommendation Q.2120 (1995), *B-ISDN meta-signaling protocol (SAAL)*
- ITU-T Recommendation Q.2130 (1994), *B-ISDN ATM Adaptation Layer- Service Specific Coordination Function for Support of Signaling at the User Network Interface (SSCF at UNI)*
- ITU-T Recommendation Q.2931 (1995), *Broadband integrated services digital network (B-ISDN). Digital Subscriber Signaling System No.2 (DSS 2): User-network interface (UNI) layer 3 specification for basic call/connection control*
- CCITT Recommendation V.35, *Data transmission at 48 Kilobits per second using 60-108 kHz Group Band Circuits*
- ITU-T Recommendation X.25 (1993), *Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit*
- ITU-T Recommendation X.710 (1991), *Common management information service definition for CCITT applications*
- ITU-T Recommendation X.711 (1991), *Common Management Information Protocol, Specification for CCITT applications*
- ITU-T Recommendation X.721 (R/1994), *Information Technology – Open Systems Interconnection – Structure of Management Information: Definition of Management Information*

2.5 Other normative references

2.5.1 ANSI

- ANSI EIA/TIA-232-E-91, *Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, July 1991

2.5.2 The ATM Forum

- AF-SAA-0049.001, The ATM Forum, *Audiovisual Multimedia Services: Video on Demand Specification 1.1*, 1997

2.5.3 Electronic Industries Association (EIA) and Telecommunications Industry Association (TIA)

- RS-232-C, *see* ANSI/EIA/TIA-232-E-91

2.5.4 European Telecommunications Standards Institute (ETSI)

- ETSI 300 468, *Digital Video Broadcasting, Specification for Service Information in DVB Systems*, January 1997
  (Note: includes normative reference to the *Informative Annex C: Conversion Between Time and Date Conventions.*)

2.5.5 Institute for Electrical and Electronic Engineers (IEEE)

- IEEE 802.2, *see* ISO/IEC 8802-2
- IEEE 802.3, *see* ISO/IEC 8802-3

- IEEE 1394-1995, *Standard for a High Performance Serial Bus*, August 1996

2.5.6 Internet Society

- RFC 768, J. Postel, *User Datagram Protocol (UDP)*, 08/28/1980
- RFC 791, J. Postel, *Internet Protocol (IP Addressing)*, 09/01/1981
- RFC 793, J. Postel, *Transmission Control Protocol (TCP)*, 09/01/1981
- RFC 826, D. Plummer, *Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware*, 11/01/1982
- RFC 1006, D. Cass, M. Rose, *ISO transport services on top of the TCP: Version 3*, 05/01/1987
- RFC 1155, K. McCloghrie, M. Rose, *Structure and Identification of Management Information for TCP/IP-based Internets*, 05/10/1990
- RFC 1157, M. Schoffstall, M. Fedor, J. Davin, J. Case, *A Simple Network Management Protocol (SNMP)*, 05/10/1990
- RFC 1212, K. McCloghrie, M. Rose, *Concise MIB Definitions*, 03/26/1991
- RFC 1213, K. McCloghrie, M. Rose, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, 03/26/1991
- RFC 1662, W. Simpson, *PPP in HDLC-like Framing*, 07/21/1994
- RFC 1700, J. Reynolds, J. Postel, *Assigned Numbers*, 10/20/1994
  (Note: this supercedes RFC 1060 and 1340. The IETF regularly updates "Assigned Numbers" so this reference may be obsolete in the future)

2.5.7 Object Management Group (OMG)

- *Common Object Request Broker: Architecture and Specification*, Version 2.1, August 1997
  (Note: known as OMG CORBA 2.1. Includes definition of the Common Data Representation and the Remote Procedure Call mechanism and encoding rules for General Inter-ORB Protocol, and Internet Inter-ORB Protocol).

3. Definitions

This clause defines new terms, and the intended meaning of certain common terms, used in this part of ISO/IEC 16500. Annex A of ISO/IEC 16500-1 defines additional terms and, in some cases, alternative interpretations that are appropriate in other contexts. For convenience, the normative definitions below are included in the annex.

3.1. Access Network (AN): a part of the Delivery System consisting of a collection of equipment and infrastructures, that link a number of Service Consumer Systems to the rest of the Delivery System through a single (or a limited number of) common port(s).

3.2. Access Node: the element of the Access Network containing centralized functions responsible for processing information flows in preparation for transport through the selected distribution network.

3.3. Application Programming Interface (API): set of inter-layer service request and service response messages, message formats, and the rules for message exchange between hierarchical clients and servers. API messages may be executed locally by the server, or the server may rely on remote resources to provide a response to the client.

3.4. Control Plane: a classification for objects that interact to establish, maintain, and release resources and provide session, transport, and connection control functions that facilitate transparent information transfers between ISP clients.

3.5. call: (in signaling) An association between two or more users, or between a user and a network entity, that is established by use of network capabilities. This association may have zero or multiple information exchange mechanisms established within this call, for example in connection-oriented or in connectionless modes. (ITU-T Rec. Q.9)

3.6. channel: a connection conveying signals between two blocks (the conveyed signals represent information). Channels also convey signals between a block and the environment. Channels may be unidirectional or bi-directional.

3.7. client: a service consuming object or system (block); (a synonym for user).

3.8. conditional access: a means of allowing system users to access only those services that are authorized to them.

3.9. connection: an association of transmission channels or circuits, switching and other functional units set up to provide a means for a transfer of user, control and management information between two or more end points (blocks) in a telecommunications network. (ITU-P.10).

3.10. connection service: provides basic functions to create, maintain, and tear down connections.

3.11. content-information: information that does not alter the state of the object intercepting the information flow, e.g., audio, video, or data in a television program that is processed transparently by a television receiver (the control state of the receiver will not change as a result of such information).

3.12. control-information: information that may change the state of the object intercepting the information flow, e.g., a remote control channel up command input. (In some cases an object may interpret a message but reject a request and remain in its current state.)

3.13. Core Network: a portion of the Delivery System composed of networks, systems, equipment and infrastructures, connecting the Service Providers to the Access Networks.
NOTE: The term Core Network, in the DAVIC use, is wide sense as it includes the notion of the access networks that are needed to link the Service Providers Systems to the core network in strict sense (i.e., exclusive of any access network). This kind of access networks are not under consideration within DAVIC.

3.14. Delivery System (DS): The portion of the DAVIC System that enables the transfer of information between DS-users.

3.15. domain: a scope that delimits (makes clear what is included and what is not) the extent of influence of one object on another. Domain boundaries may represent regulatory, ownership, span-of-control and other influence factors.

3.16. downstream: information flow direction is from an End Service Provider System to an End Service Consumer System.

3.17. End-Service Consumer (ESC): a user, either human or machine, whose primary interaction with the system is through the STU.

3.18. End-Service Consumer System (ESCS): a system that (predominantly) consumes information. ESCSs are ISPS and ESPS clients. The ESCS includes the STU and the ESC.

3.19. End-Service Provider (ESP): an entity with jurisdiction over a domain that contains a system that (predominantly) provides information to clients.

3.20. End-Service Provider System (ESPS): a system that (predominantly) provides information to clients. ESPSs are ISPS clients and may also be clients of other ESPSs. ESPSs consist of hardware and software sub-systems that use ISP services to provide video and multimedia services to ESCSs.

3.21. layer: a collection of objects of the same hierarchical rank.

3.22. layer service: functions provided by a layer to a layer above through a Service Access Point.

3.23. Intermediate-Service Provider (ISP): ISPs provide adjunct services and convey information among ESPs and ESCs.

3.24. Management Entity: The Management Entity is responsible for the operation and maintenance functions of a network. Several instances of information flow S5 exist for the Management entity.

3.25. Management Plane (MP): a plane that contains those interfaces and functions which support interactions which may be typified as being temporally disjoint from an off-hook interaction. Interactions among Management Plane objects may also occur concurrently with an off-hook interaction.

3.26. management-information: information exchanged by Management Plane objects; may be content-information or control-information.

3.27. navigation: the process of reaching a service objective by means of making successive choices; the term may be applied to the selection of a service category, a service provider or an offer within a particular service.

3.28. network: a collection of interconnected elements that provides connection services to users.

3.29. Network Interface Unit (NIU): the NIU accepts network specific content-information flows from the Delivery System and provides a non-network specific interface to the Connectivity Entity in the STU. (Additional definitions of the NIU may exist).

3.30. Network Termination (NT): the element of the Access Network performing the connection between the infrastructure owned by the Access Network operator and the Service Consumer System (ownership de-coupling). The NT may be passive or active, transparent or not.

3.31. physical interface: an interface where the physical characteristics of signals used to represent information and the physical characteristics of channels used to carry the signals are defined. A physical interface is an external interface. It is fully defined by its physical and electrical characteristics. Logical information flows map to signal flows that pass through physical interfaces.

3.32. plane: a category that identifies a collection of related objects, e.g., objects that execute similar or complementary functions; or peer objects that interact to use or to provide services in a class that reflects authority, capability, or time period. Management-plane service objects, for example, may authorize ISP-clients' access to certain control-plane service objects that in turn may allow the clients to use services provided by certain user-plane objects.

3.33. port: an abstraction used by transport protocols to distinguish among multiple destinations associated with particular applications running on a host computer: and application may specify the ports it wants to use; some ports are reserved for standard applications/ services such as e-mail (also known as well-known ports).

3.34. procedure: an encapsulation of the behavior of part of a process. A procedure is defined in one place but may be referred to several times within a process.

3.35. process: a communicating extended finite state machine. Communication may take place via signals or shared variables. The behavior of a process depends on the order of arrival of signals in its input port.

3.36. protocol: set of message formats (semantic, syntactic, and symbolic rules) and the rules for message exchange between peer layer entities (which messages are valid when)

3.37. reference point: a set of interfaces between any two related blocks through which information flows from one block to the other. A reference point comprises one or more logical (non-physical) information-transfer interfaces, and one or more physical signal-transfer interfaces.

3.38. S1: content-information flow, from a source to a destination object on the User Plane of any service layer.

3.39. S2: control-information flow from a source to a destination object on the Control Plane of the Application Service Layer (SL1).

3.40. S3: control-information flow from a source to a destination object on the Control Plane of the Session and Transport Service Layer (SL2).

3.41. S4: control-information flow from a source to a destination object on the Control Plane of the Network Service Layer (SL3).

3.42. S5: management-information flow from a source to a destination object on the Management Plane of the container object: the objects may be peers (service layer is known), or the service layer may be unspecified.

3.43. server: any service providing system.

3.44. Service Gateway (SGW): an element of the service domain through which a client may browse available services; also a mechanism for a client to obtain an object reference (message connection) to an instance of a service.

3.45. Service Provider System (SPS): a general reference to ESP or ISP systems.

3.46. session services: provide basic functions to create, modify, maintain, and tear down sessions (negotiate and allocate network resources).

3.47. session: an interval during which a logical, mutually agreed correspondence between two objects exists for the transfer of related information. A session defines a relationship between the participating users in a service instance.

3.48. Stream Service Element: this Service Provider System entity allows for the processing of the content-information flows at the stream level. The content stream is a sub-set of the actual content-information flow and processing at the stream level allow the ability to uniquely align a service offering.

3.49. User Plane (UP): a classification for objects whose principal function is to provide transfer of (end) user information: user information may be user-to-user content (e.g., a movie), or private user-to-user data.

3.50. Value-Added Service Provider (VASP): This provider offers, for example, a Video-on-Demand Service to the end-user. Within the Systems Reference Model this is the ESP.

4. Acronyms and abbreviations

This clause defines the acronyms and abbreviations used in this part of ISO/IEC 16500. Annex B of ISO/IEC 16500-1 defines acronyms and abbreviations used within ISO/IEC 16500.

AAL	ATM Adaptation Layer
ACSE	Application Control Service Element
AFI	Authority and Format Identifier
ANSI	American National Standards Institute
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CBD	Connection Block Descriptor
CLNP	ConnectionLess Network Protocol
CLP	Cell Loss Priority
CMIP	Common Management Information Protocol
CMISE	Common Management Information Service Element
CORBA	Common Object Request Broker Architecture
CPCS	Common Part Convergence Sub-layer
DLL	Data Link Layer
DSM-CC	Digital Storage Media - Command and Control
DSM-CC U-N	Digital Storage Media - Command and Control User-to-Network
DSM-CC U-U	Digital Storage Media - Command and Control User-to-User
DVB	Digital Video Broadcasting
GIOP	Generic Inter-ORB Protocol
HFC	Hybrid Fiber Coax
HO-DST	High Order - Domain Specific Part
IBC	Inter-Integrated Circuit
IDLE	Interface Definition Language
IEC	International Electrotechnical Commission
IIOP	Internet Inter-ORB Protocol
ILMI	Integrated Local Management Interface
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union - Telecommunications sector
L1GW	Level 1 Gateway
LAPB	Link Access Procedure Balanced
LAPD	Link Access Procedure D-channel
MAC	Medium Access control
MIB	Management Information Base
MPEG	Moving Pictures Experts Group
MPEG-TS	MPEG-2 Transport Stream
N-ISDN	Narrowband Integrated Services Digital Network
NIU	Network Interface Unit

NMS	Network Management System
NSAP	Network Service Access Point
OAM	Operation and Maintenance
OMG-CDR	Object Management Group - Common Data Representation
OMG-GIOP	Object Management Group - Generic Inter-ORB Protocol
PDU	Protocol Data Unit
PES	Packetized Elementary Stream
PID	Packet Identifier
PSI	Program Specific Information
PSTN	Public Switched Telephone Network
PVC	Permanent Virtual Connection
RF	Radio Frequency
ROSE	Remote Operation Service Element
RPC	Remote Procedure Call
SAAL	Signaling ATM Adaptation Layer
SAR	Segmentation and Re-assembly
SDU	Service Data Unit
SNMP	Simple Network Management Protocol
SSCOP	Service Specific Connection Oriented Protocol
STU	Set Top Unit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNI	User Network Interface
VASP	Value Added Service Provider
VCI	Virtual Channel Identifier
VP	Virtual Path
VPCI	Virtual Path Connection Identifier
VPI	Virtual Path Identifier

5. Conventions

The style of this Specification follows the *Guide for ITU-T and ISO/IEC JTC1 co-operation, Appendix H: Rules for presentation of ITU-T | ISO/IEC Common text* (March, 1993).

6. S1 flow: high and mid layer protocols

6.1 S1 flow description

As defined in ISO/IEC 16500-1 clause 7, the S1 information flow is for content-information flow from a source object to a destination object. For ISO/IEC 16500, S1 is defined to be a uni-directional flow from the server to the STU carrying encoded video/audio content and associated data, binary objects to be used by the STU, and other content-information types. ISO/IEC 16500 has selected ISO/IEC 11172-2,-3 (MPEG-1 video and audio) and ISO/IEC 13818-1,-2,-3 (MPEG-2 systems, video, and audio) for the coding of the video/audio content information and the Transport Stream systems layer for the multiplexing of video, audio and other data types; and ISO/IEC 13818-6 (MPEG-2 digital storage media command and control) for the binary object (and other data types) download protocol. Refer to ISO/IEC 16500-6 for information on other content-information types.

6.2 Overview of protocol stacks

The allowable protocol stacks for carrying S1 information flows are defined in the following figures. When S1 consists of an MPEG program (a single video stream, associated audio streams and data), the MPEG-2 Transport Stream format shall be used as indicated in the diagrams below. In all cases where ATM is used in the access network (in particular, for ATM across the A1 interface), the Transport Stream shall contain one and only program (referred to as a single-program Transport Stream or SPTS). See Figure 6.1. In all other cases – e.g., ATM layer which does not cross A1 (also Figure 6.1) or for a non-ATM HFC access network (Figure 6.2) – the Transport Stream may contain more than one program (referred to as a multi-program Transport Stream or MPTS). See ISO/IEC 16500-4 for information on the lower layer protocols.

When S1 consists of “download” data or other content information, the information shall be carried within a “high bandwidth” Transport Stream (per either Figure 6.1 or Figure 6.2). In addition, S1 download data may be carried in a separate ATM Virtual Channel as shown in Figure 6.3. The selection of the option shall be done during service activation time using DSM-CC resource descriptors and binding lists (see ISO/IEC 16500-2).

S1 information shall be carried transparently through the Delivery System, except in the case where MPEG Transport Stream re-multiplexing is performed in cascaded SPS networks (see DAVIC 1.3.1a Part 3). This exception occurs because the originating SPS may deliver MPEG programs in SPTS format which are subsequently multiplexed into MPTS prior to delivery to the STU. In all cases, the data/elementary stream information and PES/private_section information shall be passed transparently through the Delivery System. For all MPEG-2 Transport Streams, the PSI tables shall be included. Other data – e.g., DVB Service Information, ATSC Program and System Information, or conditional access information – is optional.

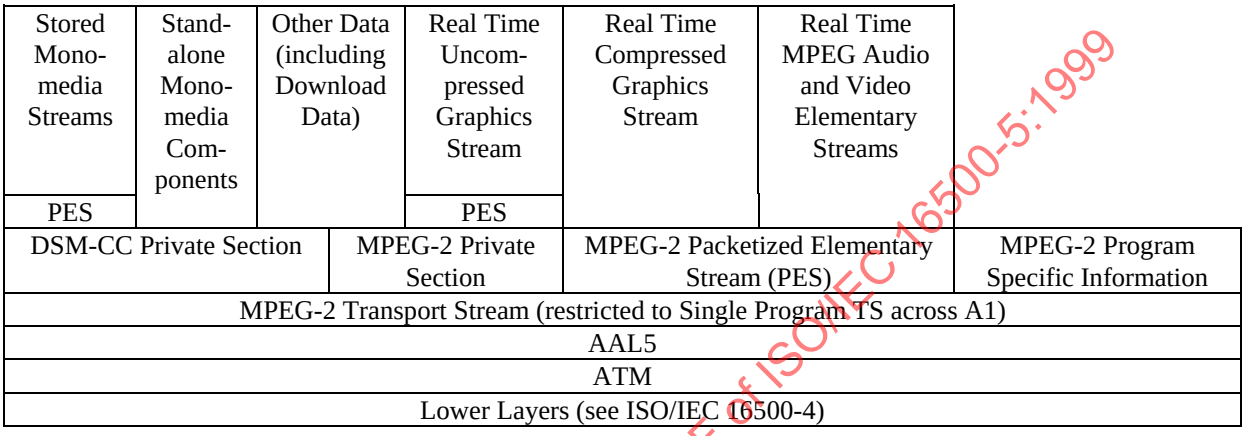


Figure 6.1 — MPEG TS protocol stack for an ATM-based transmission system

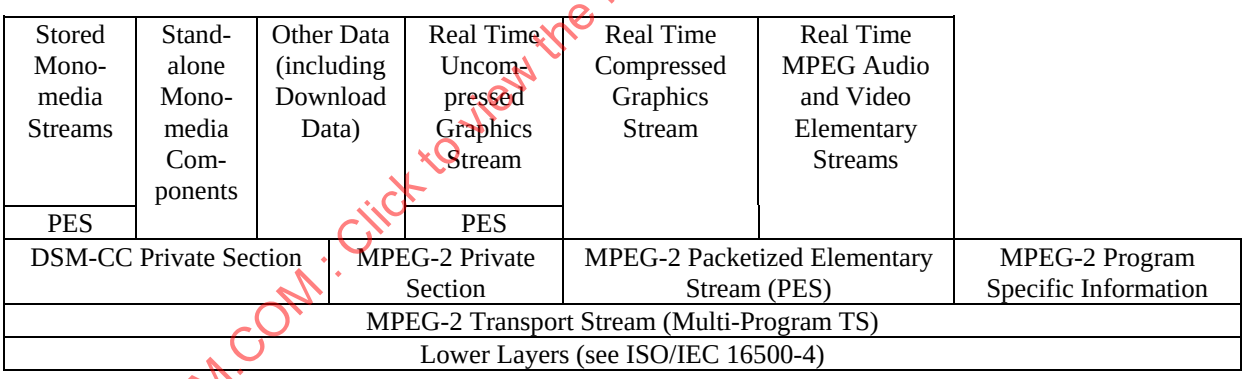


Figure 6.2 — MPEG TS protocol stack for the non-ATM based transmission system

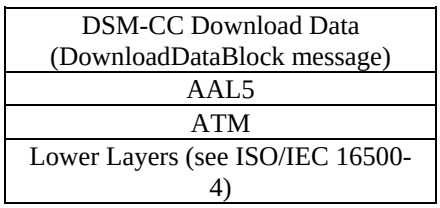


Figure 6.3 — Separate ATM VC protocol stack for S1 Download Data

6.3 Description of specific protocols

This section describes the specific protocols used in the protocol stacks identified in the section above. Wherever required the protocols layers are grouped by pairs in order to explicitly focus on the mapping issues between different protocols. Only the protocols which are specific to the S1 flow are described in detail, while the protocols which are common to other flows are described in a separate section.

6.3.1 MPEG-2 Packetized Elementary Stream (PES)

Defined in ISO/IEC 13818-1.

6.3.2 MPEG-2 Private Section

Defined in ISO/IEC 13818-1.

The choice of using the CRC form (private_section_syntax_indicator field set to 1, the “long form”) or “no CRC” for (private_section_syntax_indicator field set to 0, the “short form”) is not specified.

6.3.3 DSM-CC Private Section

The DSM-CC Private Section, DSMCC_section, is defined in ISO/IEC 13818-6, clause 9 “Transport”.

The choice of using the CRC, Checksum, or “no coverage” form has not been specified.

6.3.4 MPEG Elementary Streams

MPEG Video ES and MPEG Audio ES are content information specified in Parts 9 and 12 of this Specification. This includes Real Time Streams (but excludes Real Time Stream Graphics), as defined in ISO/IEC 16500-6. The syntax is also defined in ISO/IEC 11172 MPEG-1 Parts 2 and 3 and ISO/IEC 13818 MPEG-2 Parts 2 and 3.

6.3.5 Real time uncompressed graphics streams

Defined in ISO/IEC 16500-6. According to ISO/IEC 16500-6 and shown in Figure 6.2 and Figure 6.3, this data type is first encapsulated into MPEG-2 PES packets and then encapsulated into MPEG-2 Private Sections.

6.3.6 Real time compressed graphics streams

A format for graphics that may be used for subtitling; the format complies to ETS 300 743. This data type is encapsulated into MPEG-2 private PES packets as defined in ETS 300 743.

6.3.7 Stored monomedia streams

Stored Monomedia Streams are content types defined in ISO/IEC 16500-6 and may include MPEG Audio, Linear Audio (AIFF-C), MPEG Video, Uncompressed Graphics and Compressed Graphics. Stored Monomedia Streams are encapsulated into MPEG-2 PES packets.

The content type may be DSM-CC User-User Objects or pointers to these objects. DSM-CC User-User Objects may be carried in DSM-CC User-User Object Carousels (defined in ISO/IEC 13818-6, clause 8), which in turn are transmitted using the DSM-CC Data Carousel scenario for DSM-CC Download (defined in ISO/IEC 13818-6, clause 7). Alternatively, User-User Objects may be carried within an RPC reply (in which case, it is an S2 flow; see Section 7). When carried within an MPEG TS, this information shall first be encapsulated in DSMCC_sections per ISO/IEC 13818-6, clause 9.

The content type may be DSM-CC Download Modules (defined in ISO/IEC 13818-6, clause 7) or Images (Download Images are logically sub-divided into one or more Download Modules). Download Modules are delivered using one of the three DSM-CC download scenarios – Data Carousel, Flow Control, or Non-Flow Control. Note that Modules are actually divided into data blocks and transported as DownloadDataBlocks. When carried within an MPEG TS, this information shall first be encapsulated in DSMCC_sections per ISO/IEC 13818-6, clause 9.

6.3.8 Stand-alone monomedia components

Stand-alone Monomedia Components may be a still picture bitmap or a graphics bitmap or text, as specified in ISO/IEC 16500-4.

Stand-alone Monomedia Components may be carried by DSM-CC using the same mechanisms as for Stored Monomedia Streams; see subclause 6.3.7.

6.3.9 Other data

Other Data includes any type of information not defined elsewhere in subclause 6.3. Some examples are:

- Tunneled IP data over MPEG (see Section 7.2.3 of this specification).
- Application code; for the Application Format and the mapping to DSM-CC, see ISO/IEC 16500-6 subclause 9.
- Download Data. For carriage by DSM-CC the same mechanisms may be used as for Stored Monomedia Streams; see subclause 6.3.7.
- DVB Service Information, as defined in ETS 300 468. Service Information data is encapsulated into MPEG-2 private sections.
- ATSC Program and System Information, as defined in ATSC Standard A/65.

6.3.10 MPEG-2 Program Specific Information

Defined in ISO/IEC 13818-1. Examples of MPEG-2 PSI (Program Specific Information) are the Program Association Table, Program Map Table, Conditional Access Table, and Network Information Table.

6.3.11 MPEG-2 Transport Stream (TS)

Defined in ISO/IEC 13818-1. This is one of the two systems layers defined for MPEG-2.

6.3.11.1 Tunneling IP over MPEG

This section specifies how IP datagrams are carried over MPEG in the downstream direction to support the Internet Access with a delivery with a broadband broadcast network and a narrowband return channel.

6.3.11.1.1 Broadband MPEG TS encapsulation format and filtering

Datagrams are encapsulated in datagram_sections which are compliant to the DSMCC_section format for private data as defined in ISO/IEC 13818-6. The mapping of the section into MPEG-2 Transport Stream packets is defined in ISO/IEC 13818-1 MPEG-2 Systems.

The syntax of the datagram is fully aligned with the DVB specification for Data Broadcasting.

The syntax and semantics of the datagram_section are defined below.

Table 6.1 — Syntax of datagram_section

Syntax	No. of bits	Mnemonic
datagram_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
MAC_address_6	8	uimsbf
MAC_address_5	8	uimsbf
reserved	2	bslbf
payload_scrambling_control	2	bslbf
address_scrambling_control	2	bslbf
LLC_SNAP_flag	1	bslbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
MAC_address_4	8	uimsbf
MAC_address_3	8	uimsbf
MAC_address_2	8	uimsbf
MAC_address_1	8	uimsbf
if (LLC_SNAP_flag == '1') {		
LLC_SNAP()		
} else {		
for (j=0;j<N1;j++) {		
IP_datagram_data_byte	8	bslbf
}		
}		
if (section_number == last_section_number) {		
for (j=0;j<N2;j++) {		
stuffing_byte	8	bslbf
}		
}		
if (section_syntax_indicator =='0') {		
checksum	32	uimsbf
} else {		
CRC_32	32	rpchof
}		
}		

The semantics of the datagram_section are as follows:

table_id: this is an 8-bit field which shall be set to 0x3E (DSM-CC sections with private data [5]).

section_syntax_indicator: this field shall be set as defined by ISO/IEC 13818-6 [5].

private_indicator: this field shall be set as defined by ISO/IEC 13818-6 [5].

reserved: this is a 2-bit field that shall be set to '11'.

section_length: this field shall be set as defined by ISO/IEC 13818-6 [5].

MAC_address_[1..6]: this 48-bit field contains the MAC address of the destination. The MAC address is fragmented in 6 fields of 8-bits, labeled MAC_address_1 to MAC_address_6. The MAC_address_1 field contains

the most significant byte of the MAC address, while MAC_address_6 contains the least significant byte. Figure 6.4 illustrates the mapping of the MAC address bytes in the section fields. Note that the order of the bits in the bytes is not reversed and that the most significant bit of each byte is still transmitted first.

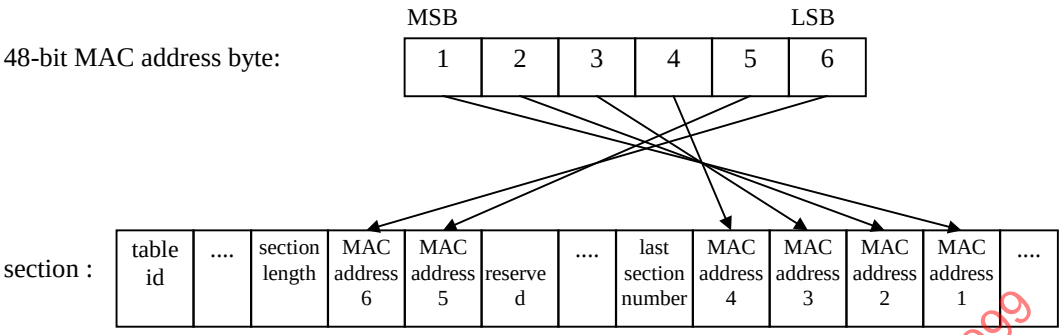


Figure 6.4 — Mapping of MAC address bytes to section fields.

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

payload_scrambling_control: this 2-bit field defines the scrambling mode of the payload of the section. This includes the payload starting after the MAC_address_byte_1 but excludes the checksum or CRC32 field. See Table 6.2. The scrambling method applied is user private.

Table 6.2 — Coding of the payload_scrambling_control field.

value	payload scrambling control
00	unscrambled
01	defined by service
10	defined by service
11	defined by service

address_scrambling_control: this 2-bit field defines the scrambling mode of MAC address in this section. See Table 6.3. This field enables a dynamic change of MAC addresses. The scrambling method applied is user private.

Table 6.3 — Coding of the address_scrambling_control field.

value	address scrambling control
00	unscrambled
01	defined by service
10	defined by service
11	defined by service

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

current_next_indicator: this is a 1-bit field. It shall be set to a value of '1'.

section_number: this is an 8-bit field. If the datagram is carried in multiple sections, then this field indicates the position of the section within the fragmentation process. Otherwise it shall be set to zero. When carrying IP datagrams, each datagram shall be encapsulated to exactly one section and the value of the section_number shall be set to zero always (i.e., the MTU for IP is 4080 bytes).

last_section_number: this 8-bit field shall indicate the number of the last section that is used to carry the datagram, i.e., the number of the last section of the fragmentation process.

LLC_SNAP: this structure shall contain the datagram according to the ISO/IEC 8802-2 [11] Logical Link Control (LLC) and ISO/IEC 8802-1a SubNetwork Attachment Point (SNAP) specifications. If the payload of the section is scrambled (see payload_scrambling_mode), these bytes are scrambled.

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

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

checksum - This field shall be set as defined by ISO/IEC 13818-6 [5]. It is calculated over the entire datagram_section.

CRC_32 - This field shall be set as defined by ISO/IEC 13818-6 [5]. It is calculated over the entire datagram_section.

6.3.12 ATM Adaptation Layer 5 (AAL5)

Defined in ITU-T Rec. I.363.5.

6.3.12.1 Encapsulation of MPEG-2 Transport Streams for distribution to the STU

The mapping of Transport Streams into AAL5 which are intended to cross the A1 reference point is defined in af-saa-0049.001, The ATM Forum, “Audiovisual Multimedia Services: Video on Demand Specification 1.1”. The default mapping case of “N=2” shall be used: Two MPEG-2 TS packets shall be mapped onto an AAL5 CPCS-SDU (common part convergence sub-layer – service data unit), which gives a 376 octet size for the AAL5 CPCS-SDU. In the case where the MPEG program contains an odd number of TS packets (such as at the end of a MPEG program where only one TS packet remains), the last AAL5 PDU (protocol data unit) shall contain only one MPEG-2 TS packet.

