TA Document 1999035

AV/C Tuner Subunit Model and Command Set version 2.0

October 24, 2000

Sponsored by:
1394 Trade Association

Accepted for Release by:
1394 Trade Association Board of Directors.

Abstract:
This specification defines a model and command set for analog and digital tuners operating over IEEE1394-1995. The command set makes use of the Function Control Protocol (FCP) defined by IEC61883, Consumer audio/video equipment - Digital Interface standard, for the transport of audio/video command requests and responses. The audio/video devices are implemented as a common unit architecture within 1394-1995.

Keywords:
Audio, Video, 1394, Digital, Interface, Tuner.

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in working groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
Regency Plaza Suite 350
2350 Mission College Blvd.
Santa Clara, Calif. 95054, USA

1394 Trade Association Specifications are adopted by the 1394 Trade Association without regard to patents which may exist on articles, materials or processes or to other proprietary intellectual property which may exist within a specification. Adoption of a specification by the 1394 Trade Association does not assume any liability to any patent owner or any obligation whatsoever to those parties who rely on the specification documents. Readers of this document are advised to make an independent determination regarding the existence of intellectual property rights, which may be infringed by conformance to this specification.

Table of contents

List of figures

List of tables

1. Overview

1.1 Purpose

This document defines a model and command set for AV/C Tuner Subunits. The model makes full use of the AV/C Descriptor Mechanism, which is defined in reference [R3] below.

The AV/C Tuner model supports a variety of different broadcast systems, including both analog and digital. The model allows the implementation of an AV/C Tuner Subunit supporting several of these broadcast systems.

1.2 Scope

This document defines the subunit dependent information, descriptors, and commands of the AV/C Tuner Subunit with respect to the AV/C Descriptor Mechanism that applies to all Tuners. Information that is specific to a broadcast system is not defined in this specification.

2. References

The following standards contain provisions, which through reference in this document, constitute provisions of this standard. All the standards listed are normative references. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[R1] IEEE Std 1394-1995, Standard for a High Performance Serial Bus.

[R2] IEC 61883-1, Consumer audio/video equipment – Digital interface – Part 1: General.

[R3] 1394 TA-1998003, AV/C Digital Interface Command Set General Specification, Version 3.0.

[R4] 1394 TA-1998010, Enhancement to the AV/C General Specification 3.0, Version 1.0

[R5] 1394 TA-1998004, AV/C Tuner Model and Command Set, Version 1.0

[R6] EN300 468 V1.3.1 (1997-09) Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB Systems

3. Definitions

3.1 Conformance levels

3.1.1 expected: A key word used to describe the behavior of the hardware or software in the design models *assumed* by this Specification. Other hardware and software design models may also be implemented.

3.1.2 may: A key word that indicates flexibility of choice with *no implied preference*.

3.1.3 shall: A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory requirements.

3.1.4 should: A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

3.1.5 reserved fields: A set of bits within a data structure that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall zero these fields. Future revisions of this specification, however, may define their usage.

3.1.6 reserved values: A set of values for a field that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall not generate these values for the field. Future revisions of this specification, however, may define their usage.

NOTE —The IEEE is investigating whether the “may, shall, should” and possibly “expected” terms will be formally defined by IEEE. If and when this occurs, draft editors should obtain their conformance definitions from the latest IEEE style document.

3.2 Glossary of terms

3.2.1 Available Information Instances: All the Information Instances contained in an input signal from the antenna destination plug or the demux destination plug.

3.2.2 Bouquet: A collection of Services marketed as a single entity. These Services may span more than one Multiplex.

3.2.3 byte: Eight bits of data, used as a synonym for octet.

3.2.4 Component: One or more entities, which together make up an event, e.g. video, audio, or tele-text.

3.2.5 CSR Architecture: A convenient abbreviation of the following reference (see clause 2): ISO/IEC 13213 : 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information Technology—Microprocessor systems—Control and Status Register (CSR) Architecture for Microcomputer Buses.

3.2.6 Currently Selected Information Instances: The Information Instances that have been selected and made available on the tuner’s source plugs.

3.2.7 Data: The legal and navigational entities that are included in the transmission. It is not the same as “Data Component”.

3.2.8 Demultiplexing: A process or technique of unpacking a single packed stream back to multiple streams of signals. This is the reverse process of Multiplexing.

3.2.9 Information Instance: An actual piece of information in a Multiplex, including the Multiplex itself. It is one of the defined Information Types (Multiplex, Service, Component, or data).

3.2.10 Information Type: A Multiplex, a Service, a Component, or a Data.

3.2.11 Multiplex: A stream of all the digital data carrying one or more Services and data within a single physical transponder or channel.

3.2.12 Multiplexing: A process or technique of packing multiple streams of signals into a single stream for efficiency and cost purposes. Examples are time division multiplexing or frequency division multiplexing.

3.2.13 Multiplex Selection: The selection of a complete Multiplex stream without Demultiplexing. This is a special “Currently Selected Information Instances”. When this happens, the demuxer is available to perform its function as a separate operation.

3.2.14 Network: The vehicle of delivering the signals to the receiver. There are three types of networks defined: Satellites, Cable, and Terrestrial.