The AAL5 CPCS-SDU, together with the AAL5 CPCS-PDU trailer of 8 octets requires 384 octets and maps into exactly 8 ATM cells with zero CPCS padding octets.

In the case where the AAL5 PDU contains only one MPEG-2 TS packet, the AAL5 CPCS PDU will include the AAL5 CPCS-SDU, the 8 octets trailer and 44 CPCS padding octets.

A “Null” Service Specific Convergence Sub-layer shall be required.

6.3.12.2 Encapsulation of MPEG-2 Transport Stream for contribution or broadcast distribution not intended to be delivered to the STU

In order to meet the higher quality of service requirements for the delivery of Transport Streams as contribution to SPS entities or distribution to the access network (excluding ATM across A1 to the STU), AAL5 encapsulation shall be “PCR aware” at the A11, A9*, and A9 reference points. “PCR aware” is defined as guaranteeing that if a TS packet contains an Program Clock Reference, it shall only be in the second of the two TS packets within the AAL5 PDU (see the previous sub-clause). Otherwise, the AAL5 PDU shall be truncated to 5 cells and only contain one TS packet (plus the 8 octet trailer and 44 CPCS padding octets).

For high-quality broadcast services (defined in DAVIC 1.3.1a Part 4) over ATM networks, “PCR aware” encapsulation shall be used, except as follows:

- Constant bit rate MPTS, where all programs are CBR-encoded MPEG, “PCR aware” ATM encapsulation is optional.
- Constant bit rate MPTS, where at least one of the programs is VBR-encoded MPEG, shall not use “PCR aware” ATM encapsulation.

Variable bit rate MPTS shall not be supported by ISO/IEC 16500.

7. S2 flow: high and mid layer protocols

7.1 S2 flow description

As defined in ISO/IEC 16500-1 subclause 7, the S2 information flow is for control information flow from an application service layer source object to a peer destination object. ISO/IEC 16500 has chosen the International Standard MPEG-2 Digital Storage Media Command & Control (DSM-CC) for the high level S2 interface between the STU and Server (ISO/IEC 13818-6 clause 5, User-to-User Interface), the control of binary object and other data type information download (ISO/IEC 13818-6 clause 7, User-Network Download Protocol), and the carriage of IP-encapsulated S2 information over an MPEG Transport Stream (ISO/IEC 13818-6 clause 9, Transport).

The description of the service domain interfaces organizes the service domain around the objects which are visible to the set-top device. The text which follows then lists the interfaces which these objects export. The interfaces are those of ISO/IEC 13818-6 MPEG-2 DSM-CC specification, clause 5 User-to-User Interface, and clause 8 Normal Play Time, Stream Mode and Stream Events.

Transport of Normal Play Time, Stream Mode and Stream Events is specified in Section 9.2.8 "NPT Time Stamps and DSM-CC Descriptors" of ISO/IEC 13818-6.

7.2 Overview of protocol stacks

7.2.1 Download control

When S2 consists of "download" control information, the information shall be carried either within a separate channel as shown in Figure 7.1 (separate ATM Virtual Channel) and in Figure 7.2 (separate channel over, e.g., ISDN or PSTN), or in a "high bandwidth" SPTS or MPTS (where applicable) as shown in Figure 7.3 and Figure 7.4 (the latter is for the Data Carousel scenario). In the cases where an SPTS or MPTS downstream is used, the associated (i.e., in the same domain) upstream channel shall be as shown in Figure 7.1.

The selection of the option is done during service activation time using DSM-CC resource descriptors and binding lists (see ISO/IEC 16500-2).

DSM-CC Download Control
TCP
IP
AAL5
ATM
Lower layer protocols

Figure 7.1 — Separate ATM VC protocol stack for Download Control

In the case of enhanced broadcast, data via the broadcast channel shall be transported according to the protocol stack in Figure 7.2.

DSM-CC Download Control
TCP
IP
PPP(MP)
Lower layer protocols

Figure 7.2. — Download Control across the interaction channel for Enhanced Broadcast services

DSM-CC Download Control
TCP
IP
DSM-CC Private Section
Lower layer protocols (see Figure 6-1 and Figure 6-2)

Figure 7.3 — MPEG-2 TS protocol stack for Download Control, for cases other than the Data Carousel Scenario[1]

DSM-CC Download Control (DownloadInfoIndication message)
DSMCC Private Section
Lower layer protocols (see protocol stacks in Figure 6-1 and Figure 6-2)

Figure 7.4 — MPEG-2 TS protocol stack for Download Control, for the Data Carousel Scenario case

7.2.2 User-User interaction

The protocols to support user-to-user interaction for ATM networks are defined in Figure 7.5 and for MPEG Transport Stream networks in Figure 7.6. In the case where an MPEG TS downstream is used, the associated (i.e., in the same domain) upstream channel shall be as shown in Figure 7.5.

DSM-CC User-User Interface
Object Management Group - Common Data Representation (OMG-CDR)
Object Management Group – GIOP/IIOP (OMG-GIOP/IIOP)
TCP
IP
AAL5
ATM
Lower layer protocols (see ISO/IEC 16500-4)

Figure 7.5 — ATM protocol stack for User to User Interaction

IP shall be carried over ATM according to subclause 11.3 (IP).

For compression of the IP Address and Control Fields (RFC 1332), the following protocols shall be supported in the PPP data link layer (RFC 1700):

002l Internet Protocol

002d Van Jacobson Compressed TCP/IP

002f Van Jacobson Uncompressed TCP/IP

For the PPP link, the following configuration shall be supported as recommended for PSTN type links (Appendix A to RFC 1662):

Async Control Character Map

Magic Number

Address and Control Field Compression

Protocol Field Compression

[1]DSM-CC Download Control messages are DownloadInfoRequest, DownloadInfoResponse, and DownloadCancel

DSM-CC User-User Interface
Object Management Group - Common Data Representation (OMG-CDR)
Object Management Group – GIOP/IIOP (OMG-GIOP/IIOP)
TCP
IP
DSM-CC Private Section
Lower layer protocols (see Figure 6-2)

Figure 7.6 — MPEG TS downstream protocol stack for User to User Interaction

For enhanced broadcast services, DSM-CC User to User interaction is supported over the broadcast channel as depicted in Figure 7.7 and over the interaction channel as specified in Figure 7.8.

DSM-CC U-U Primitives	
DSM-CC Object Carousels	
MPEG-2 DSM-CC Private Section	
MPEG-2 TS	

Figure 7.7 — U-U Primitives across the broadcast channel (ii)

Note that DSM-CC sections are a category of MPEG Private sections and are defined in DSM-CC ISO/IEC 13818-6 DSMCC.

DSM-CC U-U
OMG-CDR
OMG-GIOP/IIOP[2]
TCP
IP
PPP(MP)
Lower layer protocols

Figure 7.8 — User-User Interaction (including Data transfer) across the interaction channel

PPP (MP) - Multilink Point-to-point protocol. Multilink PPP allows the use of multiple PPP links to be aggregated to provide a single transmission link for use by the IP layer. It is an extension to PPP and is inter-operable with PPP. (IETF RFC1717).

7.3 Description of specific protocols

The description does not attempt to re-define the DSM-CC interfaces. The readership should refer to the DSM-CC specification for the exact functions. Rather, the description classifies each interface with respect to its adoption in this specification. If there are comments on an interface, the following identification will be used:

- Pragmatics: The interface and the semantics are clear. The comment relates to implementation concerns outside the scope of DSM-CC but within the scope of DAVIC. The comment is the implementation agreement.

In each case, reliance is made on standard protocols, and these are indicated. Where additional details on how these standards are to be applied, these are indicated.

7.3.1 DSM-CC option choices summary

In some instances, DSM-CC allows a choice of options. DAVIC has selected the following DSM-CC options:

[2] See clauses 7.3.1 and 7.3.2 of part 7 for an exact specification of the options needed in a DAVIC compliant system.

1. Universal networked objects (i.e., GIOP/IIOP) as the Remote Procedure Call (RPC).
2. Common Data Representation (CDR) as the data encoding for User-to-User messages.
3. Inter-operable Object Reference as the encoding of inter-operable addressing and identification of objects.
4. IIOP Protocol Profile as a required object reference encoding. In the case of enhanced broadcast services, BIOP is a required object encoding reference for object carousels. BIOP is specified in DSM-CC ISO/IEC 13818-6 DSMCC.

1. IOP Inter-operable Object Reference as a required object reference encoding.
2. NPT Descriptors Must be Placed in the MPEG-2 Delivery Stream at intervals of no greater than 1 second.
3. DAVIC will use the interface definition module name *DAV*.
4. The DAVIC Stream Service (DAV::Stream) is composed of the DSM::Stream, DSM::Event, and DSM::Service interfaces.
5. DAVIC will implement a mini- Interface Repository Application Service which is composed of DSM::Interfaces and DSM::Directory. This service will be used to verify interface definitions and serve as a mechanism for making application service interfaces available.

7.3.2 Remote Procedure Call

DAVIC compliant systems require an inter-operable protocol stack for the presentation and session layer that will transport User-User primitives between STU and Server, or Server and Server. The definition of DAVIC upper layer protocols is identical to the OMG-GIOP/IIOP specification in CORBA 2.1. This solution comprises:

a) a framework which allows nodes to select a protocol stack, or interpose an object which translates protocol stacks,

b) conventions to encode the message payload at the presentation level (the Common Data Representation),

c) a message set for request/response at the session level,

d) a pervasive protocol solution at the transport level and network level.

7.3.3 Download

The DSM-CC Download protocol (ISO/IEC 13818-6 clause 7) is used to satisfy the needs listed above. The protocol may be thought of as two flows. The first flow is control which negotiates download parameters. The second flow is the download data, which the download service transmits to the set top, and the acknowledgments which set top returns to the download service. The distinction between the flows anticipates the allocation of different flows to different network resources. As described further in ISO/IEC 16500-2, the architecture allows the download control to be bound to a connection of modest bandwidth, while the download data is bound to another connection with serious bandwidth. The connection which supports the media stream, for example, may embed the download data as private data. The discussion of the structure which binds interfaces to network resources describes the technique further.

The service decides what to download from the set top profile. The set top provides the profile first session protocol message. The discussion on the set top profile describes the contents of the set top profile further.

7.3.4 User-User interaction

User-user interaction, where an element of the Service Provider System represents one user and an element of the Set-Top Unit another, is governed by the protocols specified in ISO/IEC 13818-6 MPEG-2 Digital Storage Media Command and Control (DSM-CC). The interactions are listed below as particular functions, grouped according to the Service Provider System element to which they pertain. Each function is accompanied by a citation of the clause of the DSM-CC standard that dictates its operation, and by any constraints on the use of that standard in the DAVIC context. Unless otherwise noted, the standard operation and the constraints are normative.

7.3.5 Service Gateway functions

7.3.5.1 Service configuration phase

The service configuration phase is the exchange between the Session Gateway, which resides in the Service Domain, and the Session & Resource Manager, which resides in the network. Note that this specification refers to the element which terminates the session protocol on the service side as the ses(s) and the element which terminates the session protocol on the network side as the ses(n).

The DSM-CC Configuration Protocol (see ISO/IEC 13818-6 clause 3) is used to implement the service configuration phase. These subclauses assume that the element which implements the configuration protocol and the element which implements the session protocol are distinct elements. An interface outside the scope of this specification may exist between the configuration element and the session element on the service side, as well as between the configuration element and the session element on the network side. In both cases, the session element forwards values to the configuration element before the configuration phase, and the configuration element returns values to the session element after the configuration phase. The specification of the interactions, however, is not defined by this specification.

The configuration element on the service side understands the network address to associate with the configuration element on the network side through convention. The configuration element on the service side initiates the exchange. The data which it provides in the first message is:

- Configuration Network Address: This is the network address to which the DSM-CC network shall send DSM-CC U-N configuration messages.
- Session Network Address: This is the network address to which the DSM-CC network shall send DSM-CC U-N session messages.
- Session Profile: The session profile shall indicate the subset of DSM-CC U-N session primitives supported by the server.
- Resource Profile: The resource profile shall indicate the subset of DSM-CC U-N resources that the server expects the network to provision for a service.
- Default Service: The default service is defined as the service to be selected if the set-top does not indicate the desired service during the DSM-CC U-N session establishment.
- When the set top establishes a DSM-CC U-N session (see ISO/IEC 13818-6 for the definition of DSM-CC U-N session), it provides a client profile through the session protocol. The client profile articulates features of the device which relate to inter-operation, or which clarify what data to download. DSM-CC User Compatibility (ISO/IEC 13818-6 clause 6) describes a structure which allows the set top to describe itself. The set top shall at least define its system hardware and its system software. In both cases, it shall provide the specifier field which identifies the organization which defines the semantics. The schema for the specifier is IEEE OUI. The system hardware shall be the first declaration in the profile and the system software shall be the second declaration. The client may provide other declarations. The structure provides the vocabulary to articulate application profiles.

The configuration code on the network side returns the data shown below. It initializes the context of the ses(s) element (also known as the Session Gateway) in the service domain.

- Session & Resource Manager Network Address: The configuration phase initializes the network address to which the service side will send session protocol.
- Session Profile: The profile identifies the subset of the session protocol which the Session & Resource Manager supports.
- Resource Profile: The profile identifies the resources which the Session & Resource Manager may allocate for a service. The data includes the type field of the resource descriptor.
- Resource Allocation Option: There are two options. The service may provision network resources and inform the network, or the service may request that the network establish network resources on its behalf.

- SessionId Option: The SessionId is a unique value which identifies a session. The Session & Resource Manager instructs the ses(s) element as to whether it is to assign the value.
- Maximum Session Forward Count: The protocol which relates to session forward includes a count which increments with each attempt to forward the session. This is the maximum count.
- Maximum Message Retry Count: The session elements re-send a message up to this count (see ISO/IEC 13818-6 for the state machine definition).
- Message Time-out. The session protocol elements re-send a message if the response does not arrive before the time-out.

7.3.5.2 Browse and select services

Function: The DSM::ServiceGatewayUU and DSM::Directory interfaces provide functions, e.g., list(), which allow a client to browse the available services and select a service with its name.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: None

7.3.5.3 Install and de-install services

Function: The DSM::ServiceGatewayUU and DSM::Directory interfaces provide functions which allow a service to install itself. The service provides its name with bind() function. The companion unbind() function removes the service from the Service Gateway.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5

7.3.5.4 Mandatory User-Network Session Gateway functions

Function: The session protocol includes messages which establish the session. The request to establish a session includes fields to select a service to launch for the session. The discussion below specifies interface, semantics, and pragmatics which integrate the session protocol with the life cycle of the services found in the service domain.

Citation of normative standard: ISO/IEC IS 13818-6 IS, clause 4.

Constraints: Pragmatics: The DSM-CC specification does not specify whether a specific service supports the session protocol (see below for further description of the session protocol). Since the Service Gateway is, in practice, the first service which the set-top device encounters in a service domain. The convention is that the Service Gateway must support the session protocol. It also exports the above resource interface to support other services in the service domain.

Constraints: Pragmatics: The configuration phase shall install into the set-top device the ServiceId of the default Service Gateway for the default service domain. The set-top device uses the default value in its first session establishment attempt.

7.3.5.5 Client profile (User-Network related)

Function: The client profile articulates the device features which a service must understand in order for the service to determine if it may inter-operate with the client.

Citation of normative standard: ISO/IEC 13818-6 IS clause 6

Constraints: Pragmatic: It is necessary to adopt a convention about how to forward the profile from the client to the service. The convention is that the client is to place the profile data in the user data field during the session establishment phase. The session protocol of DSM-CC provides the mechanism. If a service launches another service (as is true for the Service Gateway) the service shall provide the client profile during the launch phase.

7.3.5.6 Life cycle of users' instances of Service Gateway

Function: The DSM::Base interface relates to object life cycle. The destroy() function allows a client to delete a single object instance. The close() function allows a client to indicate that it will no longer need to communicate with an object.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

7.3.6 Application service functions

7.3.6.1 Life cycle of service instance

Function: The DSM::Base interface, which all service objects inherit, provides functions to control the life cycle of multiple services.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: None

7.3.7 Access control

Function: The DSM::Access interface provides attributes for Access control, such as access permissions.

Pragmatics: It is used in conjunction with the DAVIC Domain, Group and Member interfaces.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: None

7.3.8 Stream service functions

7.3.8.1 Individual stream control (Pause, Resume)

Function: The DSM::Stream interface, which the DAVIC Stream object inherits, provides functions, such as pause() and resume(), to control the advance of the media stream.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: None

7.3.8.2 Stream event

Function: The DSM::Event interface, which the DAV::Stream object inherits, provides functions to express interest in certain events, which the Stream object later places in the media stream.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: Pragmatics: The DAVIC design mandates that a Stream object provide the Event interface.

7.3.8.3 Cascading streams

Function: The DSM::QStream interface enables cascading of stream control to the next stream object to transport a media stream. The interface is expected to be used to implement play lists.

Citation of informative standard: ISO/IEC 13818-6 IS, Informative Annex K, Stream Playlist.

7.3.8.4 Scale control

See subclause 7.3.12, Karaoke.

Function: The resume() function of the stream interface allows the client to select the stream scale value, that is the rate at which to transport the stream. This operation may be invoked while the state machine is in Transport mode.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: None

7.3.9 File service functions

7.3.9.1 File access

Function: The DSM::File interface provides read() and write() functions, subject to access control.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: None

7.3.10 Data base service functions

7.3.10.1 Data base access

Function: The DSM::View interface provides basic data base functions.

Citation of normative standard: ISO/IEC 13818-6 IS, clause 5.

Constraints: Scope: The interface is out of scope for STU to Server interface, but may be used for server to server communication.

7.3.11 Use of the DSM-CC User-to-User interface in the DAVIC distribution profile

The ISO/IEC 16500 Distribution profile does not mandate the use of a back-channel. Nevertheless, it is desirable in order to achieve maximum inter-profile interoperability to provide the DSM-CC User-to-User interfaces also to applications that run under the Distribution profile. DSM-CC provides a mechanism to do so in clause 11 “User-to-User Object Carousels”.

Function: The provision of (constrained versions of) the DSM-CC Base, Access, Directory, Service, File, Stream, and Event interfaces in a pure-broadcast environment.

Citation of normative standard: ISO/IEC 13818-6, clause 11 and Annex F.

Constraints: None

7.3.12 Application note: Use of ISO/IEC 16500 for karaoke (high-layer protocol aspects)

DAVIC tools may be used to support Karaoke, at least at a basic level. An MPEG may carry the video, audio, and subtitle streams of a selection chosen from a menu in any of the normal ways. This section pertains to the additional functions of pitch and tempo control.

7.3.12.1 Pitch control

Pitch control shall be provided solely by the Set-Top Unit. No support by the Service Provider System is mandated.

7.3.12.2 Tempo control

Tempo control relies on the DSM-CC Scale facility, whereby the playback speed is scaled by the scaling factor.

Citation of normative standard:

Constraints:

1. The scaling factor shall be represented by the fraction A/B where A and B are positive integers.

2. The tempo shall be changed seamlessly with no loss or duplication of content.

3. Bounds on the amount of scaling may be conveyed by either party at session establishment via private data.

4. Changes in tempo shall be conveyed to the Set-Top Unit in the data stream so that it may play the stream back at the proper rate.

Note: Increases in tempo may affect the amount of network bandwidth required to transmit the stream. Actions dealing with the consequences of tempo increase are beyond the scope of ISO/IEC 16500.

7.3.12.3 Language selection

Selection of different languages for either the audio or subtitle stream may be desired. At present this is possible by downloading a menu of choices, each of which would constitute an elementary stream in the MPEG Transport Stream.

Citation of normative standard: MPEG TS; Download protocol

7.3.13 Application note: network congestion control for enhanced broadcast

Because of the potentially large numbers of (nearly) simultaneous transactions that may occur during a popular broadcast, actions are necessary to ensure the network may cope with the demands placed upon it at that time.

Two methods may be used to avoid network congestion. The network operator may wish to implement call distribution management, using intelligent network control for example. A second possibility is to provide data to terminals to reduce the burstiness of interaction channel use. A data set (in the form of an application control object) may be downloaded into the STU in order to provide a randomizing element which may reduce the number of simultaneous interaction channel activations.

To allow for the second kind of traffic shaping described above, an object implementing two mechanisms is defined:

- reduce the number of callers if necessary (the threshold randomizer)
- spread the responses in time (the time randomizer)

The first point is achieved by signaling a threshold to the receiving set-top box. If the user wants to participate in an interactive program, the set-top generates a random 16 bit number. Only if this number is higher than the threshold, the box will place a call to the service provider. The second point is implemented by giving start and stop times in between which a set-top box may place it's call. This should happen at a random moment in this interval. If the service provider is unreachable, the set-top may try again after a specified time. Again a random time in the resulting interval should be chosen. This minimum waiting time after a try is introduced to prohibit traffic explosions when there's very little response time left.

The `giveNextDialTime` function of the `trafficShaping` object implements the above mentioned mechanisms. If the `giveNextDialTime` returns true then the time when the set-top may dial out is given in the variable `dialTime`. Otherwise false is returned and the set-top is not allowed to call out. A description of the semantics of the variables involved is also given.

```
typedef short date;
typedef char[3] time;
struct dateTime
{
  date aDate;
  time aTime;
}
interface trafficShaping
{
  attribute short numResponseThreshold;
  attribute dateTime responseStartTime;
  attribute dateTime responseEndTime;
  attribute time retryWaitTime;
  attribute short responseAttemptLimit;
```

```
  // next time:  result false -> not allowed any more
  boolean giveNextDialTime (out dateTime dialTime);
};
```

numResponsesThreshold: this 16-bit field signals the threshold above which a set-top unit may initiate a call to a service provider. The set-top unit should compare this value with an internally generated 16-bit unsigned integer random number.

responseStartTime: this 40-bit field contains the start time of the responses in Universal Time, Coordinated (UTC) and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit Binary Coded Decimal (BCD) (see also Annex C of ETS 300 468).

EXAMPLE 1: 93/10/13 12:45:00 is coded as '0xC079124500'

responseEndTime: this 40-bit field contains the end time of the responses in UTC and MJD. This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit BCD.

retryWaitTime: a 40-bit field containing the date and time after which a set-top box is allowed to retry. It has the same coding as the response_end_time

responseAttemptLimit: this 8-bit field signals how many times a set-top box may try to call to a service provider. A value of zero (0) means unrestricted calling.

7.3.14 A10 content transfer interface

Function: The interface provides functions to copy(), modify(), move() content objects between nodes indicated by pathnames. DSM::Directory operations get() and bind() are invoked as sub-routines of the Content Interface operations.

Citation of normative standards: ISO/IEC 13818-6 clause 5, Directory interface

Constraints: None

7.3.14.1 IDL syntax

```
Module DAV {
        enum NotFoundReason { missing_node, not_context, not_object };
        exception NotFound {
                NotFoundReason why;
                CosNaming::Name rest_of_name;
        };
        exception AlreadyBound {};
        exception CannotProceed {
                CosNaming::NamingContext cxt;
                CosNaming::Name rest_of_name;
        };
        exception InvalidName{};
        exception NoAuth{
                unsigned long minor;
                unsigned long completed;
                sequence<octet> authData;
        };
        interface Content : DSM::Directory {
                const DSM::AccessRole copy_ACR = DSM::WRITER;
                const DSM::AccessRole modify_ACR = DSM::WRITER;
                const DSM::AccessRole move_ACR = DSM::WRITER;
                const DSM::AccessRole load_ACR = DSM::WRITER;
                void copy (in CosNaming::Name dest, in CosNaming::Name source)
                        raises(NotFound, AlreadyBound,
                                CannotProceed, InvalidName,
                                NoAuth);
```



```
                void modify (in CosNaming::Name dest, in CosNaming::Name source)
                        raises(NotFound, CannotProceed,
                                InvalidName, NoAuth);
                void move (in CosNaming::Name dest, in CosNaming::Name source)
                        raises(NotFound, AlreadyBound,
                                CannotProceed, InvalidName,
                                NoAuth);
                void load (in CosNaming::Name package)
                        raises(NotFound, CannotProceed,
                                InvalidName, NoAuth);
        };
};
```

7.3.14.2 Semantics

The Content Interface extends the DSM-CC Directory Interface to provide common application level content manipulation operations. The arguments to the Content Interface operations are CORBA CosNaming::Names. This enables implementations the freedom to choose location and instances of content. The operations are sent to a content object which is in the hierarchy of the **dest** parameter. The path Name of this parameter shall extend from the target object of this invocation. The path Name of the other parameter (**source**) will extend from the client's local Service Gateway. The transfer may occur within one Service Provider or across Service Provider Domains.

copy() is used to copy a content object from one Directory to another. A copy of the **source** object will be created, and the bound at the **dest** location.

modify() is used to replace existing content with a new source. A copy of the **source** object will be created, and the rebound at the **dest** location.

move() is used to transfer content from one location to another. The **source** is unbound from the source parent directory, and bound to the **dest** location.

load() is used to unpack a Content Package. The load() is sent to a Content Service, which has in its name graph a content package. The **package** argument is a path name the extends from the Content Service to the content package to load. The Content Package encoding is specified by ISO/IEC 16500-6. The ISO/IEC 16500 Information Representations shall be represented as objects with a path Name, load instructions (with operation identifier as specifier by GIOP/IIOP), and the content data itself. These shall be extracted, and the load instruction(s) shall be performed on them. Unrecognized Information Representations will not be processed.

7.4 Distributed Server API

The following IDL specifies the Domain-related interfaces enabling the encapsulation of the Distributed Server. It represents generic classes for membership in domains and groups, which may be used by Service Providers, Content Providers and Applications Services for enrolling end-users and controlling access and invocation rights.

7.4.1 Member interface

The Member interface provides an object that represents capabilities information about a Principal member of a Domain. The attribute Privs uses the DSM-CC Permissions structure to hold the privileges of the member, e.g., which READER groups the member is in, which WRITER groups, etc. This format enables an efficient match with the Perms attribute of DSM-CC objects that inherit DSM::Access interface. The match shall determine whether a member has privileges to invoke an operation on an object. The PrincipalId is the unique identifier of the member within the Domain, and is used in the RPC header for messages sent to objects in the Domain. A member with MANAGER privileges may invoke grant() or revoke(), to set the Access Role of the Member to a valid DSM-CC Access Role.

7.4.1.1 IDL syntax

```
Module DAV {
        exception INV_ACCESS_ROLE{};
        interface Member : DSM::Base, DSM::Access {
```

```
                attribute DSM::Access::Perms_T Privs;
                readonly attribute DSM::Principal PrincipalId;
                typedef sequence<DSM::AccessRole> AccessRolesList;
                readonly attribute AccessRolesList AccessRoles;
                //
                const DSM::AccessRole Privs_get_ACR = DSM::MANAGER;
                const DSM::AccessRole Privs_put_ACR = DSM::MANAGER;
                const DSM::AccessRole PrincipalId_get_ACR = DSM::MANAGER;
                const DSM::AccessRole AccessRoles_get_ACR = DSM::MANAGER;
                const DSM::AccessRole grant_ACR = DSM::MANAGER;
                const DSM::AccessRole revoke_ACR = DSM::MANAGER;
                void grant (in DSM::AccessRole acr)
                        raises(INV_ACCESS_ROLE);
                void revoke (in DSM:: AccessRole acr)
                        raises(INV_ACCESS_ROLE);
        };
};
```

7.4.1.2 Semantics

The DSM::AccessRole progresses from lowest privilege to highest privilege as follows: READER, WRITER, BROKER, MANAGER, OWNER. A WRITER has both READER and WRITER privileges. A BROKER has BROKER, WRITER and READER privileges. An OWNER has OWNER, BROKER, WRITER and READER privileges. The creator of an object has OWNER privileges for that object. A MANAGER has all privileges on an object. In order to invoke an operation on an object, the Member must have the required Access Role (as specified by the operation), and must also be a member of one of the corresponding Access Role groups of that object. DSM::Base is inherited and used to destroy() the Member.

grant() is used to promote a Member to a higher Access Role.

revoke() is used to demote a Member to a lower Access Role.

7.4.2 Group interface

The Group interface provides the ability to establish multiple groups in a Domain, with Members in each group.

7.4.2.1 IDL syntax

```
module DAV {
        interface Group : DSM::Base, DSM::Access {
                typedef sequence<Member> MemberList;
                const DSM::AccessRole Role_get_ACR = DSM::MANAGER;
                const DSM::AccessRole addMember_ACR = DSM::MANAGER;
                const DSM::AccessRole addMemberList_ACR = DSM::MANAGER;
                const DSM::AccessRole removeMember_ACR = DSM::MANAGER;
                readonly attribute DSM::AccessRole Role;
                void addMember (in Member aMember);
                void addMemberList (in MemberList members);
                void removeMember (in Member aMember);
        };
};
```

7.4.2.2 Semantics

The Group interface is used to enable a Domain to have multiple Groups with Members in each. DSM::Base is inherited and used to destroy() the Group.

addMember() adds a Member to a Group.

addMemberList() adds multiple Members at one time to a Group.

removeMember() removes a Member from a Group.

7.4.3 Domain Interface

The Domain Interface enables the establishment of a boundary around a set of objects that share a common set of attributes. The Domain may be an administrative domain, a security domain, a conference, or other. The domain may be a combination of types. The DAVIC Service Provider Domain is both an Administrative and Security Domain. When inherited with a DSM::Directory, the Domain Interface establishes a containment for the complete Directory graph. A client with MANAGER privileges may establish Groups for and enroll Members to the Domain. Members with READER privileges may gain access to other Members or Groups in the Domain.

Note that MANAGER privileges means that the client has MANAGER or higher Access Role and is a member of one of the MANAGER groups of the Domain object. Also note that READER privileges means that the client has READER or higher Access Role and is a member of one of the READER groups of the Domain object.

7.4.3.1 IDL syntax

```
Module DAV {
        typedef sequence<string> Roster;
        interface RosterIterator {
                const DSM::AccessRole next_n_ACR = DSM::READER;
                const DSM::AccessRole destroy_ACR = DSM::READER;
                void next_n (in unsigned long count, out Roster members);
                void destroy();
                };
        exception NAME_TAKEN{};
        exception INV_NAME{};
        exception NAME_NO_EXIST{};
        exception NOT_A_PEER{};
        // Domain will be typically be inherited with Directory, which inherits Access
        interface Domain {
                typedef DSM::opaque GroupToken;
                typedef unsigned short type;
                typedef sequence<type> CategoryList;
                typedef sequence<string> GroupList;
                const type security = 1;
                const type admin = 2;
                const type conference = 3;
                readonly attribute CategoryList Category;
                //
                const DSM::AccessRole Category_get_ACR = DSM::MANAGER;
                const DSM::AccessRole newMember_ACR = DSM::MANAGER;
                const DSM::AccessRole newGroup_ACR = DSM::MANAGER;
                const DSM::AccessRole findMember_ACR = DSM::BROKER;
                const DSM::AccessRole findGroup_ACR = DSM::BROKER;
                const DSM::AccessRole listMembers_ACR = DSM::READER;
                const DSM::AccessRole listGroups_ACR = DSM::READER;
                const DSM::AccessRole resolveMembers_ACR = DSM::READER;
                const DSM::AccessRole resolveGroups_ACR = DSM::READER;
                void newMember (in string nameId,
                                        out DSM::Principal aPrincipal,
                                        out Member aMember)
                        raises(INV_NAME, NAME_TAKEN);
                void newGroup (in string nameId, in DSM::AccessRole acr,
                                        out GroupToken aToken, out Group aGroup)
                        raises(INV_NAME, NAME_TAKEN);
                void findMember (in string nameId, out Member aMember)
                        raises(NAME_NO_EXIST);
                void findGroup (in string nameId, out Group aGroup)
                        raises(NAME_NO_EXIST);
                void listMembers (in DSM::u_long count,    //READER
                                        out Roster members, out RosterIterator it);
```

```
                void listGroups (out GroupList groups);       //READER
                Object resolveMember(in string nameid)       //READER
                        raises(NAME_NO_EXIST, NOT_A_PEER);
                Object resolveGroup(in string nameid)         //READER
                        raises(NAME_NO_EXIST);
        };
};
```

7.4.3.2 Domain Interface semantics

Exceptions: INV_NAME indicates the characters or length of the NameId is not valid. NAME_TAKEN indicates the proposed nameId already exists in this Domain. NAME_NO_EXIST indicates the nameId is not present in this Domain. NOT_A_PEER indicates the Member is not capable of responding to requests.