3.2.15 quadlet: Four bytes of data.

3.2.16 Service: A sequence of programs under the control of a broadcaster, commonly called a service provider, which can be broadcasted as part of a schedule.

3.2.17 Transponder: A physical device capable of transmitting a stream of signals called a Multiplex.

3.2.18 Tuner: A physical or logical entity that can capture the broadcast signals and feed them to other parts in the unit for further processing.

3.3 Acronyms and abbreviations

AV/C Audio Video Control

DVB Digital Video Broadcasting

FCP Function Control Protocol

SID Subunit Identifier Descriptor

4. Tuner Subunit Model

4.1 Broadcasting concepts and Service Delivery Model

The following diagram and definitions illustrate the Digital Video Broadcasting (DVB) service delivery model as defined in reference [R6]: **European Telecommunication Standard prETS 300 468**. While not all broadcast systems will follow this exact model, it serves as a good reference for the types of data that the AV/C tuner model will be dealing with, and it helps to establish a common language for many parts of the broadcast process:

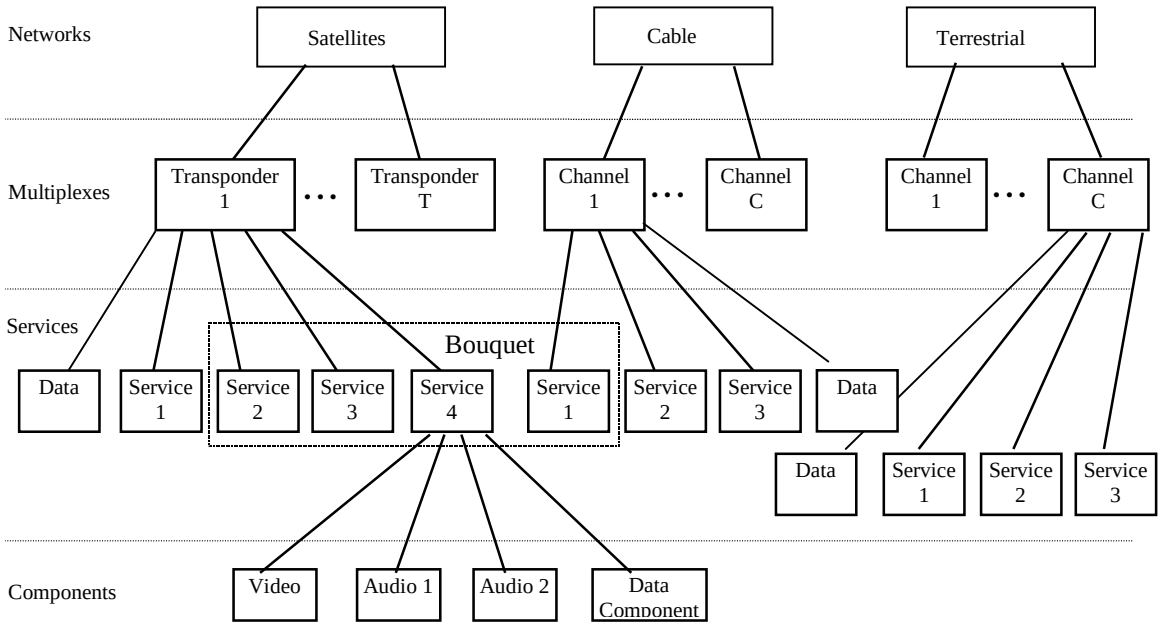


Figure 4.1 – DVB Broadcasting Delivery Model

Note: When a Service is provided by the tuner, the minimum necessary navigation data conformant to the requirements of the service provider shall be included when they are vital to the device receiving the data. For example, in the case of a DVB service being sent to a recording device for later playback, legal Program Association Table (PAT) and Program Map Table (PMT) must be sent, in addition to the service. For details, please refer to the **Guidelines on Implementation and Usage of Service Information**, which is part of the document of Reference [R6].

Note: Bouquet is not used in the AV/C Tuner Subunit Model.

The hierarchical relationships of Networks, Multiplexes, Services, and Components described above form the foundation of the AV/C Tuner Descriptor Mechanism, which is based on the AV/C Descriptor Mechanism described in reference [R3] and [R4].

The analog broadcasting model would be slightly different. An analog transponder has only one single Service. Hence, an analog “Multiplex” consists of a single analog Service. An analog Service may have several Components, similar to the digital Service.

4.2 AV/C Tuner Subunit Conceptual Model

The main purpose of the Tuner Subunit is to allow a controller to select a Service.

As illustrated in the following figure, the conceptual model of a Tuner Subunit consists of two functional stages: the tuning and the demultiplexer (also called demuxer or demux), two destination plugs: antenna destination plug and demux destination plug, and one or more source plugs.