Domain newMember() creates a new Member object with a unique nameId and PrincipalId in the given Domain.

Domain newGroup() creates a new Group with a unique nameId and AccessRole.

Domain findMember() will return a Member given a nameId.

Domain findGroup() will return a Group given a nameId.

Domain listMembers() will list the NameIds of Members in this Domain. The RosterIterator interface is used to iterate through very long lists.

Domain listGroups() will list the NameIds of Groups in this Domain.

Domain resolveMember() will return an addressable peer object. The exception NOT_A_PEER indicates the Member cannot receive messages (The Member is likely a Client only)

Domain resolveGroup() will return an addressable multi-point object for sending messages to a Group of peers.

7.4.3.3 RosterIterator Semantics

The RosterIterator is used as a follow-up to Domain listMembers(). It provides a mechanism similar to CosNaming BindingIterator to iterate through very long Member lists

RosterIterator next_n returns the next portion of a Member list

RosterIterator destroy() destroys the RosterIterator.

8. S3 flow: high and mid-layer protocols

8.1 S3 flow description

The S3 flow is a bi-directional flow used for the exchange of control information related to the service to be provided. Server, STU and delivery system (specific entity in the delivery system, depending on the specific service) may all be involved in the exchange of information. The S3 flow is not transparent to the delivery system at any layer. A number of procedures, required for different services, are classified as requiring an exchange of control information of S3 type. In ISO/IEC 16500 the following control protocols are classified as S3:

- Session Control;
- STU Configuration;
- Interface Initialization;
- Switched Video Broadcasting Channel Change.

8.2 Session control

ISO/IEC 16500 will use the DSM-CC User-Network Session Protocol (ISO/IEC 13918-6 clause 4) for session control.

8.2.1 Overview of protocol stacks

ISO/IEC 16500-2 defines several scenarios in which the S3 flow for session control must be supported. The diagrams below summarize the protocol stacks used for these scenarios.

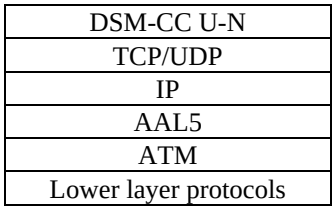


Figure 8.1 — Protocol Stack for Session Control for ATM based transmission

TCP will be used at the Server side, while UDP will be used at the STU side.

IP shall be carried over ATM according to subclause 11.3.1a “IP”.

The Enhanced Broadcast Scenario defines the physical tools for using the PSTN/ISDN/PLMN channel as a return channel with PPP link.

The difference between the PPP link and the normal DAVIC return channel scenarios is that the PPP link provides a connection to an IP network instead of ATM as all other DAVIC scenarios. When the S3 flow shall be carried over the PPP link, the IP address of the SES_n entity shall be well known to the STU. DSM-CC U-N messages shall be sent using normal UDP/IP over the PPP link instead of the dedicated ATM virtual circuit used in other DAVIC scenarios.

In this case the protocol stack is as follows:

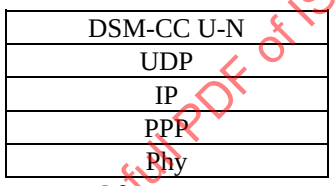


Figure 8.2 — Protocol Stack for Session Control for Internet Access for Enhanced Broadcast Scenario.

It should be noted that in this case the STU needs an IP address for itself in addition to the possible other IP address given to the PC. This IP address will be assigned to the STU using IPCP during the PPP link setup. The IP address for the STU does not need to be a real Internet address; it may be taken from a address space which is defined to be private address space in RFC1597 (i.e., packets from those addresses will not be routed to the Internet, but they may be used for internal communication in the sub-net.)

8.2.2 Description of specific protocols

This section describes the specific protocols used in the protocol stacks identified in the section above. Wherever required the protocols layers are grouped by pairs in order to explicitly focus on the mapping issues between different protocols. Only the protocols which are specific to the S3 flow are described in detail, while the protocols which are common to other flows are described in a separate section.

8.2.2.1 DSM-CC User-Network

The session control protocol for ISO/IEC 16500 is DSM-CC (ISO/IEC 13818-6) User-Network signaling.

8.2.2.2 DSM-CC U-N messages related to session control supported by ISO/IEC 16500

Only some command sequences of the DSM-CC U-N specification for session control are supported in ISO/IEC 16500. These are Client Session Set-Up, Server Add and Release Resource, Server Client and Network Session Tear-Down. In addition the related DSM-CC U-U messages that appear in the command sequences are supported.

Table 8.1 shows the message required for each of the command sequences listed above. The Table also lists the messages required for Session Transfer.

Table 8.1 — DSM-CC U-N messages supported in ISO/IEC 16500

	ISO/IEC 16500 messages
Session set-up	Client Session SetUp Request Server Session SetUp Indication Client Session Proceeding Indication Server Session SetUp Response Client Session SetUp Confirm DSM ServiceGateway attach (Note 1)
Session tear-down	Client Session Release Request Server Session Release Indication Server Session Release Response Client Session Release Confirm Server Session Release Request Client Session Release Indication Client Session Release Response Client Session Release Confirm DSM ServiceGateway detach (Note 1)
Service transfer	DSM ServiceGateway suspend (Note 1) DSM ServiceGateway resume (Note 1)
Add resources	Server Add Resource Request Client Add Resource Indication Client Add Resource Response Server Add Resource Confirm
Release resources	Server Delete Resource Request Client Delete Resource Indication Client Delete Resource Response Server Delete Resource Confirm
Miscellaneous	Server Status Request Server Status Confirm Server Status Indication Server Status Response Client Status Request Client Status Confirm Client Status Indication Client Status Response

Note 1: These messages are DSM-CC U-U messages closely associated with the DSM-CC U-N messages in the command sequence.

ISO/IEC 16500 supports only client initiated session set-up. Session tear-down may be initiated by either client, or server, or network (SRM). Session transfer is always initiated by the server. Addition and release of resources is always initiated by the server.

In ISO/IEC 16500-2 other information flows, related to other procedures, are illustrated such as, e.g., Client initiated add resources; however these procedures are to be considered of informative nature only and are not required to be supported by *ISO/IEC 16500* compliant systems.

All messages are structured according to the DSM-CC U-N specification defined in ISO/IEC 13818-6. All the procedures are according to ISO/IEC 13818-6. The following section shows the use of the message information elements and parameters for *ISO/IEC 16500*.

The following DSM-CC User-to-Network command sequences shall be supported for *ISO/IEC 16500*.

- Client-initiated session set-up
- Server-initiated resource addition
- Server-initiated resource deletion
- Client-initiated session tear-down
- Server-initiated session tear-down
- Network-initiated session tear-down

- Service Transfer

The following tables describe the DSM-CC U-N command sequences and the relevant messages together with notes on the implementation and selected options for DAVIC compliant systems.

8.2.2.3 Client-initiated session set-up

This is the basic command sequence used to set-up a session. It follows the DSM-CC specification, with the following options: ClientSessionProceedingIndication is sent after the session proceeding timer 'Sid.tProceed' expired no Connect messages are used.

Table 8.2 — Client Session Set-Up messages

DSM-CC U-N Session Message	*Note for the usage*
ClientSessionSetup Request	• The session ID shall be assigned by the STU. • The client ID shall be the E.164 NSAP address of the STU. The E.164 NSAP address HO-DST contains the IP address used in setting the TCP connection for the S2 information flow. • UserData shall be limited to 0x400 bytes. • The UserData field shall contain the input parameters of DSM ServiceGateway Attach which are rClientRef, rClientProfile, aEndUser, aSuspendContext and rPathSpec.
ClientSession Proceeding	Sent from the network to the client only when the timer value Sid.tProceed expires before ClientSessionSetupConfirm is sent to the STU.
ServerSessionSetup Indication	
ServerAddResource Request	• No uuData is allowed • The Server shall assign resourceNum to all resources included in the Server Add Resource Request. • The message shall contain resource descriptors for a group of resources providing the S2 connection. If a S1 flow is set-up at session start (e.g., for download purposes), a second set of resource descriptors for the S1 connection will be included in the message. See Note 1 below. • Table 8.9 gives the resource descriptors supported by *ISO/IEC 16500*.
ServerAddResource Confirm	• In the case of PVC, the AtmConnection resource descriptors shall contain the VPI/VCI selected for the requested ATM channel(s). See Note 1 below.
ServerSessionSetup Response	• The uuData field shall contain the output parameters of DSM ServiceGateway Attach which are aResumeContext, rPathRef and rDateTime. Note 1: When all resources are to be allocated by the server (and not by SRM), as is the case of end-to-end ATM SVC then parameters (i.e., resource descriptors) in ServerAddResourceRequest are carried in ServerSessionSetupResponse, and ServerAddResourceRequest/Confirm are not used.
ClientSessionSetup Confirm	• Table 8.9 gives the resource descriptors supported by *ISO/IEC 16500*.

8.2.2.3.1 Server-initiated resource addition

This command sequence is used to add in-session resources

Table 8.3 — Server Add Resource messages

ServerAddResource Request	• In this command sequence, uuData is allowed. • The Server shall assign resourceNum to all resources included in this message. • Table 8.9 gives the resource descriptors supported by *ISO/IEC 16500*.
ClientAddResource Indication	• For the physical instances #3 and #4 of the ATM access network case, the message shall contain the AtmSvcConnection or AtmConnection resource descriptors that indicate VPI and VCI for the added ATM connection(s). • Table 8.9 gives the resource descriptors supported by *ISO/IEC 16500*.
ClientAddResource Response	This basically is a check that the STU can support the resources agreed upon
ServerAddResource Confirm	• For the physical instance #4, the message shall contain the AtmConnection resource descriptors which indicate VPI and VCI for the added ATM connection(s).

8.2.2.3.2 Server-initiated resource deletion

Table 8.4 — Server Delete Resource messages

ServerDeleteResource Request	• The S2 resources are only allowed to be deleted if other S2 resources have been successfully added.
ClientDeleteResource Indication	
ClientDeleteResource Response	
ServerDeleteResource Confirm	

8.2.2.3.3 Server-initiated session tear-down

Table 8.5 — Server Session Tear-Down messages

ServerSessionReleaseRequest	
ClientSessionReleaseIndication	
ClientSessionReleaseResponse	
ServerSessionReleaseConfirm	

8.2.2.3.4 Client-initiated session tear-down

Table 8.6 — Client-Initiated Session Tear-Down messages

ClientSessionReleaseRequest	• The uuData field shall contain the input parameters of DSM ServiceGateway detach.
ServerSessionReleaseIndication	
ServerSessionReleaseResponse	• The uuData field shall contain the output parameters of DSM ServiceGateway detach.
ClientSessionReleaseConfirm	

8.2.2.3.5 Network-initiated session tear-down

Table 8.7 — Network-Initiated Session Tear-Down messages

ClientSessionReleaseIndication	
ServerSessionReleaseIndication	
ClientSessionReleaseResponse	
ServerSessionReleaseResponse	

8.2.2.3.6 Service Transfer

Table 8.8 — Service Transfer

DSM-CC		*Note for the usage*
U-U msg	***U-N msg***	
Scenario I & Scenario II SUSPEND	-	e-e control passed as an S2 information flow SUSPEND "in" parameters = aReason SUSPEND "out" parameters = aResumeContext Ref: DSM-CC subclause 5.5.6.6 DSM ServiceGateway suspend
	Scenario I Client-initiated session tear-down U-N messages	as in Table 8.6 Client initiated session tear-down
	Scenario II Server-initiated resource deletion U-N messages	as in Table 8.4 Server delete resource messages
ATTACH "in"	e-e control carried in the uuData of ClientSessionSetUpRequest and ServerSessionSetUpIndication in Client-initiated session set-up below	contains parameters (= in ServiceContextList, aPrincipal, downloadInfoReq, aSuspendContext, rPathSpec)
ATTACH "out"	e-e control carried in the uuData of ServerSessionSetUpResponse and ClientSessionSetUpConfirm in Client-initiated session set-up below	contains parameters (= out ServiceContextList, downloadInfoResp, aResumeContext, rPathRefs, resolved IORs)
	Client-initiated session set-up U-N messages	as in Table 8.2 Client session set-up
DETACH "in"	e-e control carried in the uuData of ClientSessionRequest and ServerSessionReleaseIndication in Client-initiated session tear-down U-N messages below	uuData of ClientSessionReleaseRequest contains detach "in" parameters (=aSuspend) Ref: DSM-CC subclause 5.5.6.5 DSM ServiceGateway detach
	Client-initiated session tear-down U-N messages	as in Table 8.6 client initiated session tear-down
Scenario II RESUME	-	e-e control passed as an S2 information flow RESUME "in" parameters = aSuspendContext, RESUME "out" no parameters Ref: DSM-CC subclause 5.5.6.7 DSM ServiceGateway resume
	Scenario II Server-initiated resource addition	as in Table 8.3

Scenario I = Server A Session is released
Scenario II = Server A Session resources are minimized
Scenario III = Server A Session kept in full

The Table below illustrates the DSM-CC Resource Types required for *ISO/IEC 16500*. the Resource Types are defined for the two cases of ATM based Access Network (indicated as A) and MPEG TS based Access Network (HFC non-ATM, indicated as B).

Table 8.9 — DSM-CC Resource Types required by *ISO/IEC 16500*

Physical Instance Scenario	*Server-side resources (exist in ServerAddResourceRequest or ServerSessionSetupResponse)*	*Client-side resources (exist in ClientAddResourceIndication or ClientSessionSetupConfirm)*
1	AtmSvcConnection SharedResource SharedRequestId MpegProgram Ip	A: AtmSvcConnection SharedResource SharedRequestId MpegProgram Ip B: Physical Channel TS Downstream MpegProgram ATM Connection SharedResource SharedRequestId Ip
2	same as 1	same as 1A
3	same as 1	same as 1A
4	same as 1 except AtmConnection replaces AtmSvcConnection	A: same as 1A except AtmConnection replaces AtmSvcConnection B: same as 1 B

8.2.3 DSM-CC U-N resource descriptor usage

8.2.3.1 Inclusion of resource descriptors in DSM-CC U-N messages

When a resource (resource A, ex AtmConnection) is used at the underlying layer of another resource (resource B, TsDownStream), resource A is regarded as the ‘parent resource’ of the resource B. The following rule shall be satisfied when resource descriptors are included DSM-CC U-N messages:

Every resource must not be defined in the resource list ahead of its parent resource.

8.2.3.2 Usage of fields

Table 8.10 shows the usage of resource descriptor field. The usage is specified for each resource descriptor field in each message.

Table 8.10 — Resource Descriptor Field Usage

Descriptor Type	Field Name	Usage (Notes 1 and 2)			
		Request	Indication	Response	Confirm
Atm Connection	atmAddress	I(S)	I(S)	Ind	Req
	atmVci	I(S)	I(S)	Ind	Req
	atmVpi	I(S)	I(S)	Ind	Req
AtmSvc Connection	CalledPartyNumber	I(S)	N / I(S) (Note 3)	Ind	Req
	CalledPartySubaddress	I(S)	N / I(S) (Note 3)	Ind	Req
	CallingPartyNumber	I(S)	N / I(S) (Note 3)	Ind	Req
	CallingPartySubaddress	I(S)	N / I(S) (Note 3)	Ind	Req
	TrafficDescriptor	I(S)	N / I(S) (Note 3)	Ind	Req
	ConnectionIdentifier	N	N / I(S) (Note 3)	Ind	N
TsDown Stream	downstreamBandwidth	I(S)	I(S)	Ind	Req
	downstreamTransportId	I(S)	I(S)	Ind	Req
Mpeg Program	mpegProgramNum	I(S)	I(S)	Ind	Req
	mpegPmtPid	I(S)	I(S)	Ind	Req
	mpegCaPid	I(S)	I(S)	Ind	Req
	mpegPidCount	I(S)	I(S)	Ind	Req
	mpegPid	I(S)	I(S)	Ind	Req (Note 4)
	stream_type	I(S)	I(S)	Ind	Req
	associationTag	I(S)	I(S)	Ind	Req
Ip (Note 5)	sourceIPAddress	I(S)	I(S)	Ind	Req
	sourceIPPort	I(S)	I(S)	Ind	Req
	destinationIPAddress	I(S)	Req	Req	Req
	destinationIPPort	N	N	I(S)	N/I(S)
	ipProtocol	I(S)	I(S)	Ind	Req
Physical Channel	channelId	(Note 6)	I(S)	Ind	(Note 6)
	direction	(Note 6)	I(S)	Ind	(Note 6)
Shared Resource	sharedResourceNum	I(S)	I(S)	Ind	Req

Note 1: The columns “in Req”, “in Ind”, “in Resp” and “in Conf” indicate the usage in ServerAddResourceRequest, ClientAddResourceIndication, ClientAddResourceResonse and ServerAddResourceConfirm messages, respectively. When resource descriptors are included in a ServerSessionSetupResponse message, the usage of fields should follow the usage in ServerAddResourceRequest. When resource descriptors are included in a ClientSessionSetupConfirm message, the usage of fields should follow the usage in ClientAddResourceIndication message.

Note 2: The following notations are defined for the usage of fields:

N: The field shall be null in the message

I(S): The field shall be indicated in the single form.

I(R): The field shall be indicated in the range form.

I(L): The field shall be indicated in the list form.

Ind: The field shall be identical to the one in the associated indication message.

Req: The field shall be identical to the one in the associated request message.

Note 3: The field should be indicated in the Physical Instance #3, and should be null in other instances.

Note 4: The implication is that the network should perform PID re-mapping when the server side PIDs and client side PIDs are different.

Note 5: For Ip resource descriptors, the term ‘source’ is used for the server, and the term ‘destination’ is used for the client. See 1.1.3 for additional constraints on the usage of Ip resources.

Note 6: The descriptor is used only in the client side.

8.2.4 DAVIC-defined resource descriptors

8.2.4.1 EnhancedBroadcast descriptor

The EnhancedBroadcast descriptor covers the following modes:

1 = with MPEG TS and PSTN/NISDN/PLMN Downstream and PSTN/NISDN/PLMN Upstream

2 = with MPEG TS Downstream and PSTN/NISDN/PLMN Upstream

3 = PSTN/NISDN/PLMN Downstream and Upstream

8.2.4.2 DhcpOffer descriptor

The DhcpOffer descriptor covers the following parameters (RFC 1541 RFC 1533):

htype	Hardware address type = 1 for 10mb Ethernet
hlen	= 6 for 10mb Ethernet
hops	= 0
xid	Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server
secs	= 0
ciaddr	= 0
yiaddr	IP address offered to client
siaddr	IP address of next bootstrap server
flags	if ‘giaddr’ is not 0 then ‘flags’ from client message else 0
chaddr	identical to 4=1
giaddr	identical to 4
sname	Server host name or options
file	Client boot file name or options
options	Requested IP address - MUST NOT IP address lease time - MUST Use 'file'/'sname' fields - MAY DHCP message type - DHCPOFFER Parameter request list - MUST NOT Message - SHOULD Client identifier - MUST NOT Class identifier - MUST NOT Server identifier - MUST Maximum message size - MUST NOT All others - MAY

8.2.5 Operations to cope with rainy day scenarios

8.2.5.1 Status check procedure

The following usage of DSM-CC U-N Status operation shall be supported for the SRM to detect unexpected terminal down.

- If the network did not receive any DSM-CC U-N message from a terminal (STB or Server) for some pre-determined period of time, the network (SESn) may send a ClientStatusIndication or ServerStatusIndication message to the terminal. The statusType field of the messages shall be set to 0x0001, which indicates request of a list of the active sessions. Upon the receipt of the request message, the terminal shall send a ClientStatusResponse or ServerStatusResponse message to the network.

- If the terminal did not respond to the status request message, the network shall release the sessions associated with the terminal. To release the sessions, the network shall send a ClientSessionReleaseIndication or ServerSessionReleaseIndication message to the counterpart terminal of each session. The reason code to be included in ClientSessionReleaseIndication shall be set to 0x0004 (RsnSeProcError) which indicates that procedure error has been detected at the Server. The reason code to be included in ServerSessionReleaseIndication shall be set to 0x0002 (RsnClProcError).

8.2.5.2 Reset procedure

DSM-CC U-N Reset operation shall be supported for the terminal to request the SRM to release all active sessions in case of terminal failure.

8.3 STU configuration

ISO/IEC 16500 will use the DSM-CC U-N Configuration protocol (ISO/IEC 13818-6 clause 3) for the configuration of the STU. The STU configuration operation allows the STU to get necessary information to start session set-up (e.g., IP address). In the following, the STU configuration server is defined as the network-side peer entity of the STU configuration.

8.3.1 Overview of protocol stacks

DSM-CC U-N configuration
UDP
IP
LLC/SNAP
AAL5
ATM
Physical Layer

Figure 8.3 — Protocol stack for STU configuration

8.3.2 Description of specific protocols

Step 1: The STB sends a DSM-CC U-N configuration request message with a UDP/IP datagram. If the STB knows the IP address of the STU configuration server, the STB shall fill this address in the IP header field of the message as the destination address. Otherwise the STB shall set the destination address to 255.255.255.255 (all 1). The destination port number shall be the well-known value assigned to DSM-CC U-N. If the STB knows its own IP address for STU configuration, the STB shall set this address in the IP header field of the message as the source address. Otherwise the source IP address shall be set to 0.0.0.0 (all 0).

Step 2: Upon the reception of a DSM-CC U-N configuration request, the STU configuration server formulates a DSM-CC U-N configuration confirm message and sends it to the STB with a UDP/IP datagram. If the source IP address of the configuration request message is 0.0.0.0, the destination IP address of the configuration confirm message shall be set to 255.255.255.255. Otherwise, the destination IP address of the configuration confirm message shall be the source IP address of the configuration request message.

8.3.2.1 Usage of DSM-CC U-N configuration messages

The usage of networkParameter field in networkConfigurationParameters() of DSM-CC U-N configuration confirm is specified in the below table.

parameters	# of octets
networkNumberingPlanIndicator	1
sessionControlChannelFlag	1
srmIpAddress	4
stbIpAddress	4

Parameters:

networkNumberingPlanIndicator: as specified in ISO/IEC 16500-2 clause 10 (Note: this parameter is only useful when the VC for STU configuration is a PVC).

sessionControlChannelFlag: This parameter indicates whether the STB needs to establish a separate ATM VC for DSM-CC session control. If the value is 0x00, the STB may use the same ATM VC for STB configuration and DSM-CC session control. If the value is 0x01, the STB shall obtain necessary information through DIIP and avail a separate VC for DSM-CC session control.

srmIpAddress: This parameter indicates the SRM's IP address.

stbIpAddress: This parameter indicates the STB's IP address for DSM-CC session control.

8.4 Interface initialization (DIIP)

This section specifies the method of assigning to the STB the initial parameters that allow connections (channels) from the STB to different network entities (e.g., SESn, BCU-Broadcast Control Unit) to be used. The protocol specified in this section is called DIIP (DAVIC Interface Initialization Protocol 1.0).

The specification is based on the following principles:

1. Each protocol message is transported using one ATM-cell.
2. The protocol runs on the 'one transaction for one channel basis, which means that if more than one channel is needed, then multiple transactions are required.
3. The protocol is defined for a point to point configuration. The usage for point to multi-point configurations is for further study.
4. The protocol is defined to be used independent of the physical architecture on which it is transported. This implies that the user side termination point of the protocol resides inside the STU.
5. The protocol operates over a specific predefined VPI/VCI value.
6. The protocol supports both provisioned VC (PVC) allocated by the network or switched VC (SVC) to be established. In the case of PVC, the VPI/VCI of the channel is notified to the STB. In the case of SVC, the DIIP protocol returns the network address of the entity to which a connection shall established.

8.4.1 Overview of protocol stacks

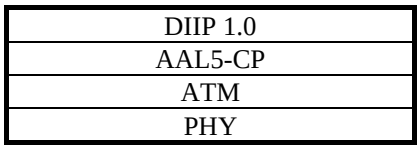


Figure 8.4 — Protocol stack for interface initialization

The messages are carried with a fixed VPI,VCI value. Another value than VPI,VCI=0,1 should be used to allow co-existence with meta-signaling as defined in ITU-T Q.2120.

8.4.2 Description of specific protocols

The DIIP 1.0 utilizes the following three messages:

- ASSIGN_REQUEST
- ASSIGNED
- DENIED

according to the following procedure:

1. The STB sends a ASSIGN_REQUEST message.
2. If the initialization can be successfully completed, the network returns a ASSIGNED message. Otherwise, the network returns a DENIED message.

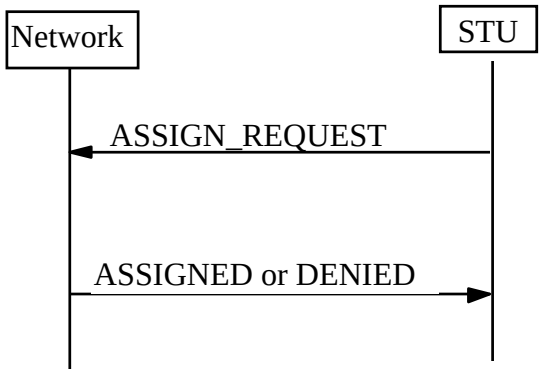


Figure 8.5 — DIIP message sequence

8.4.2.1 Message format

8.4.2.1.1 DIIP general message syntax

Syntax	Num. of Bytes
DiipMessage(){	
DiipHeader()	9 (composite)
DiipBody()	variable (composite)
}	

8.4.2.1.2 DiipHeader() definition

Syntax	Number of Bytes
DiipHeader(){	
protocol_discriminator	1
protocol _version	1
message type	1
device ID	6
}	

The ***protocol_discriminator*** shall be set to 0x02.

The ***protocol_version*** shall be set to 0x01.

The ***message type*** shall be coded according to Table 8.13.

The ***device_ID*** shall be a unique terminal identification number which shall be a MAC-ID.

8.4.2.1.3 DiipBody() definition

DiipBody for ASSIGN_REQUEST

Syntax	Number of Bytes
DiipBody_AssignRequest(){	
channel_type	1
device_type	1
}	

The ***channel_type*** shall indicate the type of the requested channel (see Table 8.15).

The ***device_type*** indicates the capability of the STB, e.g., Q.2931 capable or non capable (see Table 8.14).

DiipBody for ASSIGNED

Syntax	Number of Bytes
DiipBody_Assigned(){	
channel_type	1
VcTypeIndicator	1
if(VcTypeIndicator == PVC)	(composite)
PvcDescriptor()	(composite)
else if(VcTypeIndicator == SVC)	
SvcDescriptor()	
}	

The ***VcTypeIndicator*** shall indicate whether the requested channel is a provisioned VC (PVC) allocated by the network or a switched VC (SVC) to be established. The values are defined in Table 8.17.

The ***channel_type*** shall indicate the type of the assigned channel (see Table 8.15).

PvcDescriptor

Syntax	Number of Bytes
PvcDescriptor(){	
VPI	2
VCI	2
Peak Cell Rate	3
}	

The ***VPI*** and ***VCI*** shall indicate the VPI and VCI of the provisioned channel.

The ***Peak Cell Rate*** shall indicate the peak cell rate of the provisioned channel.

SvcDescriptor

Syntax	Number of Bytes
SvcDescriptor(){	
Network Address	20
}	

The ***Network Address*** shall indicate the network address of the network entity to which a connection shall be established. The address shall be encoded based on the E.164 NSAP address format and it must be compliant with both the cases of E.164 based networks and of E.164 NSAP based networks.

DiipBody for DENIED

Syntax	Number of Bytes
DiipBody_Denied(){	
channel_type	1
Cause	1
}	

The ***channel_type*** shall indicate the type of the denied channel.

The ***Cause*** shall indicate the reason for denial of the request. The values are specified in Table 8.16.

8.4.2.2 Information element format

Table 8.11 — Protocol discriminator.

Value (hex)	**Usage**	**Comments**
0x00	*Not allocated*	
0x01	reserved	
0x02	DIIP	
0x03-0xFF	reserved	

Table 8.12 — Protocol version

Value (hex)	**Usage**	**Comments**
0x00	*Not allocated*	
0x01	Version number 1.0	
0x02-0xFF	reserved	

Table 8.13 — Message type

Value (hex)	**Usage**	**Comments**
0x00	*Not allocated*	
0x01	Assign request	
0x02	Assigned	
0x03	Denied	
0x04-0xFF	*reserved*	

Table 8.14 — Device type

Value (hex)	**Usage**	**Comments**
0x00	*Not allocated*	
0x01	Q.2931 non capable	
0x02	Q.2931 capable	
0x04-0xFF	*reserved*	

Note: Other values for device_type are for further study.

Table 8.15 — Channel Types.

Value (hex)	**Usage**	**Comments**
0x00	*reserved*	
0x01	Session Control	DSM-CC U-N session
0x02	STU Configuration	DSM-CC U-N Configuration
0x03	Switched Broadcast Control	DSM-CC SDB-CCP
0x04-0xEF	*Reserved*	
0xF0-0xFF	*Network Implementation Specific Connections*	*Proprietary Interface*

Peak Cell rate

Peek Cell Rate is an integer value of 3 bytes which indicates the cell rate in number of cells per seconds.

Table 8.16 — Cause

Value (hex)	**Usage**	**Comments**
0x00	not applicable	
0x01	No resource provisioned	Used in case a PVC, network address, or Peak Cell Rate values have not been provisioned
0x02	DIIP version not supported	
0x03	device ID invalid	
0x04	device type invalid	
0x05-0xFF	reserved	

Table 8.17 — VcTypeIndicator

Value	Name	Semantics
0x00	reserved	-
0x01	PVC	The channel to the network entity is a provisioned VC allocated by the network.
0x02	SVC	The channel to the network entity is a switched VC to be established by signaling procedures.
other values	reserved	

8.4.3 DIIP transaction state machine

The state machine should be instantiated for each transaction. Transaction is identified by the device_ID and the channel_type.