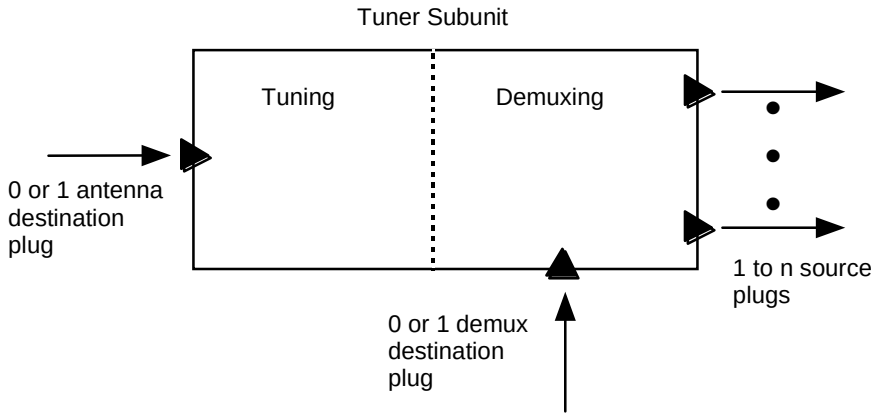


Figure 4.2 – AV/C Tuner Subunit Conceptual Model

Input to the antenna destination plug is from the external antenna input plugs, which is one type of external input plugs of the unit. The formats of the input streams are specified by the corresponding broadcast systems.

The Tuning function and the antenna destination plug form an integral part of a Tuner Subunit. A Tuner Subunit can not have a Tuning function without the antenna destination plug, or vice versa. Input to the tuning function is via a single antenna destination plug. Output of the tuning function is implicitly passed to the demuxing function.

Input to the demux destination plug is from the serial bus input plug (digital tuner), the external input plug (analog tuner), or a source plug of another subunit in the same unit (both digital and analog tuner).

Input to the demuxing function is either from the tuning function, which is implicitly connected to the demuxing function, or from a single demux destination plug. Output of the demuxing function goes to the source plugs.

Inputs to the source plugs are from the demuxing function. Output streams on the source plugs are transport streams of the corresponding broadcast systems.

For discussion purposes, this document will refer to any unit (device) which contains a Tuner Subunit as a “receiver”. Of course, it is possible to have a Tuner Subunit in many different types of devices, such as a television (TV), a set top box (STB), or a video cassette recorder (VCR), etc.

A receiver unit has two types of input plugs: external input plugs and serial bus (1394) input plugs, and two types of output plugs: external output plugs and serial bus output plugs.

The external input plug can either be an external antenna input plug, or some other types of external input plug.

A Tuner Subunit is allowed to support input from *only* the demux destination plug.

If the Tuner Subunit is performing a Multiplex selection operation which does not require the use of the demux, then it may also allow a controller to explicitly use the demuxing function as a separate operation, as described in Annex A.3 on page 88.

4.2.1 Tuning Function

Tuning is the process of matching the physical carrier frequency of the transponder or channel, capturing and decoding the signals. Each transponder or channel uses a specific carrier frequency specified by the broadcasting system service provider.

A Tuner Subunit can be designed to handle more than one physical carrier frequency, but it can only handle one carrier frequency at a time. If a receiver unit is designed to handle more than one frequency at a time, it shall be modeled as having separate Tuner Subunits, one for each frequency that can be handled simultaneously.

A Tuner Subunit can be designed to handle more than one type of signal (**analog video, DVB**, etc.), but it can only handle one type of signal at a time. If a receiver unit is designed to handle more than one type of signal at a time, it shall be modeled as having separate Tuner Subunits, one for each signal that can be handled simultaneously.

It is possible to have more Tuner Subunits than there are physical connectors.

4.2.2 Demuxing Function

Demuxing, or demultiplexing, is the reverse process of multiplexing. While multiplexing packs one or more streams for transmission for efficiency and cost purposes, demuxing unpacks the received signals and returns them back to their original streams, which are the Services within the Multiplex.

Since an analog “Multiplex” consists of a single analog Service, the demuxing function of an analog tuner does not really demux the analog signals. However, the demuxing function may provide the function of selecting the Components within the Service, such as closed-captioned text, secondary audio channel, etc.

4.2.3 Destination Plugs

There are two types of destination plugs in a Tuner Subunit: the antenna destination plug, and the demux destination plug.

For the purpose of identifying the antenna or demux destination plugs in the parameters of the CONNECT command, the following values are defined for these plugs:

Table 4.1 – Destination Plug Number Assignment

Destination Plug	Plug Number Value
Antenna Destination Plug	0
Demux Destination Plug	1

Notice that even if the antenna destination plug is not implemented by a tuner subunit, the demux destination plug uses the plug number of 1, as if the antenna destination plug is implemented.

4.2.3.1 Antenna Destination Plug

Input to the tuner antenna destination plug is from an antenna, which is normally attached to an external antenna input plug on the receiver unit. Note that it does not make sense to connect a serial bus input plug of the receiver to the tuner antenna destination plug, because the tuner antenna destination plug receives its input from antennas. Any attempt to connect a serial bus input plug to the tuner antenna destination plug shall be returned with the response code NOT IMPLEMENTED.

If a service selection command is issued to the Tuner Subunit, which requires the connection of the antenna destination plug to a different external antenna input plug, then the connection will **automatically** be established, overriding a previously established connection which may have been made by a controller with the CONNECT command.

If the Tuner Subunit has an existing connection which has been *locked*, then a subsequent service selection command which conflicts with that connection shall generate a response of REJECTED. If the connection is *permanent*, then the conflicting command shall generate a response of NOT IMPLEMENTED.

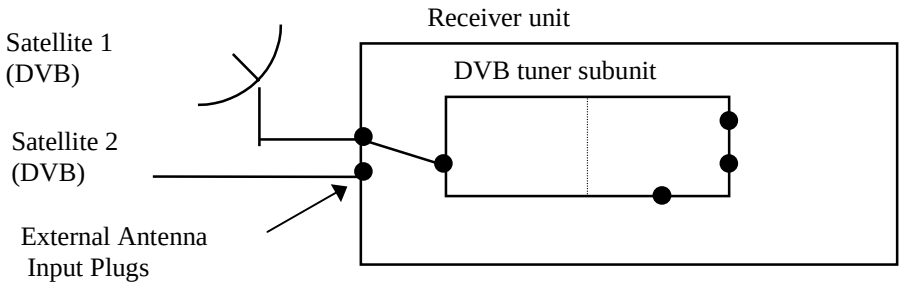


Figure 4.3 – Multiple Antennas and External Antenna Input Plugs

In the above figure, there are two external antenna input plugs that provide DVB signals, but there is only one Tuner Subunit that can handle DVB. To select a DVB service, it is necessary to establish a connection between one of these two external antenna input plugs and the antenna destination plug of the Tuner Subunit. It should be clear that there can be only one connection at a time.

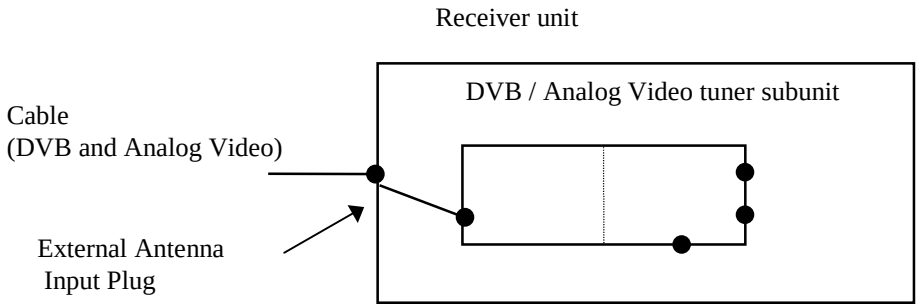


Figure 4.4 – Single Antenna and External Antenna Input Plug

In the above figure, there is a single external antenna input plug which provides both a DVB signal and an Analog Video signal, and the Tuner Subunit is able to handle both types of signals. This connection can be permanent since there is only one external input plug that carries the signals. However, the individual plugs (external input, subunit antenna destination) must still be reported via the appropriate PLUG INFO command.

4.2.3.2 Demux Destination Plug

The demux destination plug of the digital Tuner Subunit shall be connected to the serial bus input plug of the receiver unit while demuxing a multiplexed stream over 1394. This connection is for digital tuners only.

The demux destination plug may be connected internally to the source plug of another subunit within the same receiver unit. This connection may be permanent.

An analog tuner demux destination plug may be connected to a non-antenna external input plug. This connection may be permanent. This connection is for analog tuners only.

4.2.4 Source Plugs

A Tuner Subunit has one or more source plugs. Outputs of the Tuner Subunit are sent to the source plugs. All output streams on the source plugs are from the same Multiplex, except in the case of a complete Multiplex Selection, where the complete Multiplex is on one single source plug, and the services on other source plugs are from the Multiplex on the demux destination plug.

Source plugs are usually connected to the receiver unit's external output plugs, serial bus output plugs, or the destination plugs of a subunit within the same receiver unit.

4.2.5 An Example of a Receiver with a Tuner Subunit

The following diagram illustrates the relationships between the receiver unit's serial bus input / output plugs, external input / output plugs, and the Tuner Subunit's antenna / demux destination plugs and source plugs:

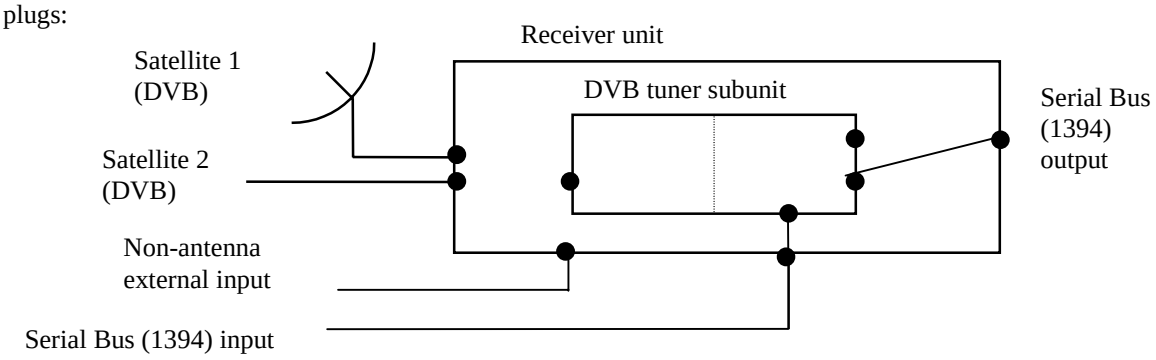


Figure 4.5 – Relationships of Unit / Subunit Plugs

4.2.6 Connections

All current connections of Tuner Subunits shall be reported by the connection status commands. This includes all permanent connections.