8.4.3.1 STB state machine

1. The following constant parameters are defined:
   - Tmax.Retrans: the maximum waiting time before message retransmission
   - Nmax.Retry: the maximum number of retransmission before the conclusion of transaction
2. The following variables are defined:
   - t.Retrans: the retransmission timer
   - n.Retry: the retransmission counter

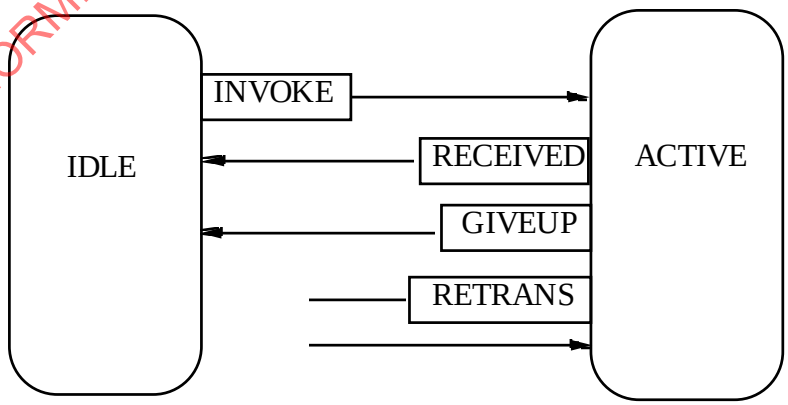


Figure 8.6 — STB state machine model

Table 8.18 — Transition table for the STB state machine

Current State	Event			
	Name	Condition	Action	Next State
IDLE	INVOKED	The application has invoked DIIP.	1. n.Retrans := 0 2. Send the ASSIGN message 3. t.Retrans := 0	ACTIVE
ACTIVE	RETRANS	t.Retrans > Tmax.Retrans	1. n.Retrans++ 2. Resend the message 3. t.Retrans := 0	ACTIVE
	GIVE UP	n.Retrans > Nmax.Retrans	1. Inform the application of the transaction failure	IDLE
	RECEIVED	The STB has received response.	1. Process the response.	IDLE

The recommended value of Tmax should be between 1000 and 3000 ms.

The recommended value of Nmax should be equal 2.

8.5 Network state machine

No constant or variable is defined for the network state machine.

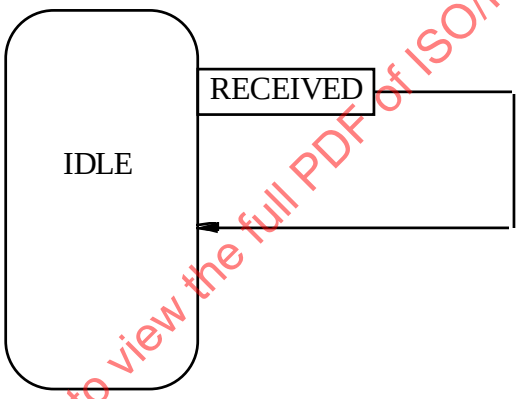


Figure 8.7 — Network state machine model

Table 8.19 — Transition table for the Network state machine

Current State	Event			
	Name	Condition	Action	Next State
IDLE	RECEIVED	The application has received message.	Process message and respond.	IDLE

8.6 Switched video broadcasting channel change

The Switched Video Broadcast Service offers end users a large number of broadcast TV programs over access networks with a relatively low capacity per end user.

For example, in the case of an ADSL access network, capacity constraints may limit the number of simultaneously received TV programs to one or two. However, the end user may want to select these TV programs out of a much larger set of programs. Based on the DSM-CC SDB-CCP (switched digital broadcast - channel change protocol), the CCP protocol allows users to select a program from a remote unit, so that only the desired program(s) is (are) passed across the access network, overcoming the reduced channel capacity across the A1 interface.

8.6.1 Overview of protocol stacks

The DSM-CC SDB-CCP protocol has been adapted to the DAVIC environment. The protocol has been engineered to be transported on top of AAL5 with a null SSCP.

DSM-CC SDB-CCP
AAL5
ATM
Physical Layer

Figure 8.8 — Protocol stack for the CCP function

8.6.2 CCP message sequence

The CCP has been designed in such a way that messages will fit into the payload of one ATM cell. The basic CCP consists of two messages: the request to change channel (ProgramSelectRequest) and the response to the request (ProgramSelectConfirm).

The message sequence is shown in Figure 8.9.

1. The Broadcast Control Unit in the STU sends a ProgramSelectRequest message to request the Broadcast Control Unit at the network side to switch a selected program to the STU. If the STU knows the VPI/VCI values associated with the selected program, it may tune to that VPI/VCI before receiving the response from the network.
2. The network replies to the ProgramSelectRequest message sending a ProgramSelectConfirm message to the STU.

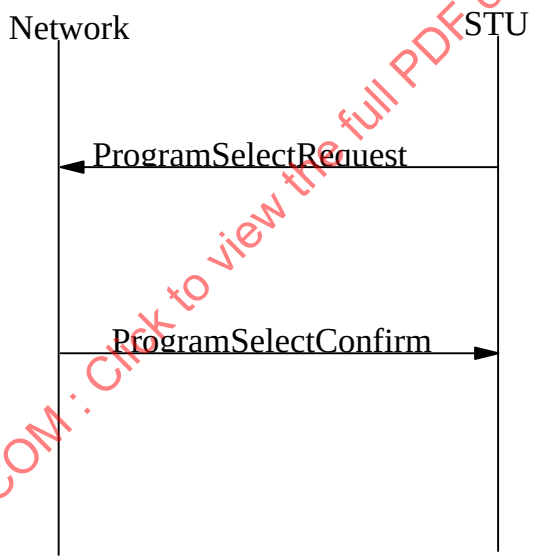


Figure 8.9 — CCP message sequence

8.6.3 CCP message structure

This section specifies the usage of the DSM-CC SDB-CCP messages for the DAVIC SVB service.

8.6.3.1 Message header

8.6.3.1.1 Syntax

Parameter	Octet
Protocol Discriminator	1
DSM-CC Type	1
Message ID	2
Transaction ID	4
Reserved	1
Adaptation Length	1
Message Length	2

8.6.3.1.2 Parameters

Protocol Discriminator

Protocol Discriminator identifies the protocol and its version.

Value	Protocol
0x11	DSM-CC
other values	reserved

DSM-CC Type

This field identifies the specific DSM-CC functional group.

Value	DSM-CC Type
0x04	DSM-CC SDB CCP
other values	reserved by DSM-CC

Transaction ID

Transaction ID is an incremental counter used to associate a response message with a request message. Each time the STU sends a request (ProgramSelectRequest) message, the STU should assign a new Transaction ID to the message. The network shall set the Transaction ID in a response (ProgramSelectConfirm) message to be identical to the corresponding request message.

Message ID

Message ID identifies the type of the message.

Value	Message Type
0x0000	reserved
0x0001	ProgramSelectRequest
0x0002	ProgramSelectConfirm
other values	reserved

Adaptation Length

Message Length

Message length indicates the number of valid bytes that follows this parameter.

8.6.3.2 Message body syntax

8.6.3.2.1 ProgramSelectRequest message

Parameter	Octet
Session ID	10
Broadcast Program ID	4
Private Data	

8.6.3.2.2 ProgramSelectConfirm message

Parameter	Octet
Session ID	10
Broadcast Program ID	4
Response	2
Private Data	

8.6.4 The usage of ProgramSelect message parameters

8.6.4.1 Session ID

The SessionId shall be assigned by the STU.

It is not required by the ISO/IEC 16500 for the STU to establish a DSM-CC session to initiate DAVIC SVB services. For these scenarios, the STU shall have a list of sessionIds dedicated for the SVB service and, each time the STU starts the SVB service, the STU shall select an unused value from the list of the dedicated sessionIds. The assignment of the dedicated sessionIds to the STU is outside the scope of the ISO/IEC 16500.

8.6.4.2 Broadcast Program ID

Broadcast Program ID identifies the video program that the STU wants to switch to. The value ‘0x00000000’ (‘NoProgram’) shall be used for the termination of the subscribed SVB service. When the user want to stop viewing, the user shall send a ProgramSelectRequest message with the BroadcastProgramId indicating ‘NoProgram’.

8.6.4.3 Response codes

The usage of response code shall follow Table 8.20.

Table 8.20 — Response Codes in ISO/IEC 16500

rspOk	0x0000	Indicates a positive acknowledgment.
rspFormatError	0x0001	Indicates that the condition is due to invalid format (e.g., missing parameter) detected.
rspNoSession	0x0002	Indicates the reception of an invalid sessionId.
rspNoNetworkResource	0x0003	
rspNoServerResource	0x0004	
rspEntitlementFailure	0x0005	Indicates that a Broadcast Program could not be delivered to a Client due to entitlement failure.
rspBcProgramOutOfService	0x0006	Indicates that a Broadcast Program cannot be delivered because it is out of service.
rspRedirect	0x0007	Indicates that the requested program is not available but user has been redirected to the view of a different program.
reserved	0x0008 - 0x7fff	ISO/IEC 13818-6 reserved
private use	0x8000 - 0xffff	For private use

8.6.4.4 PrivateData() field

The value of PrivateDataCount shall be less than or equal to 10. The content of the PrivateData() field for each ProgramSelect message is specified below.

8.6.4.4.1 PrivateData() in ProgramSelectRequest

Not used.

8.6.4.4.2 PrivateData() in ProgramSelectConfirm

In the case of ATM baseband access networks, the value of privateDataCount is 0x0005.

Table 8.21 — PrivateData() in SdbPRogramSelectRequest

Syntax	Number of Bytes

PrivateData(){
privateDataCount
DavicSvbNetworkResourceDescriptor()
}
	
	2
	composite (see 8.6.5)

The definition of DavicSvbNetworkResourceDescriptor is given below.

8.6.5 DavicSvbNetworkResourceDescriptor definition

8.6.5.1 Syntax

Table 8.22 — DavicSvbNetworkResourceDescriptor syntax

Syntax	Number of Bytes

DavicSvbNetworkResourceDescriptor(){
SvbNetworkResourceDescriptorType
Value()
}
	
	1
	composite (see 8.6.5.2)

8.6.5.1.1 SvbNetworkResourceDescriptorType

Table 8.23 — SvbNetworkResourceDesciptorType definition

Value	Name
0x01	AtmVc
other values	reserved

8.6.5.2 Value()

8.6.5.2.1 AtmVc value

Table 8.24 — AtmVC value syntax

Syntax	Number of Bytes
atmVpi atmVci	2 2

The parameters atmVpi and atmVci shall indicate the VPI and VCI of the ATM VC to which the user should tune.

8.7 Network-initiated channel changing

Optionally a CCP procedure initiated at the network side may be supported by some service providers. The procedure may be used to support, e.g., automatic program change at start of event (e.g., start of selected pay-per-view program).

This procedure is realized by the use of two additional messages in the SVB Channel Change protocol.

The two messages provide, respectively:

- the indication from the network that tuning should now change to the indicated program, and
- the confirmation from the STU that the message has been received.

The message sequence is shown in Figure 8.10:

1. The Broadcast Control Unit at the network side sends ProgramSelectIndication message to notify the STU to switch to the video program indicated in the message.
2. The STU replies to the ProgramSelectIndication message sending a ProgramSelectResponse message to the network.

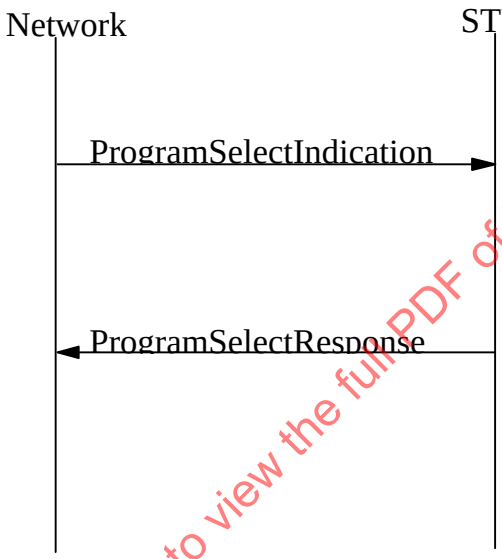


Figure 8.10 — Network Initiated CCP sequence

8.7.1 Message header

See subclause 8.6.3.1.

8.7.2 Message body Syntax

8.7.2.1 ProgramSelectIndication message

Parameter	Octet
Session ID	10
Reason	2
Broadcast Program ID	4
Private Data	

8.7.2.2 ProgramSelectResponse message

Parameter	Octet
Session ID	10
Response	2
Private Data	

8.7.3 Parameters

Parameters are defined as in subclause 8.6.4. Additionally, the parameter “reason” is used in ProgramSelectIndication as defined below.

8.7.3.1 Reason codes

The usage of reason code shall follow Table 8-25.

Table 8.25 — Reason Codes in ISO/IEC 16500

rspOk	0x0000	Indicates that a Broadcast Program has been started as a normal action (e.g., pre-subscribed Pay Per View event starts).
rsnNormal	0x0001	Indicates that a Broadcast Program has been discontinued due to Session Release.
rsnSeEntitlementFailure	0x0002	Indicates that a Broadcast Program has been discontinued due to entitlement failure.
reserved	0x0003 - 0x7FFF	ISO/IEC 13818-6 reserved
private use	0x8000 - 0xFFFF	For private use

9. S4 flow: high and mid layer protocols

9.1 S4 flow description

The S4 flow is a bi-directional flow supporting the call/connection control and the resource control functions. The call connection control functions consist of the capabilities to establish call and connection in a B-ISDN network.

The S4 flow is not transparent to the Delivery System at any layer.

9.2 Overview of protocol stacks

The following figures show, respectively the entire protocol stack used for call/connection control.

The standard B-ISDN call/connection control protocols are used in ISO/IEC 16500. In particular ITU-T Q.2931, Q.2130, and Q.2110 are specified.

Specific requirements apply to the A0 interface which makes use of a dedicated protocol for the control of the connection at the A0 interface, between the STU and the NIU. The protocols applied for this function is given in Section 12.

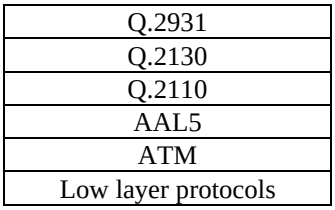


Figure 9.1 — B-ISDN Call/Connection Control Protocol Stack

A description of the end-to-end scenarios for this type of DAVIC services, showing all the flows involved with the related protocol stacks, is provided in ISO/IEC 16500-2.

For setting up the low-speed interaction channel in the case of enhanced broadcast services, existing technology as shown in protocol stacks drawn in Figure 9.2, Figure 9.3, and Figure 9.4 shall be used.

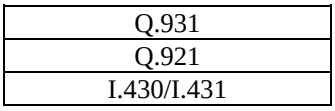


Figure 9.2 — S4 protocol stack for ISDN

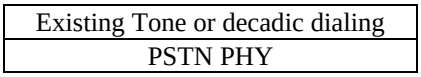


Figure 9.3 — S4 Protocol stack for PSTN

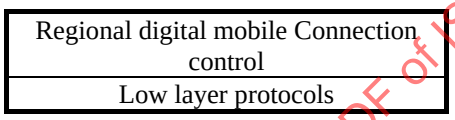


Figure 9.4 — S4 Protocol Stack for PLMN

9.3 Description of specific protocols

This section describe the specific protocols used in the protocol stacks identified in the section above for the case of B-ISDN. Wherever required the protocols layers are grouped by pairs in order to explicitly focus on the mapping issues between different protocols. Only the protocols which are specific to the S4 flow are described in detail, while the protocols which are common to other flows are described in a separate section.

9.3.1 Call/connection control protocol

The call/connection control protocol adopted is specified in ITU-T Rec. Q.2931. This protocol supports the establishment and release of calls consisting of a single point-to-point connection. This is considered sufficient for the scope of ISO/IEC 16500, where the co-ordination of more connections with different end points is performed by the session control protocol, i.e., DSM-CC U-N.

For future releases of DAVIC specs, the evolution of the B-ISDN network signaling protocols has to be taken into account. This will allow the support of advanced multimedia services within the network capabilities.

For the purpose of ISO/IEC 16500 the following clauses specify the selected message set and related information element required to implement a working sub-set of Q.2931.

9.3.1.1 Q.2931 message to be supported by ISO/IEC 16500

Table 9.1 shows the subset of messages to be supported in ISO/IEC 16500 are indicated. The call establishment and the call clearing messages are those implicitly included in the DAVIC macros of ISO/IEC 16500-2.

Table 9.1 — Q.2931 messages supported in ISO/IEC 16500

	ISO/IEC 16500 messages
Call establishment messages	CALL PROCEEDING CONNECT CONNECT ACKNOWLEDGE SET-UP
Call clearing messages	RELEASE RELEASE COMPLETE
Miscellaneous messages	STATUS STATUS ENQUIRY
Messages with Global Call Reference	RESTART RESTART ACKNOWLEDGE

The use of call establishment and call clearing messages is shown in the following section. The use of the miscellaneous messages and of the messages with global Call Reference is not shown, however they are required in order to allow recovery from errors that may occur during the connection..

9.3.1.2 Minimum call set up and release message sequences

Figure 9.5 and Figure 9.6 show the minimum required sequence of messages for the Call Set-Up and for the Release phases respectively

The figure also shows which messages are local to the UNI.

Call release may be initiated by any of the two parties (i.e., call release may be initiated by the called party). In the scenarios described in ISO/IEC 16500-2 it is always the calling party that releases the call.

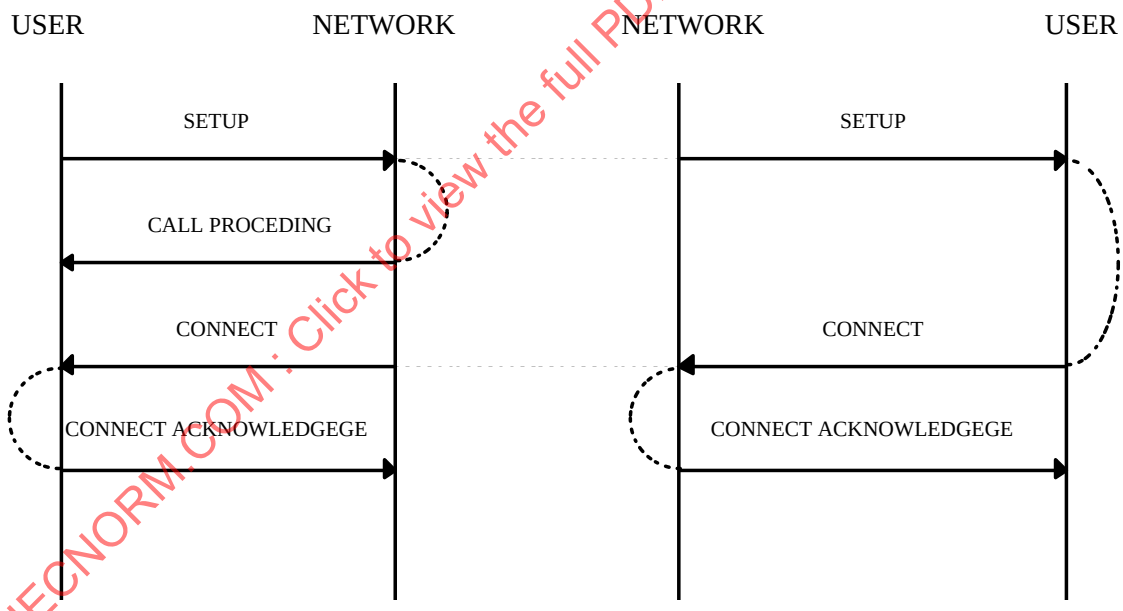


Figure 9.5 — Minimum call set-up message sequence

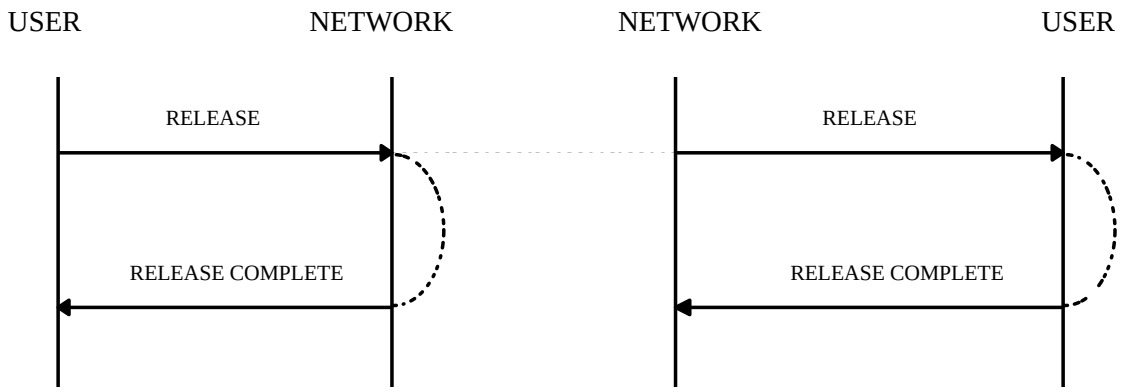


Figure 9.6 — Call clearing message sequence

9.3.1.3 Call/connection states supported at the User side of the UNI

Table 9.2 shows the states which need to be supported at the User side of the UNI. This is important for the definition of the state machine of the STU.

Table 9.2 — Call/connection states supported at the User side of the UNI

STATE		
Null	(U0)	yes
Call Initiated	(U1)	yes
Outgoing Call Proceeding	(U3)	yes
Call Delivered	(U4)	no
Call Present	(U6)	yes
Call Received	(U7)	no
Connect Request	(U8)	yes
Incoming Call Proceeding	(U9)	no
Active	(U10)	yes
Release Request	(U11)	yes
Release Indication	(U12)	yes

9.3.1.4 Information Elements for ISO/IEC 16500

The set of tables below contain in the rows the various Information Elements, and in the columns the messages required for the SET-UP, and RELEASE macros, in particular, call proceeding, connect, connect acknowledge, set-up, release, release complete. For each message, each information element has been marked with a specific character:

D	mandatory for ISO/IEC 16500 (i.e., always present in the message)
x	optional for ISO/IEC 16500
-	not supported in ISO/IEC 16500

Information Elements marked by “D” include all IEs which are mandatory in Q.2931 plus some IEs which are considered essential for ISO/IEC 16500.

Information Elements marked by “x” include all IEs which are optional in Q.2931 and are supported by ISO/IEC 16500.

Information Elements marked by “-” include all IEs which are optional in Q.2931 and are not supported by ISO/IEC 16500.

Table 9.3 shows the information elements of the common part of the messages only. The IEs of the common part are always mandatory for all messages in both directions, i.e., User-to-Network and Network-to-User.

For the other tables (Table 9.4 and Table 9.5) a differentiation has been made depending on the direction of the message.

Table 9.3 — DAVIC supported Information Elements for the message common part

	call proceeding	connect	connect ack	set-up	release	release complete
protocol discriminator	D	D	D	D	D	D
call reference	D	D	D	D	D	D
message type	D	D	D	D	D	D
message Length	D	D	D	D	D	D

Table 9.4 — DAVIC supported Information Elements for messages in the User-to-Network direction

	call proceeding	connect	connect ack	set-up	release	release complete
AAL parameters		--		D		
ATM user traffic descriptors				D		
broadband bearer capability				D		
broadband high layer information				x		
broadband repeat indication				--		
broadband low layer information		--		x		
cause					D	x
called party number				D		
called party sub address				x		
calling party number				--		
calling party sub address				--		
connection identifier	x	x		x		
end-to-end transit delay		--		--		
generic identifier transport (*)				(*)		
notification indicator	--	--	--	--	--	
OAM traffic descriptor		--		--		
QoS parameter				D		
broadband sending complete				D		
transit network selection				--		

(*) Generic identifier transport IE. Defined by ITU-T, but not included yet at present in Q.2931. Included as optional in ISO/IEC 16500 SET-UP message to indicate the session identifier.

Table 9.5 — DAVIC supported Information Elements for messages in the Network-to-User direction

	call proceeding	connect	connect ack	set-up	release	release complete
AAL parameters		--		D		
ATM user traffic descriptors				D		
broadband bearer capability				D		
broadband high layer information				x		
broadband repeat indication				--		
broadband low layer information		--		x		
cause					D	x
called party number				x		
called party sub address				x		
calling party number				--		
calling party sub address				--		
connection identifier	x (note 1)	x (note 1)		D		
end-to-end transit delay		--		--		
generic identifier transport (*)				(*)		
notification indicator	--	--	--	--	--	
OAM traffic descriptor		--		--		
QoS parameter				D		
broadband sending complete				x (note 2)		
transit network selection				--		

(note 1) Mandatory in the first response to SET-UP message, if not included in the User-to-Network SET-UP message.

(note 2) Mandatory when the called address is present.

(*) Generic identifier transport IE. Not included at the moment in Q.2931. Included as optional in ISO/IEC 16500 SET-UP message to indicate the session identifier.

For all the other messages to be supported, not included here (STATUS, STATUS ENQUIRY, RESTART, RESTART ACKNOWLEDGE) the required Information Elements and the coding is assumed according to ITU-T Rec. Q.2931.

The procedures are as specified in ITU-T Rec. Q.2931.

9.3.1.5 Coding of the Information Elements

In the following sub-subclauses, specific coding of IEs required for ISO/IEC 16500 is indicated. The IEs not explicitly indicated follow the general coding rules of ITU-T Rec. Q.2931.

9.3.1.5.1 AAL parameters IE

Bytes	FIELD	Value	Comments
1	AAL parameters IE ID	01011000	
1	coding standard, IE instruction		
2	Length of AAL parameter contents		
1	AAL-Type	00000101	AAL5
1	Forward max CPCS-SDU size ID	10001100	
2	Forward max CPCS-SDU size		the value has to be specified for each of the flows, for the S1 flow this value will be 2*188 bytes
1	Backward max CPCS-SDU size ID	10000001	
2	Backward max CPCS-SDU size		the value has to be specified for each of the flows, for the S1 flow this value is “0”
1	SSCS-Type ID	10000100	
1	SSCS-Type		the value depends on the flow, for the S1 flow this value is “Null”

9.3.1.5.2 Called party number IE

Bytes	FIELD	Value	Comments
1	called party number IE ID		
1	coding standard, IE instruction		
2	length of called party number content		
1	type of number (3 bits), addressing/numbering plan identification (4 bits)	1xxx0001 or 1xxx0000 or 10000010	xxx depends on the type of number and it is coded according to Q.2931. Allowed values are E.164 native, Unknown, ISO NSAP (Note)
	address/number digit		E.164 number(8 octets) or E.164 NSAP address (20 octets)

Note: The type of address (identification of numbering plan used) to be coded in the Called Party Number (i.e., E.164 native or E.164 NSAP) will be specified and indicated during the configuration phase. If E.164 is to be used in native form in the Called Party Address field, then the E.164 NASP may be coded in the Called Party Sub-address field.

9.3.1.5.3 Connection identifier IE

Bytes	FIELD	Value	Comments
1	Connection Identifier IE ID	10011010	
1	coding standard, IE instruction		
2	Length of connection identifier content		
1	VP-ass. Signaling (2 bits), Pref/Excl (3 bits)	10001xxx	explicit indication of VPCI (non-associated signaling), xxx depend on whether only VPCI is exclusive or both VPCI and VCI are exclusive
2	VPCI		values 0 ÷ 65535
2	VCI		values 32 ÷ 65535

9.3.1.5.4 Generic identifier transport IE

Bytes	FIELD	Value	Comments
1	Generic identifier transport IE ID	01111111	
1	coding standard, IE instruction		
2	length of generic identifier transport content		
1	identifier related standard	00000001	DSM-CC resource ID (ISO/IEC 13818-6)
1	identifier type	00000001	DSM-CC session
1	identifier length		
10	identifier value		DSM-CC session ID value
1	identifier type	00000010	DSM-CC resource number
1	identifier length		
2	identifier value		DSM-CC resource number value

The ITU-T has specified GIT (Generic Identifier Transport) information element. This shall be used in ISO/IEC 16500 in accordance with ISO/IEC DSM-CC to carry both the session ID and resource number/association Tag with the identifier related standard/application field indicating the DSM-CC protocol.

9.3.1.5.5 Other Information Elements

Other information elements required for ISO/IEC 16500 are coded according to ITU-T Rec. Q.2931. These IEs are:

- broadband bearer capability
- broadband high layer information
- broadband low layer information
- called party sub address
- ATM traffic descriptor
- broadband sending complete
- QoS parameter.

9.3.2 S-AAL protocol

The AAL for the support of the S4 flow is specified in Q.2110 (SSCOP) and Q.2130 (SSCF-UNI), which are used on top of AAL5.

10. S5 flow: high and mid layer protocols

10.1 S5 flow description

S5 flow is related to the network management.

Network Management is made up of those functions that are required to maintain and manage the network resources. Among other things, Network management includes configuration, fault and performance, usage data, traffic, and security management.

10.2 Overview of protocols stacks

CMIP defined by the ITU-T X.711 and SNMP defined by RFC 1157shall be used as network management protocols. Although CMIP is more suited for management of public networks which consist of the core and access networks, SNMP may also be used to manage these networks. DAVIC has specified several types of access networks and the choice of a network management protocol for these access networks is difficult since most of these access networks: a) have no standard management protocols associated with them, and b) contain simple network elements such as, for example in HFC access networks, amplifiers, transmitters and power supplies for which a standards based approach is seen as too demanding. For access networks, for the large and complex elements which can support a standards based approach, CMIP or SNMP is recommended, whereas for the simple network elements in which a standards based approach would mean a large overhead, a proxy-based management approach is recommended. For the Set Top Units, end-service provider equipment, and private delivery networks, SNMP is recommended.

According to the architecture and scenarios defined in Section 10 of DAVIC 1.3.1a Part 4, the following set of management interfaces and associated network management protocols have been identified:

Management Interface	Network Management Protocol
NMS - Core Network	CMIP or SNMP
NMS - Access Network	CMIP or SNMP or Proxy based
NMS - Server	SNMP
NMS - STU	SNMP
NMS - NMS	CMIP or SNMP

10.2.1 Upper layer protocol stacks for management interface

The upper layer of the protocol stack includes the application, presentation and session layers. The following figures identify the upper layer protocol stacks for CMIP and SNMP network management protocols.

Figure 10.1 and Figure 10.2 provide the upper layer stacks for CMIP and SNMP management interface respectively.

7	X.710, X.711 (CMISE) X.217, X.218 (ACSE) X.219, X.229 (ROSE)
6	X.216, X.226, X.209 (Presentation, BER)
5	X.215, X.225 (Session)

Figure 10.1 — CMIP Interface Upper Layer Stack

7	RFC 1157 (SNMP)
6	X.216, X.226, X.209 (Presentation, BER)
5	Null

Figure 10.2 — SNMP Interface Upper Layer Stack

10.2.2 Lower layer protocol for management interface

The lower layer protocols for management interfaces depend on the fact whether the network elements (NEs) are ATM capable or not. The lower layer protocols are specified on the basis of ATM capable NE or non-ATM capable NE below:

10.2.2.1 ATM capable NE

For ATM capable NEs, AAL5/ATM is to be used to transport management information flow between NMS and the managed NE. ATM virtual circuits (PVCs or SVCs) are to be used to support management channels.