All invalid connections shall generate a response of NOT IMPLEMENTED. Unless otherwise noted, all descriptions apply to both analog and digital Tuner Subunits.

A mere connection does not make or break a selection. It only makes the information available to be selected. For example, while the Tuner Subunit is tuning at a Multiplex with some Services selections, a connection of the demux destination plug with a serial bus input plug does not break the connection and selection of the antenna destination plug. The selection is done using the service selection command.

Although the definition of external antenna input plug is not within the scope of Tuner Subunit model, it is possible that an external antenna input plug may be connected to no antenna, to multiple antennas, or to an antenna that carries multiple broadcast systems. The Tuner Subunit model allows all those configurations.

4.2.6.1 Connections to the antenna destination plug

The antenna destination plug may be in a "not connected" state, for example, immediately after power on, or after a DISCONNECT command. The subunit may optionally implement a default connection between one of the external antenna input plugs and the antenna destination plug to improve the user experience.

While using tuner commands to select information instances from the antenna, the connection of the receiver unit's external antenna input plug and the Tuner Subunit's antenna destination plug shall be made **automatically** and **transparently** as part of the command execution.

The receiver unit may optionally allow controllers to explicitly connect an external antenna input plug of the receiver unit to the antenna destination plug of the Tuner Subunit using the connection command.

Table 4.2 – Connections to the Antenna Destination Plug

Non-Tuner Subunit Plug	Tuner Subunit Antenna Destination Plug	Connection Valid?	Comments
External antenna input plug	Antenna destination plug	Yes	Made automatically as part of a tuner command execution.
External (non-antenna) input plug such as video input line	Antenna destination plug	No	X
External output plug	Antenna destination plug	No	X
Serial bus input plug	Antenna destination plug	No	X
Serial bus output plug	Antenna destination plug	No	X
Another Subunit source plug	Antenna destination plug	No	X
Another Subunit destination plug	Antenna destination plug	No	X

4.2.6.2 Connections to the demux destination plug

If it is necessary to change a connection between the Tuner Subunit's demux destination plug and the receiver's external (non-antenna) input plug, serial bus input plug, or the source plug of another subunit, then this connection SHALL be established using a connection command, before issuing the corresponding service selection command.

If the connection is not established when the service selection command is issued, the command will be REJECTED.

Table 4.3 – Connections to the Demux Destination Plug

Non-Tuner Subunit Plug	Tuner Subunit Demux Destination Plug	Connection Valid?	Comments
External antenna input plug	Demux destination plug	No	X
External (non-antenna) input plug such as video input line	Demux destination plug	Yes	This connection is NOT made automatically as part of a service selection command - it must be created using a connection command, or it may be a permanent connection. This connection is for analog tuners.
External output plug	Demux destination plug	No	X
Serial bus input plug	Demux destination plug	Yes	This connection is NOT made automatically as part of a service selection command - it must be created using a connection command, or it may be a permanent connection. This connection is for digital tuners.
Serial bus output plug	Demux destination plug	No	X
Another Subunit source plug	Demux destination plug	Yes	This connection is NOT made automatically as part of a service selection command - it must be created using a connection command, or it may be a permanent connection. Valid for both digital and analog tuners.
Another Subunit destination plug	Demux destination plug	No	X

4.2.6.3 Connections of the source plug

Use the CONNECT command to establish the connection of the source plugs of the Tuner Subunit to the destination plugs of another subunit within the receiver unit, the serial bus output plugs of the receiver unit, or the external output plugs of the receiver unit. There is no automatic connection defined as part of the service selection function.

Use of the CONNECT command is not required if the connection between the Tuner Subunit source plug and a receiver unit's serial bus output plug is permanent. A controller can determine if a connection is permanent by examining the "perm" flag of the responses for the CONNECT status and CONNECTIONS status commands, which are issued to the receiver unit.

Table 4.4 – Connections of the Source Plug

Non-Tuner Subunit Plug	Tuner Subunit Source Plug	Connection Valid?	Comments
External antenna input plug	Source (output) plug	No	X
External (non-antenna) input plug such as video input line	Source (output) plug	No	X
External output plug	Source (output) plug	Yes	This connection is NOT made automatically as part of a service selection command - it must be created using a connection command, or it may be a permanent connection.
Serial bus input plug	Source (output) plug	No	X
Serial bus output plug	Source (output) plug	Yes	This connection is NOT made automatically as part of a service selection command - it must be created using a connection command, or it may be a permanent connection.
Subunit source plug	Source (output) plug	No	X
Subunit destination plug	Source (output) plug	Yes	This connection is NOT made automatically as part of a service selection command - it must be created using a connection command, or it may be a permanent connection. Valid for both digital and analog tuners.

4.3 Tuner Subunit Information Model

4.3.1 Tuner Information Types

There are four possible types of information (Information Type) that can be individually controlled within a Tuner Subunit: Multiplex, Service, Component, or Data.