For CMIP based management of ATM capable NE, two choices of transport and network layers exist as outlined below:

4	ISO/IEC 8073 (TP4)
3	X.25, ISO 8473 (CLNP)
2	AAL5/ATM
1	Physical Medium

Figure 10.3 — CMIP Lower layer stack using TP4 and CLNP

4	ISO/IEC 8073 (TP4)
3	X.25, ISO 8473 (CLNP)
4	RFC 1006 (TP0), RFC 793 (TCP)
3	RFC 791 (IP)
2	RFC 1577/AAL5/ATM
1	Physical Medium

Figure 10.4 — CMIP Lower layer stack using TP0, TCP and IP

For SNMP based management of ATM capable NE the lower layers are outlined below:

4	RFC 793 (TCP) or RFC 768 (UDP)
3	RFC 791 (IP)
2	RFC 1577/AAL5/ATM
1	Physical Medium

Figure 10.5 — SNMP Lower layer stack IP

In the case of PSTN, ISDN or PLMN physical layers, layer 2 of Figure 10.3, Figure 10.4, and Figure 10.5 shall be replaced by PPP(MP) (RFC1717).

10.2.2.2 Non-ATM capable NE

The Network Management Systems that are not ATM-capable require different lower layer protocols, such as IEEE 802.3, X.25/LAPB/RS232, LAPD/SDH DCC, etc.

For management interfaces (supporting CMIP or SNMP) using Local Area Network as lower layers, the following lower layers are used:

2	IEEE 802.2, IEEE 802.3
1	Ethernet

Figure 10.6 — Lower layer stack using Local Area Network

For management interfaces (supporting CMIP or SNMP) using X.25 network as lower layers, the following lower layers are used:

2	ISO 7776 (LAPB)
1	RS232/V.35

Figure 10.7 — Lower layer stack using X.25 network

10.3 Management information flow

For NMS that use ATM based management interface, end-to-end ATM virtual channel connection are used from the NMS to the managed network element including the intermediate NEs. The routing of management information is shown as below:

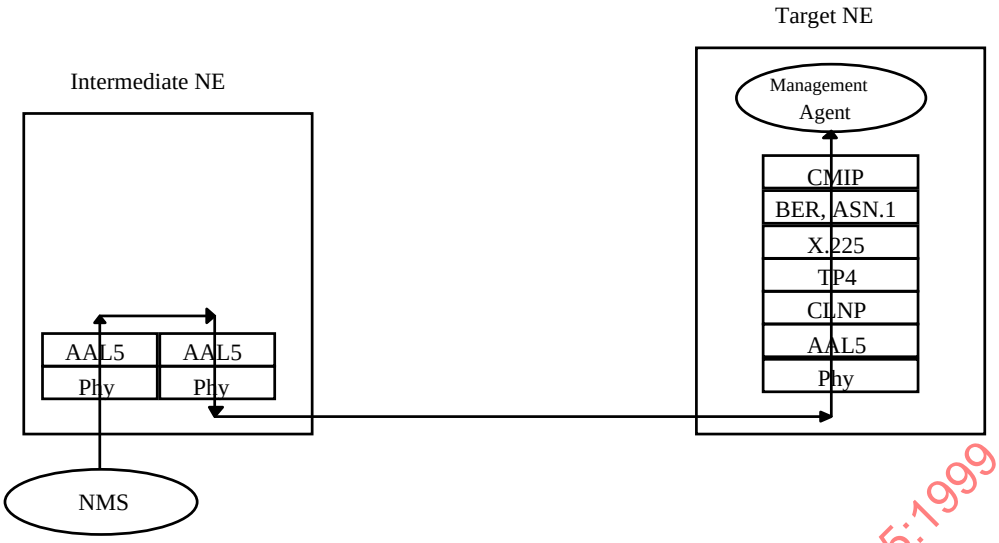


Figure 10.8 — Management Information flow for ATM based NMS

For NMS that use non-ATM based management interface, the NE that terminates the physical layer of the non-ATM based management interface is responsible for routing the management flow to the appropriate ATM virtual circuit that reaches the addressed managed network element.. The routing of management information is shown as below:

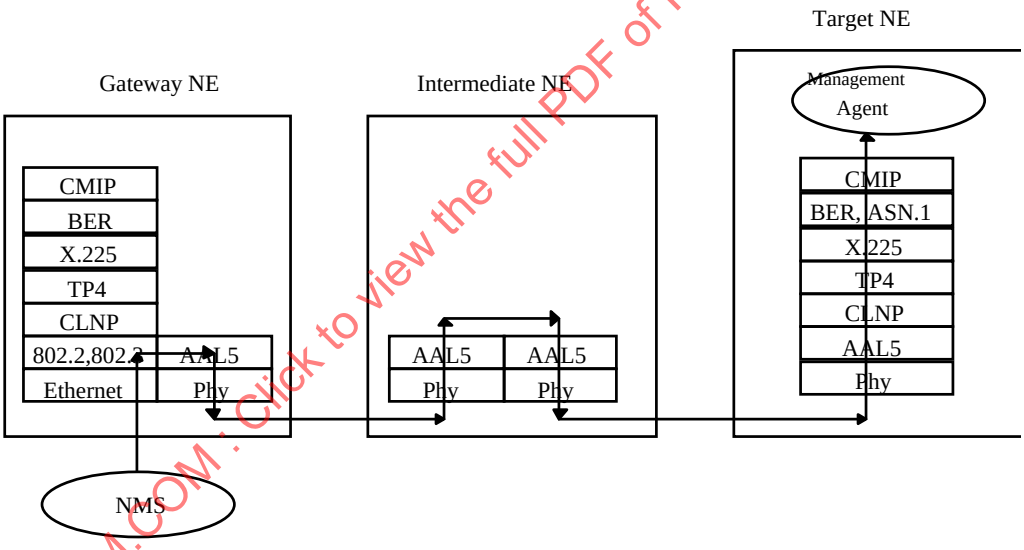


Figure 10.9 — Management Information Flow for non-ATM based NMS

10.4 Transport layer and network layer protocols

This section specifies only the transport and the network layer protocols which are specific for the S5 flow. The other possible protocols, i.e., TCP, UDP, and IP are described in clause 11 “common protocols”.

10.4.1 Transport layer protocols

10.4.1.1 TP4

The TP4 protocol provides for connection-oriented assured data delivery between two endpoints. TP4 is specified in ISO/IEC 8073 Transport Layer Protocols.

10.4.1.2 TP0

The TP0 protocol maps the session layer into the network layer. TP0 is specified in RFC 1006.

10.4.2 Network layer protocols

10.4.2.1 CLNP

CLNP provides connectionless network layer delivery services of packets between two endpoints. CLNP is defined is ISO 8473.

10.5 Management Information Bases

This subclause presents the possible Management Information Bases (MIB) for all of the high-level network elements: server, core network, access network, and STU.

10.5.1 STU MIB

The management protocol to be used by the STU is SNMP.

Many delivery systems may include ATM to the set-top. ATM Layer Fault management (M-plane) includes alarm surveillance, connectivity verification, and invalid VPI/VCI detection. End-to-end loop-back and UNI loop-back are supported.

The STU MIB is defined to be under the DAVIC (enterprise) node as registered with the IETF. The naming structure approach facilitates future version numbering of the STU MIB. This is summarized as:

DAVIC OBJECT IDENTIFIER ::= { enterprises 1493 }

DAVIC STU OBJECT IDENTIFIER ::= { DAVIC 1 }

DAVIC STU v1 MODULE-IDENTITY [.. data omitted ..] ::= { DAVIC STU 1 }

Future DAVIC MIBs (e.g., server or network MIBs) may be defined as subsidiaries to the DAVIC node (1493). It is DAVIC's responsibility to manage the entire section under the DAVIC node. For further information, contact the DAVIC Secretariat.

- The STU MIB is expressed using the Macros and style of SNMPv2. This is the current recommended method for specifying MIBs under IETF (Internet Engineering Task Force). This does *not* imply that SNMPv2 must be used. SNMPv1 (i.e., "ordinary" SNMP) may be used with this MIB.
- SNMP is implemented over UDP/IP. This is consistent with the protocol stacks defined in clause 10, "S5 flow: high and mid layer protocols".
- The STU MIB contains only management information related to the STU, not the NIU. An object ('stuNiuSpecific') is present to provide a "hook" from the STU MIB to additional NIU-specific management information, if there is an available MIB for the NIU.
- Access to MIB information may be made by an agent in the access network (i.e., a "Level 1" manager) or from a service provider (i.e., a "Level 2" manager). Security of access to MIB information is a matter of administrative policy implemented in the STU management agent, and is not explicit in the MIB definition. The MIB definition simply requires that some objects be "read-only" (i.e., not directly modifiable by a remote manager) whereas others are "read-write" and so may be updated by an authorized management entity.
- Support for DSM-CC user compatibility management. The entire DSM-CC User Compatibility (UC) string is encapsulated as a STU MIB object. This provides a linkage between DSM-CC User Compatibility and the DAVIC STU MIB, and yet allows modification to UC to be transparent to the STU MIB.
- A version mechanism to allow for future additions and changes.
- Traps may be enabled or disabled on a trap type basis.

10.5.2 Server MIB

The management protocol to be used by the server is SNMP. The server will use MIB2 specified in RFC 1213. For ATM capable servers, additional managed objects shall be included from RFC 1595.

10.5.3 Delivery system elements

10.5.3.1 SONET/SDH MIBs

CMISE MIB defined by ITU-T is recommended. These MIBs include M.3100, X.721, G.774 series, and Q.822.

10.5.3.2 Access network MIBs

CMIP or SNMP may be used for managing public access network equipment. For ATM-based access networks, a CMISE MIB is defined on ISO/IEC 16500-8.

10.5.3.3 Core network MIBs

CMIP is recommended for public network equipment though SNMP may also be used to manage core network elements. The ATM Forum has completed a CMIP MIB specification for M4 Network Element interface for ATM switches. The SNMP MIB specification work at the ATM Forum for the M4 Network Element interface is under progress.

10.6 Element capability profiles

The capability profiles are used to describe preferences and capabilities of STUs. In the future, capability profiles for users and customers will be defined.

10.6.1 STU Profile

The STU profile is defined in ISO/IEC 13818-1.

11. Common protocols

In this section, the protocol layers common to more flow protocol stacks are described. In particular depending on the specific scenarios, ATM and AAL5 could be common to S1, S2, S3, S4, and S5, TCP-UDP/IP could be common to S2, S3 and S5.

11.1 TCP

The TCP (Transmission Control) Protocol provides for connection-oriented assured data stream delivery between two endpoints. Delivery of “urgent data” is also supported.

TCP is specified in RFC 793 and is used for carrying control and management information.

11.2 UDP

The UDP (User Datagram) Protocol provides for un-assured delivery of packets between two endpoints.

UDP is specified in RFC 768 and is used for carrying control and management information.

11.3 IP

The IP (Internet) Protocol provides connectionless network-layer delivery services of packets between two endpoints.

IP is specified in RFC 791.

11.3.1 IP over ATM

When IP is tunneled within ATM, the LLC/SNAP format shall be used as defined in IETF RFC 1483 “Multi-protocol encapsulation over ATM Adaptation Layer 5”. The default Maximum Transfer Unit size shall be 9188 bytes as defined in IETF RFC 1577 “Classical IP and ARP over ATM”.

For each IP interface, there shall not be more than one ATM VC associated with the same destination IP address.

The server may assign a new IP address to the STB by means of a DSM-CC User-Network IP resource descriptor (see ISO/IEC 13818-6, clause 4, User-Network Session Messages). In this case, the STB shall configure a new logical IP interface with the assigned IP address.

For S2 flows only, the following applies. As defined in RFC 1577:

- If the STU sends an Inverse ATM Address Resolution Protocol InATMARP_REQUEST message, the server shall respond with the InATMARP_REPLY message.
- If the server sends an Inverse ATM Address Resolution Protocol InATMARP_REQUEST message, the STU shall respond with the InATMARP_REPLY message.

11.4 ATM Adaptation Layer Type 5

ATM Adaptation Layer Type 5 (AAL5) shall comply to ITU-T Rec. I.363.5.

11.5 ATM layer specifications

The ATM layer, and the ATM cell formats (for UNI and for NNI) are as specified in ITU-T Rec. I.361. The format of the ATM cell at the UNI (A1 reference point) is consistent with ITU-T Rec. I.361 both for the ATM end-to-end case and for the Non-ATM end-to-end case (e.g., Non-ATM Passband bi-directional PHY over coax).

In the following reference is always made to the ATM end-to-end case. The difference for the Non-ATM end-to-end case are noted explicitly for each of the clauses concerned.

In DAVIC systems, where there is a need for Intra-Network cells (e.g., at the A4 interface), then the NNI cell format shall be used as the intra-network cell format.

11.5.1 Reserved VPI/VCI values for uni-format cells

The reserved VPI/VCI values for the UNI are as specified in ITU-T Rec. I.361.

11.5.2 Reserved VPI/VCI values for NNI-format cells

The reserved VPI/VCI values for NNI-format cells are as specified in ITU-T Rec. I.361.

11.5.3 Reserved VPI/VCI values for intra-network cells

Table 11.1 shows the VPI and VCI values used for Intra-network-format cells in DAVIC systems. The table is derived from I.361, with some modifications in order to cater for the requirements of intra-network transmission. In particular, modifications are needed for the presence of meta-signaling, general broadcast signaling, and point-to-point signaling UNI channels, which may be multiplexed onto an Intra-Network link.

Table 11.1 — Reserved VCI and VPI values for use at the A4 interface

USE	VPI	VCI (Note 8)	PTI	CLP
Unassigned cell	000000000000	00000000 00000000	Any value	0
Invalid	Any VPI value other than 0	00000000 00000000	Any value	B
Meta-signaling	Any VPI value (Note 1)	00000000 00000001 (Note 5)	0AA	C
General broadcast signaling	Any VPI value (Note 1)	00000000 00000010 (Note 5)	0AA	C
Point-to-point signaling	Any VPI value (Note 1)	00000000 00000101 (Note 5)	0AA	C
Segment OAM F4 flow cell	Any VPI value	00000000 00000011 (Note 4)	0A0	A
End-to-end OAM F4 flow cell	Any VPI value	00000000 00000100 (Note 4)	0A0	A
VP resource management cell	Any VPI value	00000000 00000110 (Note 9)	110	C
Reserved for future VP functions (Note 6)	Any VPI value	00000000 00000111	0AA	A
Reserved for future functions (Note 7)	Any VPI value	00000000 000SSSSS (Note 2)	0AA	A
Reserved for future functions (Note 7)	Any VPI value	00000000 000TTTTT (Note 3)	0AA	A
Segment OAM F5 flow cell	Any VPI value	Any VCI value other than 00000000 00000000	100	A
End-to-end OAM F5 flow cell	Any VPI value	Any VCI value other than 00000000 00000000	101	A
VC resource management cell	Any VPI value	Any VCI value other than 00000000 00000000 00000000 00000110	110	C
Reserved for future VP functions	Any VPI value	Any VCI value other than 00000000 00000000	111	A

A. Indicates that the bit may be 0 or 1 and is available for use by the appropriate ATM layer function

B. Indicates that the bit is a “don't care” bit

C. Indicates that the originating entity shall set the CLP bit to 0. The value may be changed by the network.

NOTES

1. VPI value equal to 0 of each customer specific (p-to-p) UNI (A1) carrying user signaling with the LEX, will be mapped onto an appropriate VPI at the A4 interface in accordance with Rec. I.311.
2. SSSSS: any value from 01000 to 01111.
3. TTTTT: any value from 10000 to 11111.
4. Transparency is not guaranteed for the OAM F4 flows in a user-to-user VP.
5. The VCI values are pre-assigned in every VPC at the UNI. The usage of these values depends on the actual signaling configurations (see Rec. I.311).
6. This VCI value is reserved to provide the same function for VPs as PTI 111 is reserved to provide for VCs.
7. This VCI value is reserved for future standardization for specific functions
8. Cells with VCI values 1, 2, 5, 16 through 31, and greater than 31 are monitored by the VP OAM function. Cells with other VCI values are not monitored by the VP OAM function (see Rec. I.610). Whether a cell with a particular VCI vale is conveyed transparently between the end points is described in Section 3.1.4.1 e) of Rec. I.150.
9. The VP resource management cell is identified by this VCI value regardless of the value in the PTI field.

11.5.4 ATM Layer OAM

DAVIC shall adopt the ATM layer OAM as standardized by ITU-T Rec. I.610.

The ATM layer contains two OAM levels:

- F4: virtual path level
- F5: virtual channel level

The F4 flow provides the OAM functions to support the virtual path connections, while the F5 flow provides the OAM functions to support the virtual channel connections.

In ISO/IEC 16500 end-to-end VPs are not supported and therefore the F4 flow is not required at the UNI.

For the Non-ATM end-to-end case, no ATM OAM via F4 and F5 flows is performed, both in the downstream and in the upstream channels.

11.5.5 Connection management functions for the non-ATM end-to-end case

The main connection management aspects between core network and access network for the Non-ATM end-to-end case (more specifically for the Non-ATM Passband bi-directional over coax) are the following:

1. Connections in support of U-N default sessions are pre-provisioned and overseen by the Network related control entity.
2. Connections in support of additional U-N sessions are established by proxy signaling from the entity implementing the DSM-CC Network functionality and the access network Proxy agent. The proxy agent issues MAC connection messages to the STU.
3. In either 1 or 2, several parameters (e.g., VPI, VCI, ...) are specified for the downstream and for the upstream portion of the connections in the lower layer protocols (see e.g., ISO/IEC 16500-4, Passband bi-directional PHY on coax, Media Access Control message definitions).
4. Management (e.g., blocking, failure and performance monitoring). As part of the overall access network OAM functions, the proxy agent shall monitor events reflecting the health of a connection, the performance (via logged statistics), and administration (inclusive of blocking).

12. Connection Block Descriptors and initialization protocols for A0

12.1 Connection Block Descriptors

The Connection Block Descriptor (CBD) is the identifier of delivery system connections on the end user (A1 and A0) interfaces. The CBD fulfills the need for a logical connection identifier analogous to the VPI/VCI on server/VASP (ATM) interface, but is able to accommodate a variety of loop distribution system (i.e., access sub-network) types including but not limited to those providing a traditional ATM interface.

Format

The CBD structure is as follows:

Field	**Length** (in bytes)
Connection Index	3
Optional Data Length (in bytes)	1
Optional Data	0-255

The Connection Index is a unique logical identifier of the connection. The Optional Data information may be included if the NIU/loop distribution system require more information than is provided by the Connection Index in order to identify a requested connection.

Procedures

The following are characteristics of the use of the CBD in a DAVIC system:

- The CBD value is assigned by the loop transport system (access sub-network), in response to a set-up request from the connection controller, and is passed via this controller to the STU as the connection identifier. Only

the loop transport system interprets and/or translates the value of the CBD (as required), thus making this format technology-independent. The NIU is assumed to be related to the loop distribution system such that it also understands any system/technology-specific interpretation of the CBD contents.

- The STU uses the CBD Connection Index in its request across the STU-NIU (A0) interface to identify the specific connection that is being requested. However, the STU does not interpret the CBD Optional Data field.
- The Connection Index portion of the CBD is used by the A0 Local Control Bus to identify the logical connection.
- For initialization, it cannot be assumed that a well known connection identifier will work feasibly for all network types. Therefore, as part of the ***Network_Status*** Confirm message (see subclause 12.5.5.1), the NIU will provide the default signaling CBD to the STU. This CBD shall contain sufficient information for the STU to perform the DAVIC Interface Initialization Protocol (DIIP), described in subclause 8.4. The NIU Configuration Table (see subclause 12.5.2.1) will also indicate if the NIU is has a broadcast A1/A1* interface (e.g., MMDS, Satellite), if the network SRM is a connection level or session level interface (e.g., Q.2931 or DSM-CC U-N), and if first party or third party signaling should be used. With this information, the STU will be able to send a LCONNECT message (see subclause 12.4.3.1) forwarding the default signaling channel of the network SRM over the A0 Bi-directional bus from the NIU to the STU.

Following are two example CBD usage types:

1. The CBD consists of a connection index only and no variable length information (i.e., Optional Data Length=0). This assumes that the NIU can locate a requested connection using only the connection index. This might be appropriate for a true ATM interface, where the connection index equals the VPI/VCI. This would also be appropriate where an intelligent NIU can translate between the logical connection index used on the A0 interface and the connection identifier information used on the A1 interface specific to the loop distribution system (e.g., frequency, MPEG program number, etc.).
2. The CBD consists of a connection index plus variable length (technology-specific) information that is used by the NIU/loop transport system to identify the connection. This is appropriate where the NIU is dependent on the CBD information to locate the requested connection and where use of the connection index alone is not sufficient (e.g., HFC or Broadcast Satellite delivery systems).

12.2 A0 protocol stacks

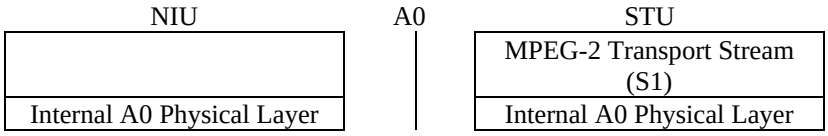
Protocol specifications for each of the three digital data buses of the A0 interface are shown below. These three buses are the High Speed Downstream Data Bus, the bi-directional Cell Bus, and the Local Control Bus. The Reset Bus and optional Analogue Pass-Through bus do not require any protocol specifications.

Specifications are provided for the two physical A0 interface types: External A0 and Internal A0. More information regarding these two physical implementations of the A0 interface may be found in ISO/IEC 16500-4 clause 8.

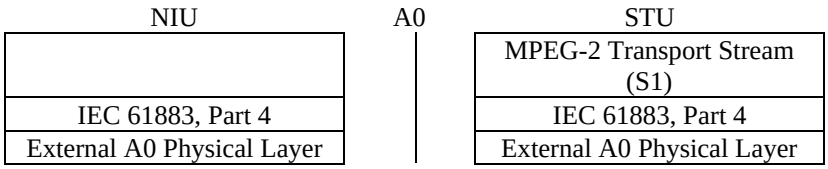
12.2.1 MPEG-2 TS downstream bus

This bus is the primary method to transmit S1 MPEG-2 transport based content across the A0 interface.

For the Internal A0 Interface:



For the External Interface:

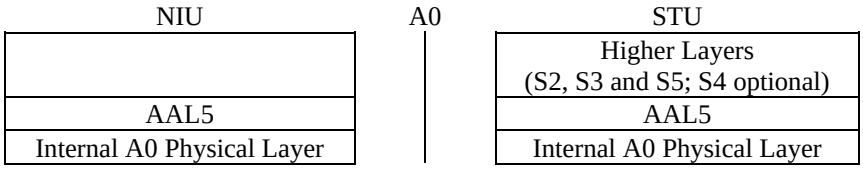


12.2.2 Bi-directional cell bus

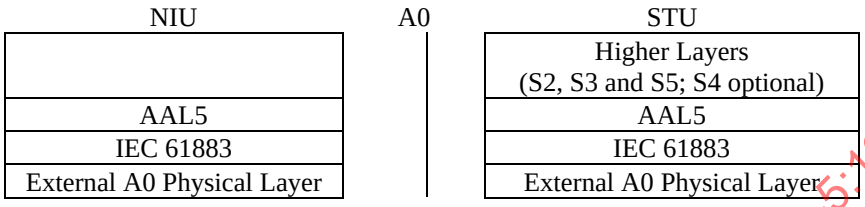
This bus provides the functionality for STU to communicate with the delivery system and VASP (DSM-CC User-to-Network, for example). The data crossing the A0 interface over this bus may include S1, S2, S3, S4, or S5 data flows. It should be noted that, in addition to the two physical Internal and External implementations, there are two levels defined for the bi-directional Cell Bus, Level “A” and Level “B”. For the Level “A”, ATM and AAL5 are simply used on the bi-directional cell bus as a frame format. The support of either OAM cells or header error checking are not required on the STU for this bus. For Level “B”, all ATM cells (including OAM cells) are passed to the STU for termination.

12.2.2.1 Level "A" interfaces

For the A0 Internal Interface, Level "A":

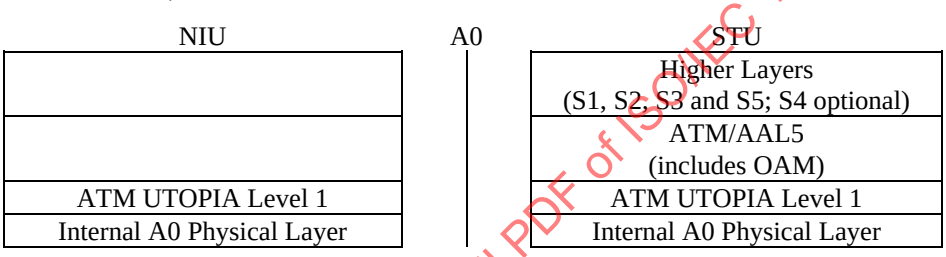


For the A0 External Interface, Level "A":

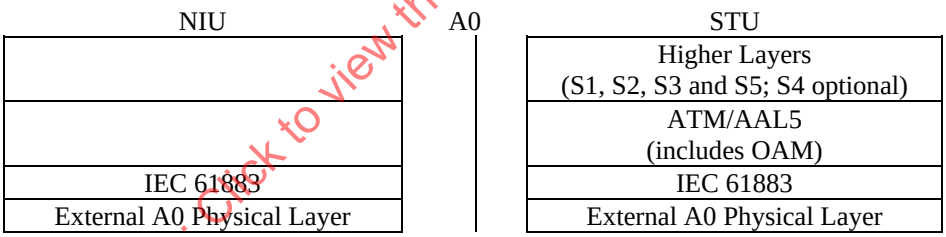


12.2.2.2 *Level "B" A0 interfaces (valid for ATM systems only)*

For the A0 Internal Interface, Level "B":



For the A0 External Interface, Level "A":



12.2.3 Local control bus

The local control bus provides a bi-directional path for the STU and NIU to exchange messages.

For the Internal A0 interface:

NIU	A0	STU
Local Initialization, CBD Control, Status and Diagnostic Functions		Local Initialization, CBD Control, Status and Diagnostic Functions
I^2C Protocol		I^2C Protocol
Internal A0 Physical Layer		Internal A0 Physical Layer

For the External A0 interface:

NIU	A0	STU
Local Initialization, CBD Control, Status and Diagnostic Functions		Local Initialization, CBD Control, Status and Diagnostic Functions
AV-CTS Command Frame Format		AV-CTS Command Frame Format
IEC 61883, Part l		IEC 61883, Part l
External A0 Physical Layer		External A0 Physical Layer

12.2.4 Optional analog pass-through bus (internal A0 only)

The Internal A0 specifications support the implementation of an optional analog pass through of baseband analog video, audio, and other text based services. For the Internal A0 interface, the NIU digitizes the analog baseband signal and passes it though to the STU. The External A0 interface simply provides a baseband analog signal for the STU to process. See ISO/IEC 16500-4 clause 8 for more information.

For the Internal A0 interface:

NIU			A0	STU
Analog Data	*Analog Video*	*Analog Audio*		
CCIR 656[1]	CCIR 656	I^2S		
A0 Internal Physical Layer				A0 Internal Physical Layer

12.3 STU/NIU initialization messages

12.3.1 Initialization functions

The A0 initialization functions are used to establish a basic level of service between the STU and the NIU. While both modules contain completely autonomous processors, the initialization functions provide the mechanism to co-ordinate and confirm that both modules are operational and able to communicate. After a reset, the NIU shall perform its Power On Self-Test (POST) procedure. Upon completion of the (POST), the NIU shall wait for a ***Wake_Up*** request from the STU (see subclause 12.3.2.1). The wake up messaging is always initiated by the STU upon completion of its POST. The STU shall not initiate a Wake_Up request until it has determined that the NIU has completed its clock synchronization on the control bus. While waiting for the ***Wake_Up*** request, the NIU may continue with its network synchronization, default connection establishment and ranging processes as appropriate. If any other request is received by the NIU before the ***Wake_Up*** request, an error indication shall be returned stating that a ***Wake_Up*** is expected.

After the ***Wake_Up*** messaging, the STU shall issue a ***Get_NIU_Config*** request (see subclause 12.5.4.4). The NIU shall reply with its current ***NIU_Config*** table (see subclause 12.5.2.1 for the table definition). This will allow the STU to determine, via the ***DAVIC_NIU_Sig_Type*** and ***DAVIC_SRM_Type*** fields, if the downloading of DAVIC client functionalities is required to perform signaling with Session Resource Control (SRC) or Network Control entities in the delivery system. Should the STU require client functionalities, it may query the NIU via the ***Image*** request. The ***Image*** message allows the STU to retrieve from the NIU network specific DAVIC client functionalities that are available on a data storage device located on the NIU (see clause 12.3.2.2 for more information).

Following the downloading of any client functionalities via the ***Image*** message exchange, the STU sends a ***Set_Physical_Address*** request. This writes the STU's ***Serial_Address*** as defined in the ***STU_Config*** table (see subclause 12.5.3.1 for information on the ***STU_Config*** table) in the NIU's physical address register. The NIU

may use this physical address to perform MAC-layer initialization functions with the delivery system. The NIU may respond to the STU by confirming the successful write of the physical address in the NIU's ***NIU_Config*** table, or by indicating that the ***Set_Physical_Address*** request was rejected (via codes in the ***Set_Physical_Address*** confirm message). If the NIU rejects the ***Set_Physical_Address*** request, the STU shall re-send the ***Set_Physical_Address*** request as defined by the <<timeout>> interval, and shall issue a ***Reset_NIU*** request if it exceeds the <<retry_count>> parameter. Following a successful ***Set_Physical_Address*** message exchange, the NIU shall issue the ***Network_Status*** message.

The ***Network_Status*** indication to the STU identifies the default/nailed up connection via the default **C**onnection **B**lock **D**escriptor (CBD) identifier. This is the only case where the NIU will first forward a CBD to the STU.

In some cases, the NIU may have the capability to establish this default connection identified by the default CBD. In other cases, additional signaling by the STU may be required to establish the default connection. See subclause 12.5.5.1 for more information regarding the ***Network_Status*** message format.

Following the ***Network_Status*** message, the STU shall perform the DAVIC Interface Initialization Protocol (DIIP) with either the Service Resource Control (SRC) or Network Control of the delivery system (see subclause 8.4). A result of the DIIP process is the assignment of a unique address for the NIU/STU pair, which will register a NIU/STU pair as a single terminal device. An E.164 form of **N**etwork **S**ervice **A**ccess **P**oint (NSAP) address shall be used as the unique NIU/STU interface address format.

Following a successful DIIP exchange by the STU, the STU shall inform the NIU of the NIU/STU network interface address via the ***Set_Network_Address*** request. This message provides the NIU with the NIU/STU address registered in the STU through the DIIP process.

Once the STU has received the ***Set_Network_Address*** confirm message format the NIU, the STU may request to get the network system clock from the NIU via the ***Get_Network_Time*** function. The NIU shall indicate if it is not able to provide the network system clock (as in the case where the NIU does not have the capability to establish the default signaling connection) in its response to the ***Get_Network_Time*** request message from the STU.