The tuner sends the physical waveform (Output Signal) to the source plugs. An Output Signal either contains one or more Information Instances, or is identical to the complete contents of an Input Signal. The Output Signal may be a mixture of different Information Types.

4.3.1.1 Multiplex

A *Multiplex* is a stream of all the digital information carrying one or more Services and data within a single physical transponder or channel. It is the physical waveform that contains one or more Services, Components, or data.

The external antenna input plug carries a “set” of Multiplexes from which **only one can be selected (or tuned) at any time**. This means that the Tuner Subunit is always either selecting no Multiplex at all, or exactly one Multiplex at the antenna destination plug.

The input at a demux destination plug is always zero or one Multiplex. When the demux destination plug is not connected, there is zero Multiplex available. When the demux destination plug is connected to either a serial bus input plug or a source plug of another subunit within the same unit, there can be one Multiplex available only.

4.3.1.2 Service

A *Service* is a sequence of programs under the control of a broadcaster that can be broadcasted as part of a schedule. For illustration purposes, we consider a Service to be the thing that people normally watch, such as CNN, BBC, Disney, etc.

4.3.1.3 Component

A *Component* is a piece of a Service. Depending on the type of broadcast system, a Service may be composed of several Components. Examples of such Components are video, audio 1, audio 2, and tele-text, etc.

4.3.1.4 Data

A *Data* is a piece of the legal and navigational information that is sent in the transmission streams. Examples of this kind of data, for DVB broadcasting, would be the Network Information Table (NIT), or the Event Information Table (EIT).

"Data" is not the same as "Data Component".

4.3.2 Relationship of the Tuner Information Types

The following figure illustrates the relationships of the Tuner Information Types:

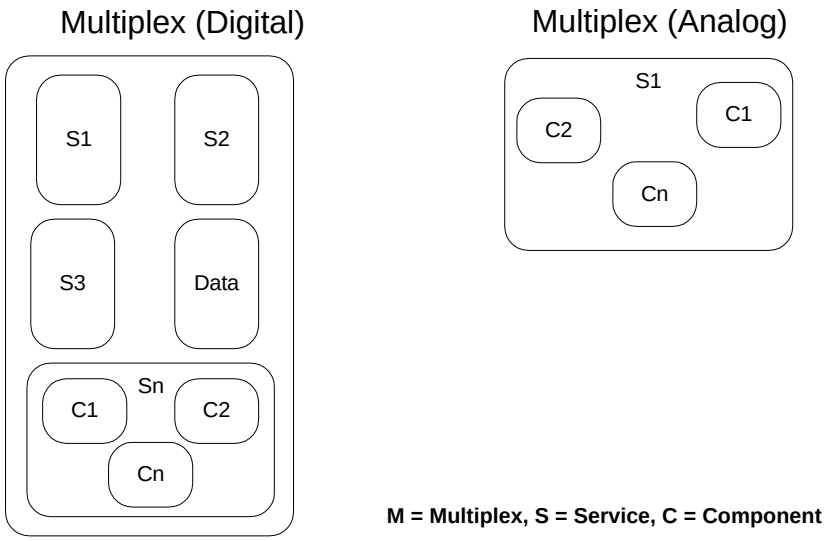


Figure 4.6 – Relationship of the Tuner Information Types

As the figure shows, the Multiplex carries one or more Services and data. In turn, the Service has one or more Components.

4.3.3 Tuner Information Model

The following figure is an example of all the information types exist in a Tuner, illustrating the Tuner Information Model with respect to the Information Types within the Tuner Subunit.