As a final step in the A0 initialization process, the NIU assesses the STU's configuration via the ***Get_STU_Config*** request (see subclause 12.5.5.3). The STU shall return to the NIU its ***STU_Config*** table (see subclause 12.5.3.1 for table definition) via the ***Get_STU_Config*** confirm message in order to enable any NIU assessment of the STU's capabilities.

While not specifically an initialization function, the ***STU_Alive*** message (subclause 12.5.4.3) may be used by the STU to periodically monitor the status of the NIU. The normal initialization messaging between the NIU and STU is shown in Figure 12.1.

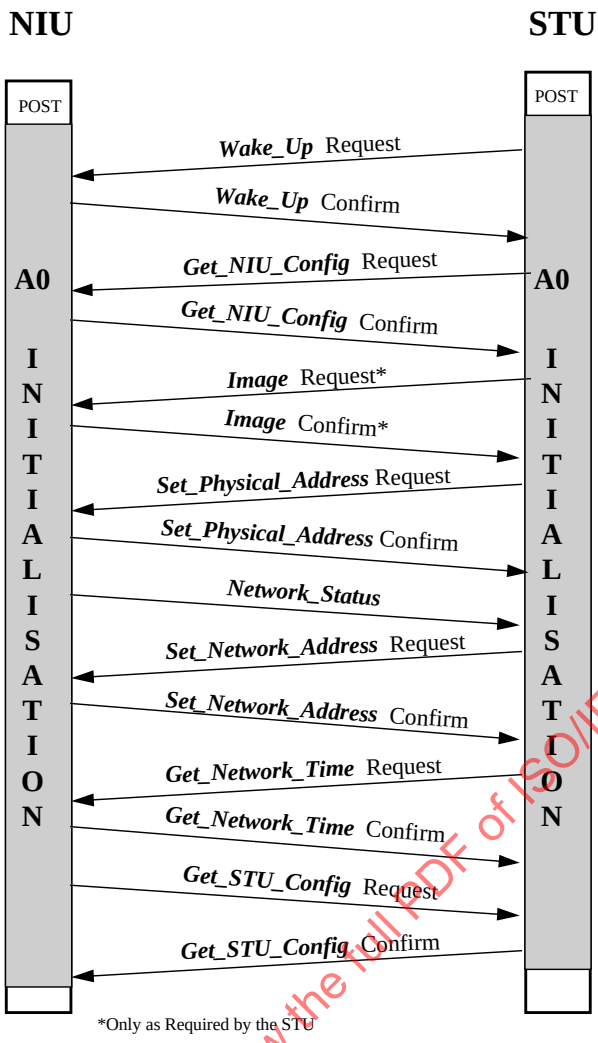


Figure 12.1 — A0 Initialization Flow

After the initialization messages are exchanged, network traffic is enabled to or from the STU on the default connection to the Session Resource Control (SRC) or Network Control entities. Specific connections must be requested (and authorized) before any A0 Local CBD Management messages are exchanged between the STU and NIU.

12.3.2 STU initialization messages

This section describes the messages that are supported and initiated by the STU system to the NIU for initialization across A0.

12.3.2.1 *Wake_Up* Request and *Wake_Up* Confirm

After a power-up or reset, the NIU will complete its POST process and wait for a ***Wake_Up*** request. The Wake_Up request is a Local Control Bus broadcast message type. If any other request is received by the NIU before the ***Wake_Up*** request, an error indication (Response Code = '01H') will be returned stating that a ***Wake_Up*** is expected.

Wake_Up Request (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Wake_Up*** Request Code *(00H)*
3	3-6	RESERVED by DAVIC for Future Use

Wake_Up Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Wake_Up*** Confirm Code *(03H)*
3	3	Command Response Code 00H - ***Wake_Up*** accepted 01H - ***Wake_Up*** command not received FFH - ***Wake_Up*** rejected
4	4-7	RESERVED by DAVIC for Future Use

After sending a ***Wake_Up*** request, the STU shall start a <<timeout>> timer and wait for a ***Wake_Up*** confirm from the NIU. If the timer expires before a confirm is received from the NIU, or if a confirm is received with a result code different from "***Wake_Up*** accepted" (e.g., 00H), the STU shall send additional ***Wake_Up*** requests to allow for the NIU to either complete its POST or some other process that may prevent it from responding to a ***Wake_Up*** request. After <<retry_count>> attempts, the STU shall assume the NIU is not present and may enter a mode of operation that does not use the NIU's services.

For the External A0 NIU only (which is based on IEEE 1394), on reception of the ***Wake_up*** message the NIU will allocate the appropriate bandwidth and channel resources required for the High Speed Downstream Data Bus and the Bi-directional Data Bus. The NIU shall establish the connections by means of the plug control registers as defined in IEC 61883, Part 1. For the High-Speed Downstream data bus the plug control registers oPCR[0] and iPCR[0] shall be used whereas for the Bi-directional Cell Bus the plug control registers oPCR[1] and iPCR[1] shall be used.

The establishment of plug control registers is not required for Internal A0 NIUs.

12.3.2.2 Image

The ***Image*** message is used by the STU to request that the NIU send via the Local Control Bus the binary image that is known to the NIU and STU by the *Module ID* (as defined in Table 12.1). The STU determines if it requires any DAVIC client functionalities by assessing the NIU's ***NIU_Config*** table (as defined in subclause 12.5.4.4) via the ***Get_NIU_Config*** (see subclause 12.5.2.1). If the STU is not in need of any software modules, the ***Image*** message exchange is not required.

Image Request (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	Message Type *(A2H)*
3	3	*Module ID* (see Table 12.1 for values)

If the *Module ID* contains NULL bytes then the NIU shall provide the entire contents of its data store.

Image Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	Message Type *(A5H)*
3	3	Response Code: 0x01 - *Module ID* found 0x04 - Function not supported 0x02 - *Module ID* not found
4	4-7	Variable Data Length (for Parameters 5 and 6)
5	8-*n*	Variable Data Bytes formatted via DSM-CC Downloading protocol (ISO/IEC 13818-6, clause 7)
6	(*n*+1)-(*n*+4)	CRC

Image Confirm is sent only in response to an ***Image*** Request message and shall contain the requested module as identified by the *Module_ID* parameter. The response code parameter communicates the status of the response. The 'Variable Data Length' is a 32 bit unsigned integer, most significant bit first, that is a count in bytes of 'Variable Data Bytes' that immediately follow the 'Variable Data Length' parameter. The Variable Data Bytes are the actual image, there shall be exactly 'Variable Data Length' bytes of data. The CRC parameter is a 32 bit CRC as defined in ISO/IEC 13818-1. The CRC shall be calculated over parameters 3,4 and 5 inclusive.

Table 12.1 shows the unique *Module_IDs* for specific functionalities in DAVIC. This table has been constructed such that new IDs may be added in future DAVIC versions. Further, there are user defined fields to accommodate functionalities not covered in DAVIC specifications.

Table 12.1 — Summary of *Module ID* Values

Module_ID	DAVIC Functionality
0x00	Channel Change API
0x01	DSM-CC User to Network Session Signaling
0x02	DSM-CC Channel Change Protocol
0x03	Q.2931 Connection Signaling
0x04 to 0x1F	Reserved for future DAVIC use
0x20 to 0xFF	User Defined

12.3.2.3 Set_Physical_Address

The ***Set_Physical_Address*** function is used to define the NIU physical address registered in the NIU's Dynamic Configuration table (as defined in subclause 12.3.2.3). The STU provides its serial address (as defined in the ***STU_Config*** table in subclause 12.5.3.1) to the NIU. The NIU may use this serial address as its physical address for performing MAC-layer initialization functions with the delivery system.

Set_Physical_Address Request (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Set_Physical_Address*** Request Code *(04H)*
3	3-10	NIU Physical Address (equal to STU Serial Address)

Set_Physical_Address Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Set_Physical_Address*** Confirm Code *(07H)*
3	3	Command Response Code 00H - NIU physical address registered FFH - Registration rejected

The NIU may respond to the STU by confirming the successful write of the physical address in the NIU's ***NIU_Config*** table ('00H' code), or by indicating that the ***Set_Physical_Address*** request was rejected (FFH code). If the NIU rejects the ***Set_Physical_Address*** request, the STU shall re-send the ***Set_Physical_Address*** request as defined by the <<timeout>> interval, and shall issue a ***Reset_NIU*** request once it has exceeded the <<retry_count>> parameter.

12.3.2.4 Set_Network_Address

The ***Set_Network_Address*** function is used to define a new NIU/STU interface address that is stored in the STU's and NIU's Dynamic Configuration tables (see subclauses 12.5.3.1 and 12.5.2.1 respectively). The NIU/STU interface address is an E.164 ATM format NSAP (20 octet) value. The STU may use this function to clear the registered NIU/STU interface address value to a new E.164 NSAP format value or a null value, as deemed appropriate by the STU.

Set_Network_Address Request (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Set_Network_Address*** Request Code *(08H)*
3	3-22	New NIU/STU Interface Address (E.164 ATM Format NSAP)

Set_Network_Address Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Set_Network_Address*** Confirm Code *(0BH)*
3	3	Command Response Code 00H - NIU/STU interface address registered 01H - NIU/STU interface address cleared FFH - Registration rejected

12.3.2.5 Get_Network_Time

The ***Get_Network_Time*** function is used by the STU to get the current system time. The NIU may either maintain a real time clock from the periodic time-stamps that are received from the delivery system or it may simply wait for the next time-stamp message to forward to the STU. The STU shall maintain a real time clock, which it shall synchronize with the system clock at initialization and then every 24 hours thereafter. The system time is maintained as a 32-bit integer quantity that represents the number of **G**lobal **P**ositioning **S**ystem seconds since 12 am, January 6th, 1980. A GPS to UTC (Universal Coordinated Time) offset is also provided as an unsigned

integer. This offset which represents the whole seconds difference between the GPS and UTC standards is subtracted from the GPS time to yield UTC.

Get_Network_Time Request (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Get_Network_Time*** Request Code *(0CH)*

Get_Network_Time Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	***Get_Network_Time*** Confirm Code *(0FH)*
3	3	Command Response Code 00H - System time appended 0FH - System time not available FFH - Request Rejected
4	4-7	System Time (GPS Seconds)
5	8	GPS to UTC Offset
6	9-12	RESERVED by DAVIC for Future Use

12.4 A0 Local CBD management

The A0 CBD control functions are used to configure and control A0 interface resources for any application running on the STU. After the initialization process is completed, the NIU may be required to monitor and respond to MAC messaging from the Network Control, as well as handle data to or from the STU associated with the default connection only. The bulk of the A0 CBD control messages are associated with the establishment and release of connections between the NIU and STU to allow for changing network and application requirements following initialization.

12.4.1 Connection set-up

For STUs connected to access networks (A1 or A1*) with interactive back channel capability, all session/connection set-ups are initiated by the STU with a set-up request to the Session Resource Control or Network Control entities (via DSM-CC U-N or Q.2931 respectively). Once the set-up request is received by the Session Resource Control or Network Control, a process is started to set-up the connection requested. The NIU exchanges control messages particular to its access network to establish the requested connection to the NIU.

For STUs connected to delivery systems with broadcast only capability (e.g., Satellite, MMDS) the NIU shall provide Service Information (SI) data as specified in ETS 300 468 as referenced in ISO/IEC 16500-6 clause 6.5. Note that the SI is a mandatory component of any MPEG Transport Stream carried by a DAVIC-compliant delivery system. Within the SI, information is available about the physical parameters of the various MPEG-2 Transport Streams accessible via the delivery system. These parameters address: RF-frequency, FEC, demodulation etc. and are carried in delivery_system_descriptors as defined in clause 6.2.6 of ETS 300 468.

From the viewpoint of A0, the ***LCONNECT*** and ***LCONNECT_ACKNOWLEDGE*** messaging is required to complete existing connections from the NIU to the STU over the A0 Bi-directional Bus or the High Speed Downstream Bus. An ***LCONNECT*** request is sent by the STU to the NIU to forward the CBD information so the NIU may ‘route’ the existing connection to one of A0’s available data buses. The ***LCONNECT_ACKNOWLEDGE*** message is sent from the NIU to the STU to confirm that the NIU has performed the ‘route’ function.

In case of broadcast only delivery systems and if the ***LCONNECT*** message refers to the broadcast information, the ***LCONNECT*** message will contain the required physical parameters using a relevant delivery_system_descriptors as defined in clause 6.2.6 of ETS 300 468.

12.4.2 Connection release

The CBD release process is also initiated by a message sent by the STU. The CBD Connection Release process is only used to manage connections across the A0 Bi-directional bus. The High-Speed Downstream Bus and Analog Pass-Through Bus may only support one virtual connection at a time. Subsequently, for these "single connection" busses each ***LCONNECT*** message supersedes the previous ***LCONNECT*** message, and an ***LRELEASE*** message is not necessary.

For STU's connected to access networks with interactive back channel capability, DSM-CC U-N or Q.2931 messages will be exchanged with the Service/Connection Control to tear down the established connection to the NIU. An ***LRELEASE*** request is required by the STU to inform the NIU that the connection corresponding to a particular CBD has been released. Similarly, STUs connected to delivery systems with broadcast only capability may simply use the ***LRELEASE*** message to 'disconnect' a particular CBD connection from the Bi-directional Bus.

In either the interactive or broadcast cases, the NIU confirms the release of the particular CBD connection with the ***LRELEASE_ACKNOWLEDGE*** message. Further, the ***LRELEASE_ACKNOWLEDGE*** message may be used to indicate the NIU's failure to comply with an ***LCONNECT*** message for either the Bi-directional Bus or the High Speed Downstream Bus.

12.4.3 Local CBD messages

The format for these messages consists of a protocol discriminator, a message type field, a connection index, and various message dependent fields. The information contents for each message type differs and is described in the following table:

Message Type	Direction	Use	Information Element
LCONNECT *(Message Code = 11H)*	STU-->NIU	Request for a virtual connection to be forwarded from the NIU to the STU.	A0 Physical bus identifier, CBD Connection Index, CBD Optional Data Length, and CBD Optional Data.
LCONNECT_ ACKNOWLEDGE *(Message Code = 12H)*	NIU-->STU	Verify that requested virtual connection has been forwarded.	CBD Connection Index, CBD Optional Data Length, and CBD Optional Data.
LRELEASE *(Message Code = 21H)*	STU-->NIU	Notify the NIU that virtual channel is no longer/not currently available	CBD Connection Index, CBD Optional Data Length, and CBD Optional Data.
LRELEASE_ ACKNOWLEDGE *(Message Code = 22H)*	NIU-->STU	Verify to the STU that virtual channel is no longer/not currently available	Cause code, CBD Connection Index, CBD Optional Data Length, and CBD Optional Data.

12.4.3.1 The *LCONNECT* message

The ***LCONNECT*** message shall be sent by the STU instructing the NIU to forward a specific connection to the STU. This message contains a Connection Block Description (CBD) uniquely identifying a specific virtual connection established at the NIU. The CBD contains two fields. The first field is a Connection Index which uniquely identifies virtual connection established at the NIU. The second field called "CBD -- Optional Data" which contains necessary information for the NIU to find the requested connection (e.g., frequency, time slot, etc.). Finally, the ***LCONNECT*** message tells the NIU on which physical bus to 'route' the virtual connection to.

In case of broadcast only delivery systems and if the ***LCONNECT*** message refers to the broadcast information, the Connection Index shall be '000000' and the CBD field shall consist of the delivery_system_descriptor which applies to the particular delivery system which is being accessed by the NIU.

In case of an NIU equipped with a PSTN/ISDN return channel, and if the ***LCONNECT*** message refers to the information carried over the PSTN/ISDN return channel, the Connection Index shall be '000001' and the CBD field shall consist of the telephone_descriptor as specified in clause 6.2.27 of ETS 300 468.

The coding of the ***LCONNECT*** message payload is described in the following table:

LCONNECT (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	Message Type *(11H)*
3	3	Physical Port Identifier
4	4-6	CBD -- Connection Index
5	7	CBD -- Optional Data Length (in bytes)
6	8-262	CBD - Optional Data

The first octet of the message payload is a protocol discriminator, which is coded as "001."

The second octet of the message payload identifies the message type ("11H" is coded to identify a ***LCONNECT*** message).

The third octet of the message is the physical bus identifier. This value identifies which physical bus of the A0 interface that will carry the data for the CBD connection. These values are:

"1" = Bi-directional Control Bus

"2" = High-Speed Downstream Bus

"3" = Analog Pass-Through Bus

(See ISO/IEC 16500-4 for more information concerning these data buses.)

The fourth, fifth and sixth octets describe the unique Connection Index for the CBD.

The seventh octet describes the length (in bytes) of the Optional Data information portion of the Connection Block Descriptor (CBD). The size of this field fixes the maximum value of the variable length CBD information to 255. A value of 0 indicates no Optional Data (e.g., as in an end-to-end ATM delivery system).

The remaining portion of the ***LCONNECT*** message is the Optional Data information. This Optional Data is used by the NIU to in some delivery systems (e.g., frequency or time slot identifiers) to locate the requested connection.

12.4.3.2 The *LCONNECT_ACKNOWLEDGE* message

The ***LCONNECT_ACKNOWLEDGE*** message shall be sent from the NIU to the STU to verify the receipt of a ***LCONNECT*** message, and the successful forwarding of the requested connection. The ***LCONNECT_ACKNOWLEDGE*** message provides CBD connection information (local connection index and optional information) to confirm the STU's ***LCONNECT*** message.

The format for the ***LCONNECT _ACKNOWLEDGE*** message is described in the following table:

LCONNECT_ACKNOWLEDGE (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	Message Type *(12H)*
3	3-5	CBD -- Connection Index
4	6	CBD -- Optional Data Length (in bytes)
5	7-261	CBD -- Optional Data

The first octet of the message payload is a protocol discriminator, which is coded as "001."

The second octet of the message payload identifies the message type (“12H” is coded to identify a ***LCONNECT_ACKNOWLEDGE*** message).

Octets three to five contain the local Connection Index for the virtual connection that the NIU has acknowledged.

The remaining octets of the message consist of a Optional Data Length (1 octet) and Optional Data (up to 255 octets). The Optional Data Length defines the length of the optional information data (in bytes). A value of 0 in the information length field indicates no Optional Data.

12.4.3.3 The *LRELEASE* message

The ***LRELEASE*** message shall be sent from the STU to the NIU to indicate the disconnection of local connections across the A0 Bi-directional Bus. Note that for the High-Speed Downstream Bus and Analog Pass-Through Bus, the ***LRELEASE*** message is not required (each successive ***LCONNECT*** supersedes the previous ***LCONNECT*** message). The format for the ***LRELEASE*** message is described in the following table:

LRELEASE (STU to NIU)

Parameter	Byte #	Description
1	1	Protocol Discriminator *(001)*
2	2	Message Type *(21H)*
3	3-5	CBD – Connection Index
4	6	CBD – Optional Data Length (in bytes)
5	7-261	CBD – Optional Data

The first octet of the message payload is a protocol discriminator, which is coded as “001.”

The second octet of the message payload identifies the message type (“21H” is coded to identify a ***LRELEASE*** message).

Octets three to five contain the connection index to be released. Octet 6 provides the length (in bytes) of the Optional Data; a value of 0 indicates no Optional Data. This Optional Data, if present, begins with octet 7 and may continue for a total of 255 bytes (e.g., octet 261).

12.4.3.4 The *LRELEASE_ACKNOWLEDGE* message

The ***LRELEASE_ACKNOWLEDGE*** message shall be used to verify the disconnection of A0 interface resources, or simply to indicate the failure of a ***LCONNECT*** message. The NIU shall recognize the limits of both the delivery system virtual channel capacity and the A0 interface virtual channel capacity. The format of this message is described in the following table:

LRELEASE_ACKNOWLEDGE (NIU to STU)

Parameter	Byte #	Description
1	1	Protocol Discriminator (001)
2	2	Message Type (22H)
3	3	Cause Code (see below)
3	4-6	CBD -- Connection Index
4	7	CBD -- Optional Data Length (in bytes)
5	8-262	CBD -- Optional Data

The first octet of the message payload is a protocol discriminator, which is coded as “001.”

The second octet of the message payload identifies the message type (“22H” is coded to identify a ***LRELEASE_ACKNOWLEDGE*** message).

The ***LRELEASE_ACKNOWLEDGE*** contains a one octet code to describe the cause of the release. These release cause codes are:

01 = connection does not exist

02 = NIU capabilities exceeded

03 = delivery system capabilities exceeded

04	=	unspecified
05-FF	=	reserved for future use

Octets 4 to 6 contain the Connection Index that has been released. Octet 7 provides the length (in bytes) of the Optional Data; a value of 0 indicates no Optional Data. This Optional Data, if present, begins with octet 8 and may continue for a total of 255 bytes (e.g., octet 262).

12.5 Status and diagnostic functions

The A0 status functions are used to monitor configuration and statistics information on both the NIU and STU. Both modules maintain configuration and status tables that may be read by their respective peer. Most of the status functions defined in A0 are solicited configuration and status requests. The two exceptions to this are the ***STU_Alive*** and ***Network_Status*** indications.

The ***STU_Alive*** indication is a periodic message from the STU to the NIU to let the NIU know that it is still functional and active. The STU includes its current status in this indication and expects the NIU to respond with its status to complete this messaging.

The ***Network_Status*** indication is an unsolicited message from the NIU to inform the STU of a noteworthy change in the state of either the NIU or the access sub-network.

Both the NIU and the STU have the capability to get configuration and statistics information from their peer with the ***Get_NIU_Config***, ***Get_NIU_Stats***, ***Clear_NIU_Stats***, ***Get_STU_Config***, ***Get_STU_Stats*** and ***Clear_STU_Stats*** functions. The STU will use the information it gathers from the NIU for diagnostic applications. The NIU may gather the configuration and statistics information from the STU for a proxy network management agent.

12.5.1 Diagnostic primitives

A0 has the provision that allows the NIU to use the display resources controlled by the STU to display diagnostic messages. The NIU may forward ASCII text strings for display by the attached STU. The ***Display_Message*** primitive (see subclause 12.5.5.7) is used to accomplish this function.

12.5.2 NIU table formats

12.5.2.1 NIU configuration table format

The NIU maintains its configuration parameters in a table that may be requested by the STU via the ***Get_NIU_Config*** message (see subclause 12.5.4.4). The information in this table is static and is read only.

Parameter	Size (bytes)	Description
NIU_Static_Config_Table_ID	1	Unique identifier for the Static NIU Configuration Table.
Static_Config_Table_Length	1	Length of this table in bytes.
NIU_Physical_Address	6	Unique physical address for this NIU.
Major_Version	1	Major version number for this NIU.
Minor_Version	1	Minor version number for this NIU.
NIU_Broadcast	1	0 = None; 2 = MPEG SI table; 3 = Switched Video Broadcast
DAVIC_NIU_Sig_Type	1	0 = Q.2931; 1 = DSM-CC U-N; protocol for signaling to either the Session Resource Manager or Network Control entities in the delivery system.
DAVIC_SRM_Type	1	0 = First Party; 1 = Proxy; signaling mode to be used for Q.2931 signaling protocol to either Session Resource Manager or Network Control entity in the delivery system.
NIU_A0_Level	3	000 = Internal A0, Level “A”, no analog pass through 001 = Internal A0, Level “B”, no analog pass through 010 = External A0, Level “A”, no analog pass through 011 = External A0, Level “B”, no analog pass through 100 = Internal A0, Level “A”, analog pass through capable 101 = Internal A0, Level “B”, analog pass through capable 110 = External A0, Level “A”, analog pass through capable 111 = External A0, Level “B”, analog pass through capable
Max_PDU_Size	2	Maximum Packet Data Unit size (including all headers and overhead) that may be handled by this NIU on both the downstream and upstream channels, on the Bi-directional Cell Bus, measured in bytes.
Rx_Buffer_Capacity	2	Total number of bytes of receive (downstream) buffer space for the Bi-directional Cell Bus.
Tx_Buffer_Capacity	2	Total number of bytes of transmit (upstream) buffer space for the Bi-directional Cell Bus.

In addition to its static configuration, the NIU also maintains a dynamic configuration table. These values are read only to the STU and may also be obtained by the ***Get_NIU_Config*** message (see subclause 12.5.4.4).

Parameter	Size (bytes)	Description
NIU_Dynamic_Config_Table_ID	1	Unique identifier for the Dynamic NIU Configuration Table.
Dynamic_Config_Table_Length	1	Length of this table in bytes.
RESERVED	4	RESERVED by DAVIC for Future Use
NIU/STU_Interface_Address	20	Assigned NIU/STU interface address, E.164 ATM format NSAP. This is set via DIIP (subclause 8.4) or the ***Set_Network_Address*** message (subclause 12.3.2.4)
NIU_Logical_Address	5	Logical address derived from the NIU/STU_Interface_Address

12.5.2.2 NIU statistics table format

The NIU maintains network traffic statistics in a table that may be requested by the STU. This table may be cleared by the STU with the ***Clear_NIU_Stats*** function (see subclause 12.5.4.6).

Parameter	Size (bytes)	Description
NIU_Stat_Table_ID	1	Unique identifier for the NIU Statistics Table.
Stat_Table_Length	1	Length of this table in bytes.
Stat_Start_Time	5	Date/Time of Last ***Clear_NIU_Stats***
DS_Pkts_Total	4	Total Downstream SDUs Received (Non-Null)
DS_Pkts_Erred	2	Erred Downstream SDUs Received
App_SDUs_Rx	2	Application SDUs Received
App_SDUs_Tx	2	Application SDUs Transmitted
NACKS_Recvd	2	Total number of NACKs received for upstream traffic

12.5.3 STU table formats

12.5.3.1 STU configuration table format

The STU maintains its configuration parameters in a table that may be requested by the NIU via the ***Get_STU_Config*** message (see subclause 12.5.5.3). The information in this table is static and is read only.

Parameter	Size (bytes)	Description
STU_Config_Table_ID	1	Unique identifier for the STU Configuration Table.
Config_Table_Length	1	Length of this table in bytes.
Serial_Number	6	Serial Number for this STU.
Major_Version	1	Major version number for this STU.
Minor_Version	1	Minor version number for this STU.
STU_A0_Level	3	000 = Internal A0, Level “A”, no analog pass through 001 = Internal A0, Level “B”, no analog pass through 010 = External A0, Level “A”, no analog pass through 011 = External A0, Level “B”, no analog pass through 100 = Internal A0, Level “A”, analog pass through capable 101 = Internal A0, Level “B”, analog pass through capable 110 = External A0, Level “A”, analog pass through capable 111 = External A0, Level “B”, analog pass through capable

In addition to its static configuration, the STU also maintains a dynamic configuration table. These values are read only to the NIU, and may also be obtained by the ***Get_STU_Config*** message (see subclause 12.5.5.3).

Parameter	Size (bytes)	Description
STU_Dynamic_Config_Table_ID	1	Unique identifier for the Dynamic STU Configuration Table.
Dynamic_Config_Table_Length	1	Length of this table in bytes.
RESERVED	4	RESERVED by DAVIC for Future Use
NIU/STU_Interface_Address	20	Assigned NIU/STU interface address, E.164 ATM format NSAP. This is set via DIIP (subclause 8.4) or the ***Set_Network_Address*** message (subclause 12.3.2.4)
STU_Logical_Address	5	Logical address derived from the NIU/STU_Interface_Address

12.5.3.2 STU statistics table format

The STU maintains statistics in a table that may be requested by the NIU. This table may be cleared by the NIU with the ***Clear_STU_Stats*** function (see subclause 12.5.5.5).

Parameter	Size (bytes)	Description
STU_Stat_Table_ID	1	Unique identifier for the STU Statistics Table.
Stat_Table_Length	1	Length of this table in bytes.
Up_Time	4	Up time for this STU (time, modulo 2^{32} in hundredths of a second since the STU was last re-initialized)

12.5.4 STU to NIU messages

This section describes the messages that are supported and initiated by the STU to the NIU. In addition, the appropriate and expected NIU responses are defined.

12.5.4.1 STU_Shut_Down

The ***STU_Shut_Down*** function is used by the STU to indicate to the NIU that it will be resetting. All connections established by the STU shall be terminated and the NIU shall wait for a ***Wake_Up*** request from the STU as at power-up.

STU_Shut_Down Indication (STU to NIU)

Parameter	Byte #	Description
1	1	***STU_Shut_Down*** Indication Code (25H)
2	2	STU Shut Down Data 00H - Time out reset 01H - STU System error reset 02H - User Requested (Front Panel) 03H - Service Provider Requested FFH - Restart

STU_Shut_Down Response (NIU to STU)

Parameter	Byte #	Description
1	1	***STU_Shut_Down*** Response Code (26H)

12.5.4.2 Reset_NIU

Reset_NIU allows the STU to either perform a “soft” reset or completely re-initialize (“hard” reset) the NIU. A soft reset will cause the NIU to terminate all connections except for the default connection.

Once this message is received by the NIU, a response is sent to the STU and then the NIU shall execute the specified reset level. The NIU shall wait for a ***Wake_Up*** request from the STU following the reset.

Reset_NIU Request (STU to NIU)

Parameter	Byte #	Description
1	1	***Reset_NIU*** Request Code (28H)
2	2	Reset Level 00H - Soft reset FFH - Hard reset

Reset_NIU Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	***Reset_NIU*** Confirm Code (2BH)
2	2	Return Code 00H - Reset accepted FFH - Reset rejected

12.5.4.3 STU_Alive

The ***STU_Alive*** message originates from the STU and is used to indicate, on a regular frequency, that the STU is still functional and active. This message shall be sent to the NIU at least every <<STU_alive_period>>.

The message is comprised of the message header plus current status information regarding the state of the STU. The STU will always initiate the ***STU_Alive*** message exchange.

STU_Alive Indication (STU to NIU)

Parameter	Byte #	Description
1	1	***STU_Alive*** Indication Code (31H)
2	2-5	RESERVED by DAVIC for Future Use

STU_Alive Response (NIU to STU)

Parameter	Byte #	Description
1	1	***STU_Alive*** Response Code (32H)
2	2-5	RESERVED by DAVIC for Future Use

12.5.4.4 Get_NIU_Config

Get_NIU_Config returns the NIU's configuration tables to the STU. Section 12.5.2.1describes the format and content of these tables. The Static Configuration table is returned first, followed by the Dynamic Configuration Table. This information may be utilized by diagnostics applications within the STU system.

Get_NIU_Config Request (STU to NIU)

Parameter	Byte #	Description
1	2	***Get_NIU_Config*** Request Code (34H)

Get_NIU_Config Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	***Get_NIU_Config*** Confirm Code (37H)
2	2	Command Response Code 00H - Get configuration request accepted (NIU configuration table attached) FFH - Get configuration request rejected
3		NIU Static Configuration Table
	3	Unique identifier for the Static Configuration Table.
	4	Length of this table in bytes.
	5-10	NIU Physical Address
	11	Major version number for STU.
	12	Minor version number for STU.
	13	NIU Broadcast Capabilities
	14	NIU Signaling Capabilities (Q.2931 or DSM-CC U-N)
	15	NIU SRM Capabilities (first party or proxy)
	16	NIU A0 Capabilities
	17-18	Maximum Packet Data Unit size (in bytes)
	19-20	Receive Buffer Space (in bytes)
	21-22	Transmit Buffer Space (in bytes)
4		NIU Dynamic Configuration Table
	23	Unique identifier for Dynamic Configuration Table.
	24	Length of this table in bytes.
	25-28	RESERVED for future use by DAVIC (e.g., bit masked registers)
	29-48	Assigned NIU/STU interface address (E.164 NSAP)
	49-52	Logical NIU address (derived from E.164 NSAP)

12.5.4.5 Get_NIU_Stats

Get_NIU_Stats returns the NIU's statistics table to the STU. Section 12.5.4.5 describes the format and content of this table. This information will be utilized by diagnostics applications within the STU system.