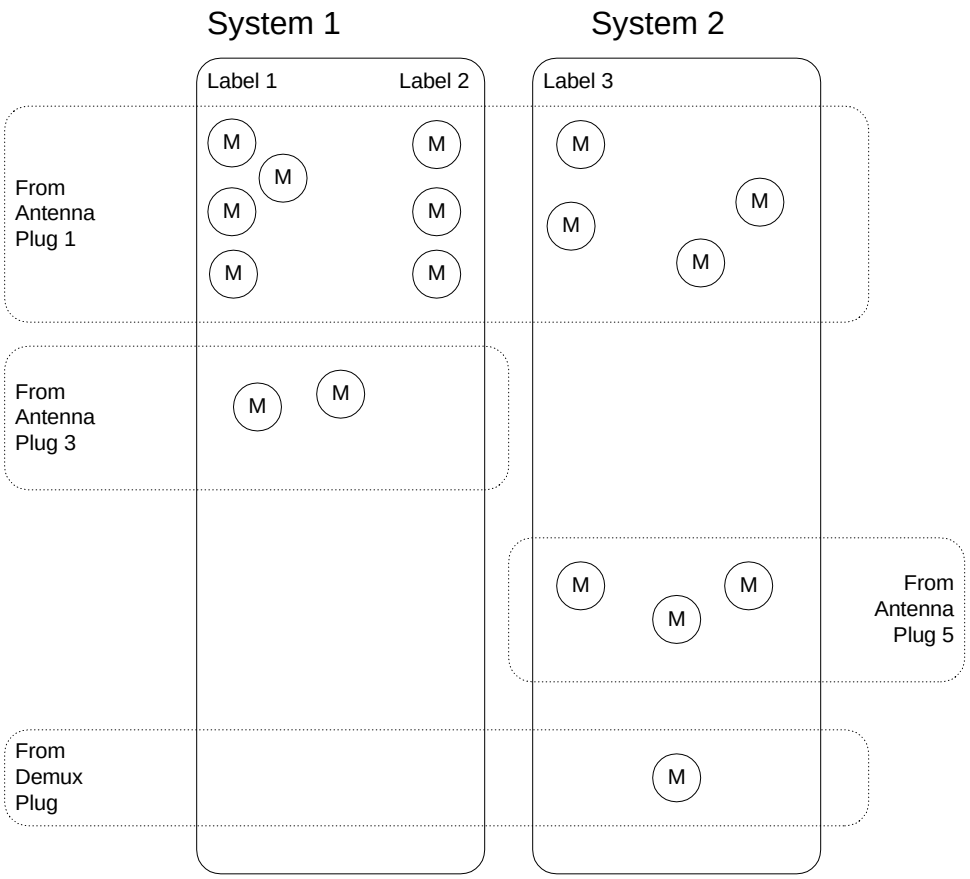


Figure 4.7 – Tuner Information Model

The figure shows three external antenna plugs and two broadcast systems. From external antenna input plug 1, the broadcast system 1 has two service providers: Label 1 and Label 2. From external antenna input plug 1, the broadcast system 2 has one service provider: Label 3. From external antenna input plug 3, the broadcast system 1 does not have service provider information, as well as broadcast system 2 from external antenna input plug 5.

The demux destination plug is receiving one Multiplex of broadcast system 2 in this example.

4.3.4 Tuner Information Instances

An Information instance is an actual piece of information in a Multiplex, including the Multiplex itself. It is one of the defined Information Types (Multiplex, Service, Component, or Data).

4.3.4.1 Available Information Instances

The Information Instances contained in an Input Signal. There are Available Information Instances from the demux destination plug and there are Available Information Instances from the antenna destination plug. These instances may be of any type.

4.3.4.2 Currently Selected Information Instances

A Tuner Subunit may have many Information Instances available at its selected destination plug. The tuner will be instructed by a controller to select one or more, or possibly all, of these Information Instances, so that they can be made available on the tuner source plugs. Those items that have been selected are referred to as the *currently selected information instances.*

4.3.4.3 Tuner Subunit Data Flow Model

The following figure illustrates the Tuner Data Flow Model where tuner Information Types pass through each stage of the Tuner Subunit functions. Only one Multiplex is selected from all the available Multiplexes on the antenna destination plug. Only one Multiplex is available for selection on the demux destination plug. Finally, one of these two Multiplexes is available to have the Services selected.

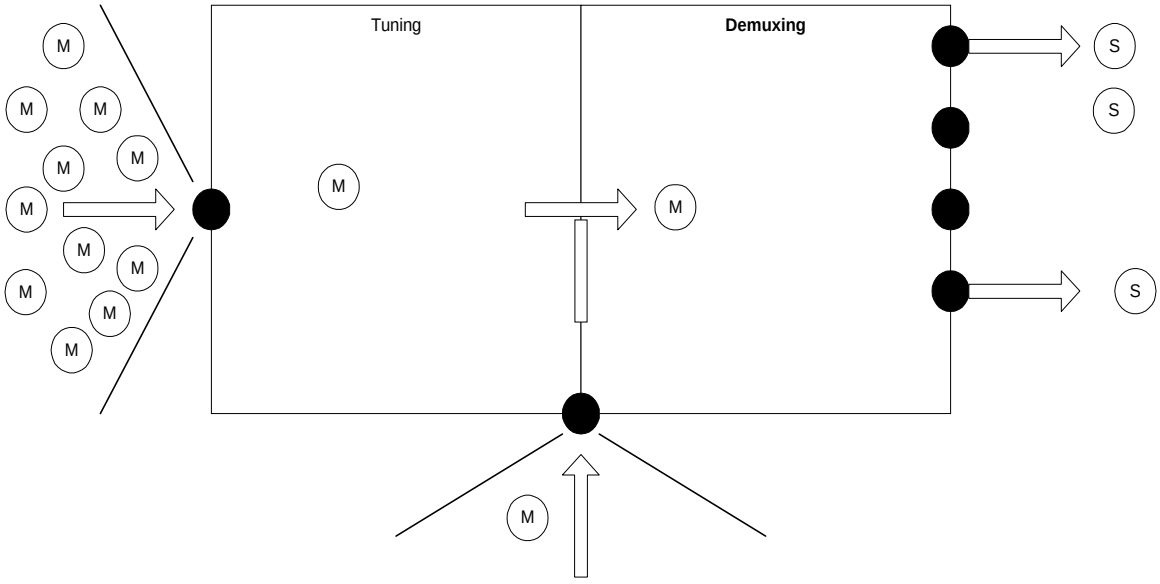


Figure 4.8 – Tuner Subunit Data Flow Model

4.3.5 Summary of the Tuner Model

- Supports all networks: Satellites, Cable, and Terrestrial.
- Has one tuning function and one demuxing function.
- Has two destination plugs: the antenna destination plug, and the demux destination plug.
- Has one or more source plugs.
- Can handle more than one broadcast system, but can handle only one at a time.
- Can handle more than one Multiplex, but can handle only one at a time.
- Can select as its output:
  - A complete Multiplex, or
  - A combination of Services, with selected Components, within a Multiplex

5. AV/C Tuner Descriptor Mechanism

With the exception of the Tuner Status Descriptor, which is Tuner Subunit specific, the AV/C Tuner Descriptor Mechanism is completely based on the AV/C Descriptor Mechanism described in reference [R3] and [R4].