Get_NIU_Stats Request (STU to NIU)

Parameter	Byte #	Description
1	1	***Get_NIU_Stats*** Request Code (38H)

Get_NIU_Stats Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	***Get_NIU_Stats*** Confirm Code (3FH)
2		NIU's Statistics Table
	2	Unique identifier for the NIU Statistics Table.
	3	Length of this table in bytes.
	4-8	Date/Time of Last ***Clear_NIU_Stats***
	9-12	Total Downstream SDUs Received (Non-Null)
	13-14	Erred Downstream SDUs Received
	15-16	Application SDUs Received
	17-18	Application SDUs Transmitted
	19-20	Total number of NACKs received for upstream traffic

12.5.4.6 Clear_NIU_Stats

Clear_NIU_Stats allows the STU to clear the statistics table maintained by the NIU (see subclause 12.5.4.6 for table format).

Clear_NIU_Stats Request (STU to NIU)

Parameter	Byte #	Description
1	1	***Clear_NIU_Stats*** Request Code (40H)

Clear_NIU_Stats Confirm (NIU to STU)

Parameter	Byte #	Description
1	1	***Clear_NIU_Stats*** Confirm Code (43H)
2	2	Command Return Code 00H - Clear statistics executed FFH - Clear statistics rejected

12.5.5 NIU-to-STU messages

This section describes the messages that are supported and initiated by the NIU to the STU. In addition, the appropriate and expected STU responses are defined.

12.5.5.1 Network_Status

The ***Network_Status*** indication is sent from the NIU to the STU upon the detection of a critical change in the NIU's or network's state. This indication may range from an event (such as the default connection being established) to a serious problem with the connection to the network (i.e., no communications with the head-end). No response is required to be sent from the STU to the NIU. This indication is unsolicited message to the STU.

Network_Status Indication (NIU to STU)

Parameter	Byte #	Description
1	1	***Network_Status*** Indication Code (8CH)
2	2	Status Descriptor Code 00H - Default connection established, Global Connection ID appended 01H - NIU controlled resource failure FFH - NIU fatal error detected, shutdown/reset eminent
3	3-5	Initialization CBD -- Connection Index
	6	Initialization CBD -- Optional Data Length
	7-261	Initialization CBD -- Optional Data

12.5.5.2 Reset_STU

Reset_STU allows the NIU to either perform a "soft" reset or completely re-initialize ("hard" reset) the STU. A soft reset will cause the STU to terminate all sessions and connections except for the default connection.

Once this message is received by the STU, a response is sent to the NIU and then the STU to execute the specified reset level. The STU shall send a ***Wake_Up*** request to the NIU following the reset.

Reset_STU Request (NIU to STU)

Parameter	Byte #	Description
1	1	***Reset_NIU*** Request Code (29H)
2	2	Reset Level 00H - Soft reset FFH - Hard reset

Reset_STU Confirm (STU to NIU)

Parameter	Byte #	Description
1	1	***Reset_NIU*** Confirm Code (2AH)
2	2	Return Code 00H - Reset accepted FFH - Reset rejected

12.5.5.3 Get_STU_Config

The ***Get_STU_Config*** messaging is initiated with a request by the NIU to the STU to forward its configuration tables (as defined in subclause 12.5.3.1). The Static Configuration table is returned first, followed by the Dynamic Configuration Table.

Get_STU_Config Request (NIU to STU)

Parameter	Byte #	Description
1	1	***Get_STU_Config*** Request Code (90H)

Get_STU_Config Confirm (STU to NIU)

Parameter	Byte #	Description
1	1	***Get_STU_Config*** Confirm Code (93H)
2	2	Command Response Code 00H - Get configuration request accepted (STU configuration table attached) FFH - Get configuration request rejected
3		STU Static Configuration Table
	3	Unique identifier for the Static Configuration Table.
	4	Length of this table in bytes.
	5-10	Serial number for this STU.
	11	Major version number for STU.
	12	Minor version number for STU.
	13	STU_A0_Level: STU A0 Capabilities
4		STU Dynamic Configuration Table
	14	Unique identifier for Dynamic Configuration Table.
	15	Length of this table in bytes.
	16-19	RESERVED by DAVIC for Future Use
	20-39	Assigned NIU/STU interface address (E.164 NSAP)
	40-44	Logical NIU address (derived from E.164 NSAP)

12.5.5.4 Get_STU_Stats

The ***Get_STU_Stats*** messaging is initiated with a request by the NIU to the STU to forward its statistics table (see subclause 12.5.5.4 for table format).

Get_STU_Stats Request (NIU to STU)

Parameter	Byte #	Description
1	1	***Get_STU_Stats*** Request Code (94H)

Get_STU_Stats Confirm (STU to NIU)

Parameter	Byte #	Description
1	1	***Get_STU_Stats*** Confirm Code (97H)
2	2	Command Response Code 00H - Get statistics request accepted (STU statistics table attached) FFH - Get statistics request rejected
3	 3 4 5-8	STU Statistics Table Unique identifier for table. Length of this table in bytes. Up time for this STU.

12.5.5.5 Clear_STU_Stats

Clear_STU_Stats allows the NIU to clear the statistics table maintained by the STU (see subclause 12.5.3.2 for table format).

Clear_STU_Stats Request (NIU to STU)

Parameter	Byte #	Description
1	1	***Clear_STU_Stats*** Request Code (98H)

Clear_STU_Stats Confirm (STU to NIU)

Parameter	Byte #	Description
1	1	***Clear_STU_Stats*** Confirm Code (9BH)
2	2	Command Return Code 00H - Clear statistics executed FFH - Clear statistics rejected

12.5.5.6 Get_STU_Qsize

Get_STU_Qsize returns the number of cells in the STU's Bi-directional Cell Bus buffer. This information may be used by the NIU to negotiate more upstream bandwidth in delivery systems capable of such features.

Get_STU_Qsize Request (NIU to STU)

Parameter	Byte #	Description
1	2	***Get_STU_Qsize*** Request Code (A0H)

Get_STU_Qsize Confirm (STU to NIU)

Parameter	Byte #	Description
1	1	***Get_NIU_QSize*** Confirm Code (A1H)
2	2	STU's Queue Size (in cells); if more than 255 cells waiting, value = 255

12.5.5.7 Display_Message

Display_Message allows the NIU to display diagnostic messages on the display resources maintained by the STU.

Display_Message Request (NIU to STU)

Parameter	Byte #	Description
1	1	***Display_Message*** Request Code (9CH)
2	2	LED bit mask
3	3-35	Message to be displayed (ASCII string, null terminated)

Display_Message Confirm (STU to NIU)

Parameter	Byte #	Description
1	1	***Display_Message*** Confirm Code (9FH)
2	2	Command Return Code 00H - Display message executed FFH - Display request rejected

12.6 Logical parameter definitions

This section contains the definitions for all logically defined parameters throughout this document.

<<***MaxDS_DU_Size***>> - Maximum Downstream Data Unit Size, the maximum size (in bytes) of downstream data units that may be handled by the NIU. This value is 1024 bytes.

<<***MaxUS_DU_Size***>> - Maximum Upstream Data Unit Size, the maximum size (in bytes) of upstream data units that may be handled by the NIU. This value is 1024 bytes.

<<***retry_count***>> - Number of times that a message is re-sent before assuming that the NIU or STU's respective peer is not responding. This value is 10.

<<***wake_up_timeout***>> - Time-out period that STU should wait after sending a Wake_Up request to the NIU before sending another Wake_Up request. This value is 5 seconds.

<<***STU_alive_period***>> - Frequency at which STU_Alive messages are sent to the NIU by the STU. This value is 5 seconds.

13. STU dataport

For ISO/IEC 16500, there are three tools described for the STU Dataport:

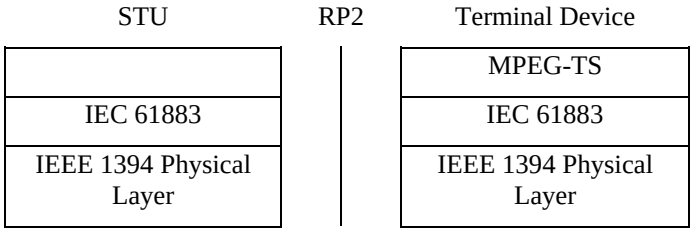
1. A *STU Multimedia Dataport (normative):* this interface is specified for the support of MPEG based services to peripheral video devices (such as DVD players and Digital VCRs) and support of IP services to peripheral PC devices. This is an IEEE 1394 interface.
2. A *STU PC Dataport (informative):* this interface is specified for the support of IP services to peripheral PC devices.
3. A *STU Parallel Dataport (informative):* this interface is specified for the support to commonly existing parallel peripheral devices (e.g., printers).

See ISO/IEC 16500-4 clause 9, for more information regarding the physical layer of the STU Dataport.

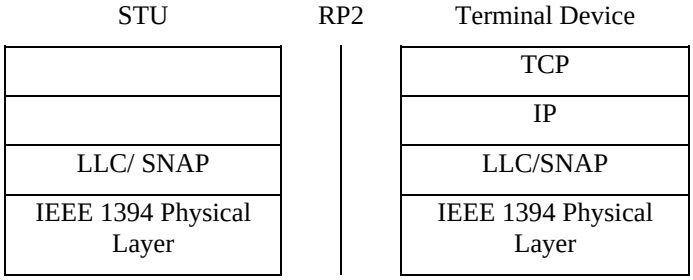
13.1 Protocol stacks for STU dataport

The following describe the protocol stacks for the two generalized services supported over the STU Multimedia Dataport. Protocol stacks for the informative specifications are not shown.

13.1.1 Protocol stacks for the STU multimedia dataport – MPEG based services



13.1.2 Protocol stacks for the STU multimedia dataport – IP based services



13.2 Support of IP services over the STU multimedia dataport

The support of IP services over IEEE 1394-1995 requires an asynchronous write packet with payload being an entire LLC/SNAP packet or possibly fragments of an LLC/SNAP packet. Fragments are necessary anytime the asynchronous write packet is smaller than the LLC/SNAP packet. This fragmentation is called link-fragmentation. Link-fragmentation is not to be confused with IP fragmentation.

13.2.1 MTU size

The MTU size is 2024 bytes. A 2024 byte MTU allows a single LLC/SNAP packet to be fragmented into four S100 asynchronous write packets. Furthermore, an MTU size of 2024 does not require fragmentation on S400 interfaces if a packet is equal to the MTU size.

This example of a 2024 byte packet over an S100 interface explains how the number 2024 bytes was derived. The size of the link-fragment header is 4 bytes; the LLC/SNAP header is 8 bytes. The first link-fragment includes the LLC/SNAP and link-fragment headers; the last three link-fragments include just the link-fragment header. The payload of the first link-fragment is 500 bytes; the payloads of the last three link-fragments are 508 bytes. The total payload is 2024 = 500 + (508 * 3).

13.2.2 Encapsulation

This section describes the encapsulation of all layers of all types of link-fragments. The layers include the IEEE 1394, link-fragment header, and the LLC/SNAP header. The link-fragment types are Begin of Packet (BOP), Continuation of Packet (COP), End of Packet (EOP), and Single Fragment Packet (SFP). See the Link-Fragmentation section for details of link-fragment types.

Table 13.1 specifies the entire encapsulation format of a the first fragment of an LLC/SNAP packet or a single link-fragment packet (SFP). This includes the 1394 header, the link-fragment header, and the LLC/SNAP header.

Table 13.1 — Encapsulation Format of BOP or SFP

quadlet	octet 1	octet 2	octet 3	octet 4
1	IEEE 1394 Async Write			
2	IEEE 1394 Async Write (cont.)			
3	IEEE 1394 Async Write (cont.)			
4	IEEE 1394 Async Write (cont.)			
5	IEEE 1394 Async Write (cont.)			
6	Link-fragment Header			
7	LLC			SNAP
8	SNAP (cont.)			
9	SNAP Packet Payload Data			

Table 13.2 specifies the format of link-fragments that are not the first link-fragment of the packet or are not a single link-fragment packet (SFP).There is no LLC/SNAP header.

Table 13.2 — Encapsulation Format of EOP or COP

quadlet	octet 1	octet 2	octet 3	octet 4
1	IEEE 1394 Async Write			
2	IEEE 1394 Async Write (cont.)			
3	IEEE 1394 Async Write (cont.)			
4	IEEE 1394 Async Write (cont.)			
5	IEEE 1394 Async Write (cont.)			
6	Link-fragment Header			
7	SNAP Packet Payload Data (continued from last link-fragment)			

Table 13.3 specifies the format of the 1394 Async write packet header.

Table 13.3 — Async Write Packet Header

quadlet	octet 1	octet 2	octet 3	octet 4
1	destination Node ID		tl rt	tcode pri
2	source Node ID		Offset	
3	Offset (cont.)			
4	data length		Extended TCode	
5	Header CRC			

Notes: The format of this packet matches a standard Async Write. These comments are only present to explain specific values shown in specific fields. For more details of the specific header fields, see IEEE 1394-1995 specification. The offset value of 0xFFFFFFFFFFFF is used for broadcast 1394 packets. 1394 broadcast messages include ARP requests. The offset specified in the ARP response information associated with a specific node must be used for unicast 1394 messages. ARP responses are 1394 unicast messages.

The source node ID from the 1394 async-write packet is used to associate a single Link-fragment with an LLC/SNAP packet. A sequence number is used to maintain the order of link-fragments.

Table 13.4 specifies the format of the link-fragment header.

Table 13.4 — Link-fragment Header Format

quadlet	octet 1	octet 2	octet 3	octet 4	
1	Stream Type = 0x00 (LLC/SNAP)	Source 1394 Node ID		seq num, 6 bits	Frag Type, 2 bits

Notes:

1. The value of the stream type shall be 0x00 to define LLC/SNAP streams.
2. The value of the source Node ID shall be the same as the source Node ID of the 1394 async write packet.. This value associates a link-fragment with a LLC/SNAP packet. This field shall be combined with the sequence number and link-fragment type to assemble a packet.
3. Link-fragments from a single LLC/SNAP packet are ordered and assembled using the sequence number. The sender shall continuously increment this number after every fragment is transmitted. This number shall NOT be reset to zero at the end of transmitting all the link-fragments of a single packet. The values of the 2 bit fragment types shall be 0x00 for single fragment packet (SFP), 0x01 for begin of packet (BOP), 0x02 for continuation of packet (COP), and 0x03 for end of packet (EOP).

LLC/SNAP specifically refers to IEEE 802.2 LLC and IEEE 802.1A Sub Network Access Protocol (SNAP). The SNAP header is used to identify the EtherType Code as listed in RFC 1700, “Assigned Numbers”. IEEE 802.2 LLC Type One Unnumbered Information (UI) communication shall be used exclusively.

Table 13.5 specifies the detailed format of the LLC/SNAP header and values.

Table 13.5 — LLC/SNAP Header and Values

quadlet	octet 1	octet 2	octet 3	octet 4
1	DSAP 0xAA	SSAP = 0XAA	CTRL = 0X03	Organ Code
2	Organ Code (cont.) = 0x000000		Ether Type = (0X0800,IP), (0X0806,ARP)	

Notes:

1. The total length of the LLC/SNAP header shall be 8 octets.
2. The value of DSAP and SSAP in the LLC header shall be 0xAA (hex).
3. The value of the Control (Ctrl) in the LLC header shall be 0x03 (hex).
4. The value of the Organization Code in the SNAP header shall be 0x000000 (hex).
5. The value of the Ether Type in the SNAP header shall be 0x0800 (hex) for IP or 0x806 (hex) for ARP.

13.2.3 Address Resolution Protocol (ARP)

In general, Address Resolution Protocol (ARP) consists of requests and responses to map unknown hardware addresses to known IP addresses. In the case of IEEE 1394, the hardware address is called the 1394 address. The 1394 address is a combination of the IEEE 1394 Bus ID, IEEE 1394 Phy ID, and IEEE 1394 offset. In addition to the 1394 address, ARP over 1394 requests retrieve a world wide unique id (WWUID) used to uniquely identify each node on the network. Other types of networks use the hardware address as the global unique identifier. The 1394 address cannot be a unique 1394 node Identifier because the node ID changes when a device is connected or disconnected from the network.

ARP requests shall be sent to the local bus broadcast Node ID of 0xFFBF with the offset of 0xFFFFFFFFFFFF. This Node ID consists of a 10 bit bus ID of 0x3FE and a 6 bit phy ID of 0x3F. The destination information in the payload of the ARP request is random and has no meaning. DAVIC suggests this portion of the packet be filled with ones.

ARP Responses shall be unicast messages. The destination information in the ARP response shall be taken directly from the source information in the ARP request. The source information in the ARP response shall be derived from information that resides in the responding node.

ARP caches shall be cleared on bus resets.

The format of the ARP request/response packet is shown in Table 13.6.

Table 13.6 — ARP Request/Response Packet

quadlet	octet 1	octet 2	octet 3	octet 4
1	Hardware Type = 0x18		Protocol Type = 0x0800	
2	SHWaLen	SWwuidLen	SIPaLen	DHWaLen
3	DWwuidLen	DIPaLen	Operation: ARP Request/Reply	
4	Source Node ID		Source IP Unicast Offset	
5	Source IP Unicast Offset (cont.)			
6	Source World Wide Unique ID			
7	Source World Wide Unique ID (cont.)			
8	Source IP Address			
9	Destination Node ID		Destination IP Unicast Offset	
10	Destination IP Unicast Offset (cont.)			
11	Destination World Wide Unique ID			
12	Destination World Wide Unique ID (cont.)			
13	Destination IP Address			

Notes:

1. The entire ARP header is not described here, just the portions relevant to this specification. Refer to RFC 826 for more details.
2. The value of the Hardware Type shall be 0x18.
   Note: This number is **not** an official IANA approved hardware type number.
3. SHWaLen and DHWaLen are the length of the source and destination addresses respectively. This includes the offset and the node id. The value of these fields is 0x08.
4. SIPaLen and SWwuidLen is the length of the World Wide Unique Id for the source and destination, respectively. The value of these fields is 0x08.
5. SIPaLen and DIPaLen is the length of the IP address for the source and destination, respectively.
6. The IP Unicast Offsets is used to send all unicast messages by that particular node. It may be 0xFFFFFFFFFFFF which is the same as broadcast messages or some other node unique value.
7. The World Wide Unique IDs (source and destination) are defined by IEEE 1394-1995.

13.2.4 Link-fragmentation

LLC/SNAP packets encapsulate IP or ARP packets. The link-fragments encapsulate portions of, or the entire, LLC/SNAP packet. There are four types of link-fragments: Begin of Packet (BOP), Continuation of Packet (COP), End of Packet (EOP), and Single Fragment Packet (SFP).

The link-fragment type of SFP is used if the entire LLC/SNAP packet is able to fit into a single link-fragment. Otherwise, the first portion of the LLC/SNAP packet is placed in a BOP link-fragment. The last portion of the LLC/SNAP packet is placed in the a EOP link-fragment. The middle portions are of the LLC/SNAP packet are placed in COP link-fragments.

The IEEE 1394 interface types are S100, S200, and S400. Requirements for the maximum size of link-fragment payloads are different for each.

- The maximum payload of a S100 link-fragment packet is 508 bytes. The number is derived by subtracting the size of the link-fragment header from the maximum payload of a S100 packet (i.e., 512-4).

- The maximum payload of a S200 link-fragment packet is 1020 bytes. The number is derived by subtracting the size of the link-fragment header from the maximum payload of a S200 packet (i.e., 1024-4).
- The maximum payload of a S400 link-fragment packet is 2024 bytes. The number equals the MTU size.

The recommended minimum sizes for S100, S200, and S400 interfaces are the same. There are no required or suggested minimum sizes for SFP and EOP link-fragments. The recommended minimum link-fragment size for BOP and COP link-fragments is 64 bytes.

13.2.5 IP unicast messages

IP Unicast messages use bus ID of 0x3ff and the node specific phy ID.

13.2.6 IP multicast and broadcast

IP Multicast and Broadcast messages shall be mapped to IEEE 1394-1995 broadcast destination Node IDs of 0xFFBF with an offset of 0xFFFFFFFFFFFF. The Node ID of 0xFFBF is the same as a bus ID of 0x3FF and the phy ID of 0x3F.

Annex A
(normative)
STU Management Information Base

```
DAVIC-SET-TOP-UNIT-MIB DEFINITIONS ::= BEGIN
IMPORTS
 MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, Integer32, IpAddress
  FROM SNMPv2-SMI
 TEXTUAL-CONVENTIONS, DateAndTime, DisplayString,
 TimeInterval, TimeStamp, TruthValue
  FROM SNMPv2-TC
 PhysAddress
  FROM RFC-1213;

DAVIC OBJECT IDENTIFIER ::= { enterprises 1493 }
davic stu OBJECT IDENTIFIER ::= { DAVIC 1 }

DAVIC STU v1 MODULE-IDENTITY
 LAST-UPDATED "9512151200Z"
 ORGANIZATION "DAVIC STU Technical Committee"
 CONTACT-INFO
  “DAVIC Secretariat”

 DESCRIPTION
        "This MIB module describes generic objects for managing the access architecture independent aspects of
        the Set Top Unit."
 ::= { davic stu 1 }

-- This section contains range definitions used below.

TestStatus ::= INTEGER { passed(1), failed(2), aborted(3) }

TestIndicator ::= INTEGER { active(1), inactive(2) }

AlarmStatus ::= INTEGER { active(1), cleared(2) }

-- The STU General Group
--
-- The Set Top Unit group is used to describe the attributes of the STU.
-- It also ties together through a variety of relationships, the
-- components to the STU. The attributes of the STU include addressing
-- information, clock information, assignment information. and customer
-- information.

stuGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 1 }

stuAdminState OBJECT-TYPE
 SYNTAX INTEGER { unlocked(1), locked(2) }
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The administrative state of the STU. The STU may be locked or unlocked by the manager. The STU
        may by locked if an event such as a security violation occurs."
 ::= { stuGroup 1 }

stuSysUpTime OBJECT-TYPE
```

```
 SYNTAX TimeTicks
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The time (in hundredths of a second) since the management portion of the STU was last re-initialized."
 REFERENCE
  "Derived from RFC1213-MIB.sysUpTime."
 ::= { stuGroup 2 }

stuSysContact OBJECT-TYPE
 SYNTAX DisplayString (SIZE (0..255))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The textual contact information for the authority that is responsible for the management of the STU."
 REFERENCE
  "Derived from RFC1213-MIB.sysContact."
 ::= { stuGroup 3 }

stuServiceLocation OBJECT-TYPE
 SYNTAX DisplayString (SIZE (0..255))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "A textual description of the physical service location assigned to the STU."
 REFERENCE
  "Derived from RFC1213-MIB.sysLocation."
 ::= { stuGroup 4 }

stuManufacturerOUICode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (3))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Manufacturer Organizational Unique Identifier, assigned to the manufacturer according to IEEE-
        802.1990"
 ::= { stuGroup 5 }

stuModelNumberCode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "A textual identifier of the manufacturer's model number of the STU."
 ::= { stuGroup 6 }

stuSerialNumberCode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "A textual identifier of the manufacturer's assigned serial number of the STU."
 ::= { stuGroup 7 }

stuEncryptionKey OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
```



```
        "This is conditional on whether encryption is supported. The encryption key is encoded with a Key
        Encryption Key (KEK). This object may be updated and alter the encryption key. This update may only
        take place if the correct KEK is used."
 ::= { stuGroup 8 }

niuMACAddr OBJECT-TYPE
 SYNTAX PhysAddress
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The physical address of the NIU. This represents the IEEE 802 MAC address. This is the hardware
        address meaningful in the communications domain that the STU's NIU is connected."
 REFERENCE
  "Derived from RFC1213-MIB.ifPhysAddress."
 ::= { stuGroup 9 }

stuE164Address OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE(8))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "An address assigned for administrative purposes, using the syntax of the E.164 address"
 ::= { stuGroup 10 }

stuIpAddr OBJECT-TYPE
 SYNTAX IpAddress
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
  "The IP address assigned to this STU."
 REFERENCE
  "Derived from RFC1213-MIB.ipAdEntAddr."
 ::= { stuGroup 11 }

stuClockTime OBJECT-TYPE
 SYNTAX DateAndTime
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The clock time of the STU. May be synchronized on occasion through the management interface."
 ::= { stuGroup 12 }

stuPowerOnSelfTestStatus OBJECT-TYPE
 SYNTAX TestStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Used to indicate the result of the last power on self test. The value is set to passed, failed, or aborted."
 ::= { stuGroup 13 }

stuPowerOnSelfTestErrorCode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The most recent power on self test error or status code defined by the STU supplier."
 ::= { stuGroup 14 }

stuPowerOnSelfTestTimeStamp OBJECT-TYPE
```

```
 SYNTAX TimeStamp
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The time that the stuPowerOnSelfTestErrorCode was most recently set."
 ::= { stuGroup 15 }

stuUserCompatibility OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "A string that represents the user compatibility information as defined by DSM-CC. This string may be
        updated by the STU manager."
 ::= { stuGroup 16 }

stuDavicLevel OBJECT-TYPE
 SYNTAX DisplayString (SIZE (0..255))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The DAVIC release or level supported by this STU."
 ::= { stuGroup 17 }

stuNiuSpecific OBJECT-TYPE
 SYNTAX OBJECT IDENTIFIER
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "A reference to MIB definitions specific to the network interface unit (NIU). If this information is not
        present, its value should be set to OBJECT IDENTIFIER { 0 0 }. The specific extensions may describe
        functions and capabilities that are NIU specific."
 ::= { stuGroup 18 }

stuVendorSpecificExtension OBJECT-TYPE
 SYNTAX OBJECT IDENTIFIER
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "A reference to MIB definitions specific to the vendor's STU implementation. If this information is not
        present, its value should be set to OBJECT IDENTIFIER { 0 0 }. The specific extensions shall only
        describe functions and capabilities that are vendor specific, beyond those functions and capabilities
        expressed in the DAVIC MIB(s)."
 ::= { stuGroup 19 }

-- The STU Status Information Group
--
-- The Set Top Unit status information group is used for storing
-- connectivity and service access attempt information of the Set Top
-- Unit and may be used in sectionalization of troubles.

stuStatGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 2 }

ConnectStatus ::= INTEGER { connected(1), unconnected(2) }

stuStatL1ConnectStatus OBJECT-TYPE
 SYNTAX ConnectStatus
 MAX-ACCESS read-only
 STATUS current
```



```
 DESCRIPTION
        "Used to indicate the connectivity status of the STU. The value connected means that the underlying
        transport path (Level 1 communications path) is available for supporting user services. The value
        unconnected means that the underlying transport path is unavailable for supporting user services."
 ::= { stuStatGroup 1 }

stuStatL2ConnectStatus OBJECT-TYPE
 SYNTAX ConnectStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Used to indicate the connectivity status of the STU. The value connected means that a communications
        path to the VASP (Level 2 communications path) is established for supporting user services. The value
        unconnected means that no communications path to a VASP (Level 2 provider) is currently established."
 ::= { stuStatGroup 2 }

-- The Security Detector Group
--
-- The Security Detector Group is used for capturing information about security threats to the STU.
-- A security detector is a component of a STU that is responsible for detecting security threats on
-- the STU. An example of a security detector is a micro switch that detect unauthorized opening
-- of the STU. Another example may be the violation of a digital seal. When the security detector
-- senses a threat it sets a security flag along with the time the flag was set. The setting of a security
-- flag may disable some functionality of the STU.

-- The security detector helps detect and prevent tampering with the STU. Setting a security flag may
-- disable a STU. The flag may be cleared through management operations.

stuSecurDetectGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 3 }

stuSecurDetectTable OBJECT-TYPE
 SYNTAX SEQUENCE OF StuSecurDetectEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "Table of security detector entries"
 ::= { stuSecurDetectGroup 1 }

stuSecurDetectEntry OBJECT-TYPE
 SYNTAX StuSecurDetectEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "An entry in the security detector table."
 INDEX { stuSecurDetectIndex } ::= { stuSecurDetectTable 1 }

StuSecurDetectEntry ::=
 SEQUENCE {
  stuSecurDetectIndex Integer32,
  stuSecurDetectDesc OCTET STRING (SIZE (0..255)),
  stuSecurFlag TruthValue,
  stuSecurSetTime DateAndTime
 }

stuSecurDetectIndex OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
```

```
            "The value of this object is used as one of the indices for this table. It is a unique identifier for this row
            in the table for this STU. The value of this object may be from 1 to N, where N is the number of
            potential security detectors for the STU."
 ::= { stuSecurDetectEntry 1 }

stuSecurDetectDesc OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
  "A textual description of the security detector."
 ::= { stuSecurDetectEntry 2 }

stuSecurFlag OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
            "The state of the security detector. The value is FALSE when no threat is detected or a detected threat is
            cleared. The value is set to TRUE when a security threat is detected. This flag may be reset to FALSE or
            cleared by the management system."
 ::= { stuSecurDetectEntry 3 }

stuSecurSetTime OBJECT-TYPE
 SYNTAX DateAndTime
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The time that the security flag was last set or cleared."
 ::= { stuSecurDetectEntry 4 }

-- The Software Module Group
--
-- The Software Module Group is used to identify and track the status of the software programs that are
-- currently loaded within the STU. The attributes of this entity represent the program header information
-- associated with each of the loaded application programs. The software module helps track the
-- applications and software capabilities of the STU.

stuSoftwareGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 4 }

stuSoftwareTable OBJECT-TYPE
 SYNTAX SEQUENCE OF StuSoftwareEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "Table of software module entries"
 ::= { stuSoftwareGroup 1 }

stuSoftwareEntry OBJECT-TYPE
 SYNTAX StuSoftwareEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "An entry in the software module table."
 INDEX { stuSoftwareIndex } ::= { stuSoftwareTable 1 }

StuSoftwareEntry ::=
 SEQUENCE {
  stuSoftwareIndex Integer32,
```



```
 stuSwFileName OCTET STRING (SIZE (0..255)),
 stuSwFileType OCTET STRING (SIZE (0..255)),
 stuSwFileSize Integer32,
 stuSwVendorName OCTET STRING (SIZE (0..255)),
 stuSwVersion OCTET STRING (SIZE (0..255)),
 stuSwWorkingMemSize Integer32,
 stuSwLoadDate DateAndTime,
 stuSwLease TimeInterval,
 stuSwDigitalSeal OCTET STRING (SIZE (0..255)),
 stuSwErrorStatus AlarmStatus,
 stuSwErrorCode OCTET STRING (SIZE (0..255)),
 stuSwErrorTimeStamp TimeStamp
 }

stuSoftwareIndex OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The value of this object is used as one of the indices for this table. It is a unique identifier for this row
        in the table for this STU. The value of this object may be from 1 to N, where N is the number of
        software modules for the STU."
 ::= { stuSoftwareEntry 1 }

stuSwFileName OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "A string giving the path and file name of the software module."
 ::= { stuSoftwareEntry 2 }

stuSwFileType OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The type of file of the Software Module."
 ::= { stuSoftwareEntry 3 }

stuSwFileSize OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The actual size of the file."
 ::= { stuSoftwareEntry 4 }

stuSwVendorName OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The name of the vendor who is the manufacturer of the software."
 ::= { stuSoftwareEntry 5 }

stuSwVersion OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
```

```
 STATUS current
 DESCRIPTION
  "The version number of the software module."
 ::= { stuSoftwareEntry 6 }

stuSwWorkingMemSize OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The amount of working RAM required by the software module in execution."
 ::= { stuSoftwareEntry 7 }

stuSwLoadDate OBJECT-TYPE
 SYNTAX DateAndTime
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The date and time the software was loaded into the STU."
 ::= { stuSoftwareEntry 8 }

stuSwLease OBJECT-TYPE
 SYNTAX TimeInterval
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "This is conditional on the support of software leasing functionality. The amount of time the software is
        leased, starting at load date."
 ::= { stuSoftwareEntry 9 }

stuSwDigitalSeal OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "This is conditional on the support of software digital seal. The digital seal for the instance of software
        (e.g., digital signature)."
 ::= { stuSoftwareEntry 10 }

stuSwErrorStatus OBJECT-TYPE
 SYNTAX AlarmStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The current error status of the software: error or cleared"
 ::= { stuSoftwareEntry 11 }

stuSwErrorCode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The most recent software error or status code defined by the software supplier."
 ::= { stuSoftwareEntry 12 }

stuSwErrorTimeStamp OBJECT-TYPE
 SYNTAX TimeStamp
 MAX-ACCESS read-only
 STATUS current
```



```
 DESCRIPTION
         "The time stamp for the most recent software error or status code."
 ::= { stuSoftwareEntry 13 }

-- The Processor Module Group
--
-- A processor module is a functional module within the Set Top Unit (STU) that provides for
-- management capabilities of the processors (e.g., CPU), including detection of hardware failures,
-- processor load measurements, etc. The processor module helps track the status of the
-- processor(s) of the STU.

stuProcessorGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 5 }

stuProcessorTable OBJECT-TYPE
 SYNTAX SEQUENCE OF StuProcessorEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "Table of processor module entries"
 ::= { stuProcessorGroup 1 }

stuProcessorEntry OBJECT-TYPE
 SYNTAX StuProcessorEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "An entry in the processor module table."
 INDEX { stuProcessorIndex } ::= { stuProcessorTable 1 }

StuProcessorEntry ::=
 SEQUENCE {
  stuProcessorIndex Integer32,
  stuProcType OCTET STRING (SIZE (0..255)),
  stuProcFunction OCTET STRING (SIZE (0..255)),
  stuProcAlarmStatus AlarmStatus,
  stuProcErrorCode OCTET STRING (SIZE (0..255)),
  stuProcErrorTimeStamp TimeStamp,
  stuProcDiagTestResult TestStatus,
  stuProcDiagTestTime TimeStamp,

         -- Optional element:
  stuProcDiagTestIndicator TestIndicator
 }

stuProcessorIndex OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
         "The value of this object is used as one of the indices for this table. It is a unique identifier for this row
         in the table for this STU. The value of this object may be from 1 to N, where N is the number of
         processor modules for the STU."
 ::= { stuProcessorEntry 1 }

stuProcType OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
```

```
        "Identifies the processor model, type, version, and revision of the processor module."
 ::= { stuProcessorEntry 2 }

stuProcFunction OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Describes the functionality of the processing unit (graphics unit, video unit, sound unit, etc.)."
 ::= { stuProcessorEntry 3 }

stuProcAlarmStatus OBJECT-TYPE
 SYNTAX AlarmStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Used to indicate the status, i.e., active or cleared, of the processor failure alarm."
 ::= { stuProcessorEntry 4 }

stuProcErrorCode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The most recent processor error or status code defined by the processor module supplier."
 ::= { stuProcessorEntry 5 }

stuProcErrorTimeStamp OBJECT-TYPE
 SYNTAX TimeStamp
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The time stamp for the most recent processor error or status code."
 ::= { stuProcessorEntry 6 }

stuProcDiagTestResult OBJECT-TYPE
 SYNTAX TestStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Used to indicate the result of the last diagnostic test. The value is 'passed', 'failed', or 'aborted'."
 ::= { stuProcessorEntry 7 }

stuProcDiagTestTime OBJECT-TYPE
 SYNTAX TimeStamp
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Used to indicate the end time (or abort time) of the last
  diagnostic test."
 ::= { stuProcessorEntry 8 }

stuProcDiagTestIndicator OBJECT-TYPE
 SYNTAX TestIndicator
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "This is conditional on support of manager initiated diagnostics. Used to indicate, perform, or abort a
        diagnostic test on the entity. The value 'active' means that a diagnostic test is currently running. The
```



```
            value 'inactive' means that no diagnostic test is currently running. Requesting changing the value from
            'inactive' to 'active' is equivalent to requesting performing a diagnostic test on the entity. Requesting
            changing the value from 'active' to 'inactive' is equivalent to requesting aborting an active diagnostic test
            on the entity."
 ::= { stuProcessorEntry 9 }

-- The Memory Module Group
--
-- A memory module is a functional module within the Set Top Unit (STU) that provides for management --
capabilities of the memory, including detection of memory failures, memory utilization measurements,
-- etc. The memory module helps track the status of the memory(s) of the STU.

stuMemoryGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 6 }

stuMemoryTable OBJECT-TYPE
 SYNTAX SEQUENCE OF StuMemoryEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "Table of memory module entries"
 ::= { stuMemoryGroup 1 }

stuMemoryEntry OBJECT-TYPE
 SYNTAX StuMemoryEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "An entry in the memory module table."
 INDEX { stuMemoryIndex } ::= { stuMemoryTable 1 }

StuMemoryEntry ::=
 SEQUENCE {
  stuMemoryIndex Integer32,
  stuMemType OCTET STRING (SIZE (0..255)),
  stuMemSize Integer32,
  stuMemAvailable Integer32,
  stuMemDiagTestResult TestStatus,
  stuMemDiagTestTime TimeStamp,
  stuMemAlarmStatus AlarmStatus,

            -- Optional element:
  stuMemDiagTestIndicator TestIndicator
 }

stuMemoryIndex OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
            "The value of this object is used as one of the indices for this table. It is a unique identifier for this row
            in the table for this STU. The value of this object may be from 1 to N, where N is the number of memory
            modules for the STU."
 ::= { stuMemoryEntry 1 }

stuMemType OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
```

```
            "Describes the type of memory within this memory module. Example include video frame buffer,
            volatile RAM, non-volatile FLASH, etc."
 ::= { stuMemoryEntry 2 }

stuMemSize OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "The Total amount of memory in this memory module."
 ::= { stuMemoryEntry 3 }

stuMemAvailable OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
            "The amount of memory currently available for use in this memory module."
 ::= { stuMemoryEntry 4 }

stuMemDiagTestResult OBJECT-TYPE
 SYNTAX TestStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
            "Used to indicate the result of the last diagnostic test. The value is 'passed', 'failed', or 'aborted'."
 ::= { stuMemoryEntry 5 }

stuMemDiagTestTime OBJECT-TYPE
 SYNTAX TimeStamp
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
            "Used to indicate the end time (or abort time) of the last diagnostic test."
 ::= { stuMemoryEntry 6 }

stuMemAlarmStatus OBJECT-TYPE
 SYNTAX AlarmStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
            "Used to indicate the status, i.e., active or cleared, of the memory failure alarm."
 ::= { stuMemoryEntry 7 }

stuMemDiagTestIndicator OBJECT-TYPE
 SYNTAX TestIndicator
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
            "This is conditional on support of manager initiated diagnostics. Used to indicate, perform, or abort a
            diagnostic test on the entity. The value 'active' means that a diagnostic test is currently running. The
            value 'inactive' means that no diagnostic test is currently running. Requesting changing the value from
            'inactive' to 'active' is equivalent to requesting performing a diagnostic test on the entity. Requesting
            changing the value from 'active' to 'inactive' is equivalent to requesting aborting an active diagnostic test
            on the entity."
 ::= { stuMemoryEntry 8 }

-- The Power Module Group
--
```



```
-- A power module is a functional module within the Set Top Unit (STU) that provides for management
-- capabilities of the power supply, including detection of power failures, etc. The power module helps
-- track the status of the power supply of the STU.

stuPowerGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 7 }

PowerAlarmStatus ::= INTEGER {
 powerSourceFail(1), unitFail(2), degradation(3), normal(4) }

stuPowerTable OBJECT-TYPE
 SYNTAX SEQUENCE OF StuPowerEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "Table of power module entries"
 ::= { stuPowerGroup 1 }

stuPowerEntry OBJECT-TYPE
 SYNTAX StuPowerEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "An entry in the power module table."
 INDEX { stuPowerIndex } ::= { stuPowerTable 1 }

StuPowerEntry ::=
 SEQUENCE {
  stuPowerIndex Integer32,
  stuPowerDegradationAlarmStatus PowerAlarmStatus
 }

stuPowerIndex OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The value of this object is used as one of the indices for this table. It is a unique identifier for this row
        in the table for this STU. The value of this object may be from 1 to N, where N is the number of power
        modules for the STU."
 ::= { stuPowerEntry 1 }

stuPowerDegradationAlarmStatus OBJECT-TYPE
 SYNTAX PowerAlarmStatus
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
      "Used to indicate the status, i.e., power source failure, unit failure, degraded, or normal, of the power
       degradation alarm."
 ::= { stuPowerEntry 2 }

-- The User Device Module Group
--
-- A user device module is a functional module within the Set Top Unit (STU) that provides for
-- management capabilities of the user devices. User devices include I/R receptors, VCR controllers,
-- printers, etc. The User Device module helps track the status of the user device(s) connected to the STU.

stuUserDeviceGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 8 }

stuUserDeviceTable OBJECT-TYPE
```

```
 SYNTAX SEQUENCE OF StuUserDeviceEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "Table of user device module entries"
 ::= { stuUserDeviceGroup 1 }

stuUserDeviceEntry OBJECT-TYPE
 SYNTAX StuUserDeviceEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
  "An entry in the user device module table."
 INDEX { stuUserDeviceIndex }
 ::= { stuUserDeviceTable 1 }

StuUserDeviceEntry ::=
 SEQUENCE {
  stuUserDeviceIndex Integer32,
  stuUserDevType OCTET STRING (SIZE (0..255)),
  stuUserDevDesc OCTET STRING (SIZE (0..255)),
  stuUserDevAlarmStatus AlarmStatus,
  stuUserDevErrorCode OCTET STRING (SIZE (0..255)),
  stuUserDevErrorTimeStamp TimeStamp
 }

stuUserDeviceIndex OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The value of this object is used as one of the indices for this table. It is a unique identifier for this row
        in the table for this STU. The value of this object may be from 1 to N, where N is the number of user
        device modules for the STU."
 ::= { stuUserDeviceEntry 1 }

stuUserDevType OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "Identifies the device model, type, version, and revision of the user device module."
 ::= { stuUserDeviceEntry 2 }

stuUserDevDesc OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
  "Describes the functionality of the user device.
  Examples include: 'IR TRANSMITTER', 'IR RECEIVER',
  'POINTER DEVICE', 'SMART CARD', 'RF DEVICE', 'KEYBOARD',
  'MAGNETIC CARD READER', AND 'SERIAL INTERFACE'."
 ::= { stuUserDeviceEntry 3 }

stuUserDevAlarmStatus OBJECT-TYPE
 SYNTAX AlarmStatus
 MAX-ACCESS read-only
 STATUS current
```



```
 DESCRIPTION
        "Used to indicate the status, i.e., active or cleared, of the user device failure alarm."
 ::= { stuUserDeviceEntry 4 }

stuUserDevErrorCode OBJECT-TYPE
 SYNTAX OCTET STRING (SIZE (0..255))
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The most recent user device error or status code defined by the user device module supplier."
 ::= { stuUserDeviceEntry 5 }

stuUserDevErrorTimeStamp OBJECT-TYPE
 SYNTAX TimeStamp
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
        "The time stamp for the most recent user device error or status code."
 ::= { stuUserDeviceEntry 6 }

-- The STU Notification Group
--
-- Types of STU notifications include: alarm notifications; state change
-- notifications; Video service heartbeat notifications; value change
-- notifications; and security notifications. An alarm notification is
-- sent when one of the following objects change value indicating an
-- alarm: stuProcAlarmStatus, stuMemAlarmStatus,
-- stuPowerDegradationAlarmStatus, stuAnalogTunerAlarmStatus,
-- stuAnalogTunerSignalAlarmStatus, or stuUserDevAlarmStatus.

stuNotificationGroup OBJECT IDENTIFIER ::= { DAVIC STU v1 9 }

stuProcAlarm NOTIFICATION-TYPE
 OBJECTS { stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
   stuProcessorIndex, stuProcAlarmStatus }
 STATUS current
 DESCRIPTION
        "This trap signifies an alarm event related to the processor module indicated by stuProcessorIndex. This
        trap should be sent to the management system when the value of the stuProcAlarmStatus changes."
 ::= { stuNotificationGroup 1 }

stuProcAlarmEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The value is set to TRUE when the processor alarm notification is enabled. It may be set to FALSE to
        disable this notification."
 ::= { stuNotificationGroup 2 }

stuMemAlarm NOTIFICATION-TYPE
 OBJECTS { stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
   stuMemoryIndex, stuMemAlarmStatus }
 STATUS current
 DESCRIPTION
        "This trap signifies an alarm event related to the memory module indicated by stuMemoryIndex. This
        trap should be sent to the management system when the value of the stuMemAlarmStatus changes."
 ::= { stuNotificationGroup 3 }
```

```
stuMemAlarmEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The value is set to TRUE when the memory alarm notification is enabled. It may be set to FALSE to
        disable this notification."
 ::= { stuNotificationGroup 4 }

stuPowerAlarm NOTIFICATION-TYPE
 OBJECTS { stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
   stuPowerIndex, stuPowerDegradationAlarmStatus }
 STATUS current
 DESCRIPTION
        "This trap signifies an alarm event related to the power module indicated by stuPowerIndex. This trap
        should be sent to the management system when the value of the stuPowerDegradationAlarmStatus
        changes."
 ::= { stuNotificationGroup 5 }

stuPowerAlarmEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The value is set to TRUE when the power alarm notification is enabled. It may be set to FALSE to
        disable this notification."
 ::= { stuNotificationGroup 6 }

stuUserDevAlarm NOTIFICATION-TYPE
 OBJECTS { stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
   stuUserDeviceIndex, stuUserDevAlarmStatus }
 STATUS current
 DESCRIPTION
        "This trap signifies an alarm event related to the user device module indicated by stuUserDeviceIndex.
        This trap should be sent to the management system when the value of the stuUserDevAlarmStatus
        changes."
 ::= { stuNotificationGroup 7 }

stuUserDevAlarmEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The value is set to TRUE when the user device alarm notification is enabled. It may be set to FALSE to
        disable this notification."
 ::= { stuNotificationGroup 8 }

-- A security notification is sent when the stuSecurFlag changes value by any means. The trap
-- should include the stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
-- stuSecurDetectIndex, stuSecurFlag, and stuSecurSetTime.

stuSecurity NOTIFICATION-TYPE
 OBJECTS { stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
   stuSecurDetectIndex, stuSecurFlag, stuSecurSetTime }
 STATUS current
 DESCRIPTION
        "This trap signifies a security event at the Set Top. This trap should be sent to the management system
        when the value of stuSecurFlag changes by any means."
 ::= { stuNotificationGroup 9 }
```



```
-- A state change notification is sent when one of the following objects changes value indicating a change
-- in state: stuPowerOnSelfTestStatus, stuStatL1ConnectStatus, stuStatL2ConnectStatus, or
-- stuSwProceduralStatus.

stuPOSTStateChange NOTIFICATION-TYPE
 OBJECTS { stuManufacturerOUICode, stuModelNumberCode, stuSerialNumberCode,
   stuPowerOnSelfTestStatus }
 STATUS current
 DESCRIPTION
        "This trap signifies a state change event related to the STU. This trap should be sent to the management
        system when the value of stuPowerOnSelfTestStatus changes."
 ::= { stuNotificationGroup 10 }

stuPOSTStateEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
        "The value is set to TRUE when the POST state change notification is enabled. It may be set to FALSE
        to disable this notification."
 ::= { stuNotificationGroup 11 }

END
```

Annex B
(normative)
Server Management Information Base

This Annex defines an SNMP MIB for management of a DAVIC compliant Server Provider System. It enables management of the base resources and monitoring of the current use of the server system.

B.1 Overview

The DAVIC SERVER MIB consists of several groups. The managed objects contained in these groups are variously distributed across a Service Provider System. They provide both monitoring and control of a SPS. They do not provide any monitoring of the underlying hardware. This is the realm of other MIBs including MIB-II (RFC1213) and HOST MIB (RFC1514). Some of the groups in the DAVIC server MIB are only instantiated once for a Server Provider System. Other groups may have multiple instantiations in a distributed implementation of the SPS. This introduction identifies which groups may have multiple instances as does the MIB definition.

B.1.1 Single Instance Groups

The following MIB groups will only have a single instance in a SPS and will typically be implemented by an SNMP agent running on the same system that supports the Service Gateway functionality

B.1.1.1 General Group

The spsGeneral Group lists the general state and history of the SPS instance. There is one instance of this group in an SPS instance. This includes a count of streams accepted and rejected as well as the maximum number of sessions and stream. The spsGeneralStateOfHealthChanged TRAP will be generated when there is a change in the state of the SPS instance.

The spsGeneral group also includes the spsHostsTable and spsInterfacesTable. These tables record the hosts which form a distributed SPS and the external interfaces to the SPS instance respectively. These allow system managers to find the individual SNMP agents on each host. The group also contains several TRAPs which are generated when the disk or memory capacity on an individual host is exceeded.

Note: For a tightly coupled distributed SPS implementation, it may not be sensible to have an SNMP agent on each processor. Such systems may implement only a single SNMP agent in which case, the spsHostsTable will only contain a single line.

B.1.1.2 Thresholds Group

This group is for controlling the threshold at which a TRAP will be generated. The group also defines TRAPs for thresholds which have been exceeded. There is only one instance of this group in the SPS instance.

B.1.1.3 Elements Group

This group lists the Service elements which have registered with the Service Gateway of this SPS instance. Each row in the spsElementTable represents a service element. There is only one instance of this group in this SPS instance. This group also contains the spsContentProviderTable which lists the content providers in this SPS and their maximum resource usage.

B.1.1.4 Service Gateway Group

This group represents the service gateway service element. It records the number of current sessions and number of sessions which have been accepted and rejected. For each current session, there is an entry in the spsSessionTable. This records the start time, current state and client(s) of the session. It also permits the manager

to terminate a session by setting spsSessionRowStatus to 'destroy'. There are two TRAPs which are generated when there is a change in status. of a session.

spsSessionTrapGeneration object identifier allows the manager to control when a TRAP is generated following a change in status of a session. This will default to only sending a TRAP when session is rejected or aborted. There will be as many Service Gateway groups as where are service gateway service elements in the SPS instance.

B.1.1.5 Content Provider Group

This group defines the content provider domains which are part of the server provider system. The spsContentProviderTable records the principle identifier of the content provider, the providers, name and the disk capacity and bandwidth allocated to this provider.

B.1.1.6 Active User Group

This group contains the spsActiveUserTable which identifies the current set of users of the Server Provider System. This table links the session identifier with the principle identifier of the user of the session. The DAVIC Server MIB does not contain any other information about the users.

B.1.2 Optionally Multiple instantiated MIB Groups

The following groups may be multiply instantiated in a Server Provider System. It is implementation choice over the number of instances of these tables and hence their corresponding SNMP sub agents.

B.1.2.1 Streams Element Group

For each host in a distributed SPS instance, there will be an instance of this group. It represents the set of stream service elements and stream service element instances on the host. This follows the DSM-CC and OMG concepts of object creation for service provision.

spsStreamSETable represents the set of stream service element. Each entry records properties of the stream service element including number of current instances of this stream service element and counters recording its use.

The spsStreamInstTable records details of individual streams being played from this SPS instance on this host. Each row represents a single stream instance and records when the stream started, stream type, the content being used, bandwidth and position in content, external interface being used etc.

There is also a TRAP spsStreamChangedStatus which will be generated when there is a change in the status of a stream instance. The generation of this TRAP is controlled by spsStreamSETrapGeneration variable and defaults to 'reject'.

B.1.2.2 Application Group

This group represents generic application service elements. The spsApplTable identifies the set of applications available on this host. Each row represents an application service element. It contains the application type, counters of use, and status. It also contains spsApplSpecific, an object identifier of a MIB specific to this application service element type.

The spsApplInstTable lists the active applications. Each row represents an application instance. This generic MIB only identifies the session identifier and when the application started. More details of this application instance may be found from the application specific MIB.

B.1.2.3 Content Group

This group represents the content service elements. On every host which supports at least one content service element, there is an spsContentSETable. Each row in this table represents a content service element. This records the content provider name and gauges of content open, on-line and off-line. This table is indexed by the OMG IOR of the content service provider element.

The spsContentInstTable is a table of the contents in this content service element. It records the type, size and usage of the content etc. It is also indexed by the content service element as well as an instance identifier for the content. This allows searching on the content based on content service element.

B.2. SNMP MIB Specification

```
DAVIC-SERVER-MIB DEFINITIONS ::= BEGIN

IMPORTS enterprises, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks
                FROM RFC1155-SMI
        OBJECT-TYPE FROM RFC1212
        DisplayString, PhysAddress FROM RFC1213
        TRAP-TYPE FROM RFC1215;

davic   OBJECT IDENTIFIER ::= { enterprise 1493 }
davicServer     OBJECT IDENTIFIER ::= { davic 2 }
davicServerV1   OBJECT IDENTIFIER ::= { davicServer 1 }
spsGeneral      OBJECT IDENTIFIER ::= { davicServerV1 1 }
spsThresholds   OBJECT IDENTIFIER ::= { davicServerV1 2 }
spsElements     OBJECT IDENTIFIER ::= { davicServerV1 3 }
spsServiceGateway       OBJECT IDENTIFIER ::= { davicServerV1 4 }
spsActiveGroup  OBJECT IDENTIFIER ::= { davicServerV1 5 }
spsStreamsElement       OBJECT IDENTIFIER ::= { davicServerV1 6 }
spsApplication  OBJECT IDENTIFIER ::= { davicServerV1 7 }
spsContent      OBJECT IDENTIFIER ::= { davicServerV1 8 }
--
--
--  edited to remove Bandwidth from Stream
--  Also removal of the userProfile group,
--          inclusion of the contentDomainTable
--          changing the specification of bandwidth
--   elementTable,
--   last n errors in application and stream table
--          simplification of content type and size
--
--
-- GROUP spsGeneral
--
--  This group lists the general state and history of the SPS instance.
--  This group typically runs on the same agent as the service gateway.
--  This group is mandatory
--
spsGeneralStateOfHealth  OBJECT-TYPE
        SYNTAX          INTEGER
                        {
                        okay    (1),
                        warning (2),
                        minor   (3),
                        major   (4),
                        failed  (5)
                        }
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " state of health of the complete server"
        ::= { spsGeneral 1 }
spsGeneralServerId      OBJECT-TYPE
        SYNTAX          DisplayString
```



```
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " Description of the SPS instance"
        ::= { spsGeneral 2 }
spsGeneralDownTime      OBJECT-TYPE
        SYNTAX          DateAndTime
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
        " Date and time the SPS instances was last stopped"
        ::= { spsGeneral 3 }
spsGeneralUpTime        OBJECT-TYPE
        SYNTAX          DateAndTime
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " Date and time the SPS instance was started"
        ::= { spsGeneral 4 }
spsGeneralMaxExclUNSess         OBJECT-TYPE
        SYNTAX          INTEGER
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
        " Max no of exclusive U-N sessions"
        ::= { spsGeneral 5 }
spsGeneralMaxCFUNSessions       OBJECT-TYPE
        SYNTAX          INTEGER
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
        " Max no. of continuous feed U-N sessions"
        ::= { spsGeneral 6 }
spsGeneralMaxStreams    OBJECT-TYPE
        SYNTAX          INTEGER
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
        " Max number of simultaneous streams"
        ::= { spsGeneral 7 }
spsGeneralStreamsAccepted       OBJECT-TYPE
        SYNTAX          Counter32
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
        " Number of streams accepted with this SPS. "
        ::= { spsGeneral 8 }
spsGeneralStreamsRejected       OBJECT-TYPE
        SYNTAX          Counter32
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
        " Number of streams rejected by this SPS"
        ::= { spsGeneral 9 }
spsGeneralStreamsAborted        OBJECT-TYPE
        SYNTAX          Counter32
        ACCESS          read-only
        STATUS          optional
        DESCRIPTION
```

```
            " Number of streams aborted by this SPS"
            ::= { spsGeneral 10 }
spsGeneralApplicationStarted        OBJECT-TYPE
            SYNTAX          Counter32
            ACCESS          read-only
            STATUS          optional
            DESCRIPTION
            " Number of applications started by this SPS"
            ::= { spsGeneral 11 }
spsGeneralApplicationRejected       OBJECT-TYPE
            SYNTAX          Counter32
            ACCESS          read-only
            STATUS          optional
            DESCRIPTION
            " Number of application requested which have been rejected by this
             SPS"
            ::= { spsGeneral 12 }
spsGeneralApplicationAborted        OBJECT-TYPE
            SYNTAX          Counter32
            ACCESS          read-only
            STATUS          optional
            DESCRIPTION
            " Number of applications which were aborted."
            ::= { spsGeneral 13 }
spsGeneralStateOfHealthChanged
            TRAP-TYPE
            ENTERPRISE { davicServerV1 }
            VARIABLES
               {
               spsGeneralStateOfHealth
               }
            DESCRIPTION
            " Trap issued when there is a change in state of health of the SPS
             instance. The new value is sent in the varbindings"
            ::= 1
--
--  The hosts table is a list of the hosts which form the SPS
--
spsHostsNumHosts            OBJECT-TYPE
            SYNTAX          Gauge32
            ACCESS          read-only
            STATUS          mandatory
            DESCRIPTION
            " Number of current hosts in this SPS instance"
            ::= { spsGeneral 14 }
spsHostsTable    OBJECT-TYPE
            SYNTAX          SEQUENCE OF SpsHostsEntry
            ACCESS          not-accessible
            STATUS          mandatory
            DESCRIPTION
            " List of hosts in this SPS instance "
            ::= { spsGeneral 15 }
spsHostsEntry    OBJECT-TYPE
            SYNTAX          SpsHostsEntry
            ACCESS          not-accessible
            STATUS          mandatory
            DESCRIPTION
            " Entry for table spsHostsTable"
            INDEX { spsHostsIndex              }
```



```
        ::= { spsHostsTable 1 }
SpsHostsEntry ::= SEQUENCE
        {
        spsHostsIndex     INTEGER,
        spsHostsName      DisplayString,
        spsHostsIpAddress         IpAddress,
        spsHostsUpTime DateAndTime,
        spsHostsThresholdMaxStreams       INTEGER,
        spsHostsThresholdMaxContentOpen           INTEGER,
        spsHostsThresholdMemory           INTEGER,
        spsHostsThresholdDiskCapacity     INTEGER,
        spsHostsThresholdBandwidth        INTEGER,
        spsHostsMaxBandwidth    INTEGER,
        spsHostsDownTime          DateAndTime,
        spsHostsStatus    INTEGER
        }
spsHostsIndex      OBJECT-TYPE
        SYNTAX          INTEGER
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " Unique identifier in the scope of the SPS for this host entry "
        ::= { spsHostsEntry 1 }
spsHostsName      OBJECT-TYPE
        SYNTAX          DisplayString
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " Host name of the host "
        ::= { spsHostsEntry 2 }
spsHostsIpAddress         OBJECT-TYPE
        SYNTAX          IpAddress
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " IpAddress of the host. Note both name and IP address are included
         for those systems which do not support any name resolution."
        ::= { spsHostsEntry 3 }
spsHostsUpTime  OBJECT-TYPE
        SYNTAX          DateAndTime
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
        " Time this host started participating in this SPS system"
        ::= { spsHostsEntry 4 }
spsHostsThresholdMaxStreams       OBJECT-TYPE
        SYNTAX          INTEGER
        ACCESS          read-write
        STATUS          mandatory
        DESCRIPTION
        " percentage of loading of streams to issue the maxStreamsExceeded
         TRAP. The TRAP is reissued whenever a new stream is requested and
         the threshold is still exceeded"
        ::= { spsHostsEntry 5 }
spsHostsThresholdMaxContentOpen           OBJECT-TYPE
        SYNTAX          INTEGER
        ACCESS          read-write
        STATUS          mandatory
        DESCRIPTION
```

```
            " percentage of loading of open content to issue the
             maxOpenContentExceeded TRAP. The TRAP is reissued whenever a
             content is requested and the threshold is still exceeded"
            ::= { spsHostsEntry 6 }
spsHostsThresholdMemory          OBJECT-TYPE
            SYNTAX          INTEGER
            ACCESS          read-write
            STATUS          mandatory
            DESCRIPTION
            " percentage of loading of memory before issuing the
             maxMemoryExceeded TRAP"
            ::= { spsHostsEntry 7 }
spsHostsThresholdDiskCapacity    OBJECT-TYPE
            SYNTAX          INTEGER
            ACCESS          read-write
            STATUS          mandatory
            DESCRIPTION
            " percentage of disk capacity used before issuing the
             maxDiskSpaceExceeded TRAP"
            ::= { spsHostsEntry 8 }
spsHostsThresholdBandwidth       OBJECT-TYPE
            SYNTAX          INTEGER
            ACCESS          read-write
            STATUS          mandatory
            DESCRIPTION
            " percentage of disk bandwidth used before issuing the
             maxDiskBandwidthExceeded TRAP"
            ::= { spsHostsEntry 9 }
spsHostsMaxBandwidth    OBJECT-TYPE
            SYNTAX          INTEGER
            ACCESS          read-write
            STATUS          mandatory
            DESCRIPTION
            " Maximum bandwidth possible from this host measured in kbps."
            ::= { spsHostsEntry 10 }
spsHostsDownTime         OBJECT-TYPE
            SYNTAX          DateAndTime
            ACCESS          read-only
            STATUS          mandatory
            DESCRIPTION
            " Time this host stopped participating in this SPS system"
            ::= { spsHostsEntry 11 }
spsHostsStatus     OBJECT-TYPE
            SYNTAX          INTEGER
                            {
                            unknown         (1),
                            starting  (2),
                            okay      (3),
                            problem (4),
                            stopping (5),
                            stopped  (6)
                            }
            ACCESS          read-write
            STATUS          mandatory
            DESCRIPTION
            " State of this host participating in the SPS. This may be used to
             control the state of the host by setting the state to 'stopped' or
             'okay'. "
            ::= { spsHostsEntry 12 }
```