5.1 Hierarchical view of the Tuner Descriptor Mechanism

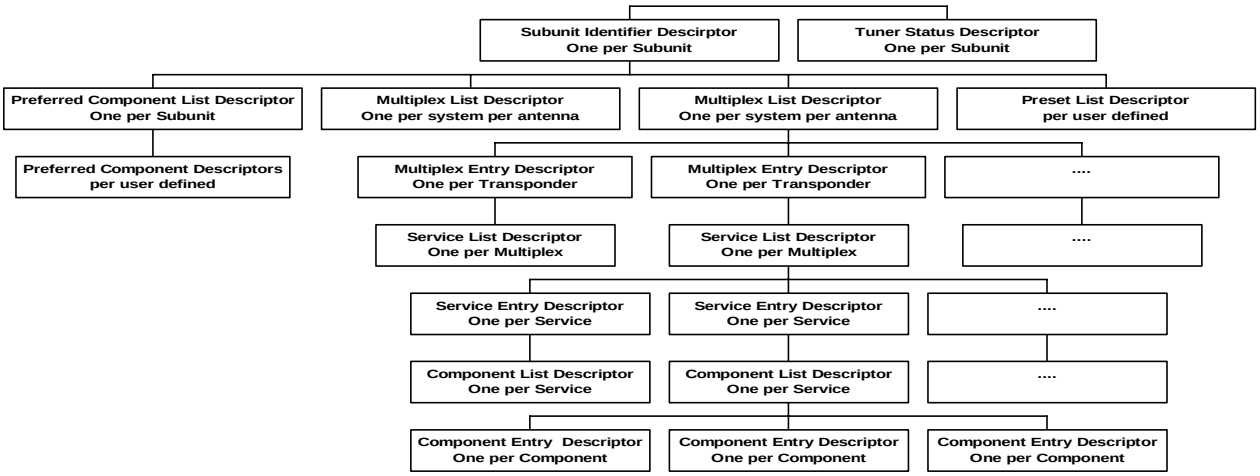


Figure 5.1 – Hierarchical view of the Tuner Descriptor Mechanism

As illustrated in the above figure, all the list descriptors at the first level, off of the Subunit Identifier Descriptor (SID), are **Root List Descriptors**. The hierarchical order of Multiplex, Service, and Component as described in Figure 4-1 is logically reflected in Figure 5-1, as Multiplex List Descriptor, Multiplex Entry Descriptor, Service List Descriptor, Service Entry Descriptor, Component List Descriptor, and Component Entry Descriptor.

In addition, the functions of a tuner are further enhanced by Preferred Components List Descriptor, Preferred Components Entry Descriptor, Preset List Descriptor, and Preset Entry Descriptor.

A Preferred Components Entry Descriptor specifies a collection of audio language preferences and subtitling language preferences, based on languages, in descending priority order. Only when the first language is not found in a Service, will the second language be selected and used. It may be created by the user and has the subunit scope (not limited to any specific broadcast system).

The Preset Entry Descriptor specifies a pre-selected combination of Components, a Service, a Multiplex, the antenna, and the broadcast system, facilitating a quick selection process based on a user's taste. It contains detailed specifications of the information types to be selected and can include a reference to a Preferred Components Entry Descriptor. The user may create the Preset List Descriptor as a root list descriptor and add Preset Entry Descriptors to its list. The preset descriptor has the subunit scope (not limited to any specific broadcast system).

The Tuner Subunit adds the following restrictions on the relationship of list descriptors and entry descriptors:

— The Multiplex List Descriptor contains Multiplex Entry Descriptor only.

— The Service List Descriptor contains Service Entry Descriptor only.

— The Component List Descriptor contains Component Entry Descriptor only.

— The Preferred Components List Descriptor contains Preferred Components Entry Descriptor only.

5.2 The Tuner Subunit Multiplex Hierarchies

A collection of Multiplex, Service, and Component lists as a group form what is called a MULTIPLEX HIERARCHY. This is a list hierarchy in which the Multiplex list is the root list, and in which several Service and Component lists may exist.

Implementation of Multiplex hierarchies is optional. However, if a Tuner Subunit does choose to support them, then the following rules shall apply:

1) All of the lists in this hierarchy are **READ ONLY**. The subunit shall return a NOT IMPLEMENTED response for any attempt by a controller to modify the lists in any way. The information in these lists is intended to be for status purposes, and is maintained only by the Tuner Subunit.

2) The portion of the hierarchy relating to the currently selected Multiplex shall be kept complete and accurate with respect to that Multiplex. This shall be done within the constraints of the physical limitations of the subunit (such as the availability of resources necessary to maintain the information), and the availability of the information from the air.

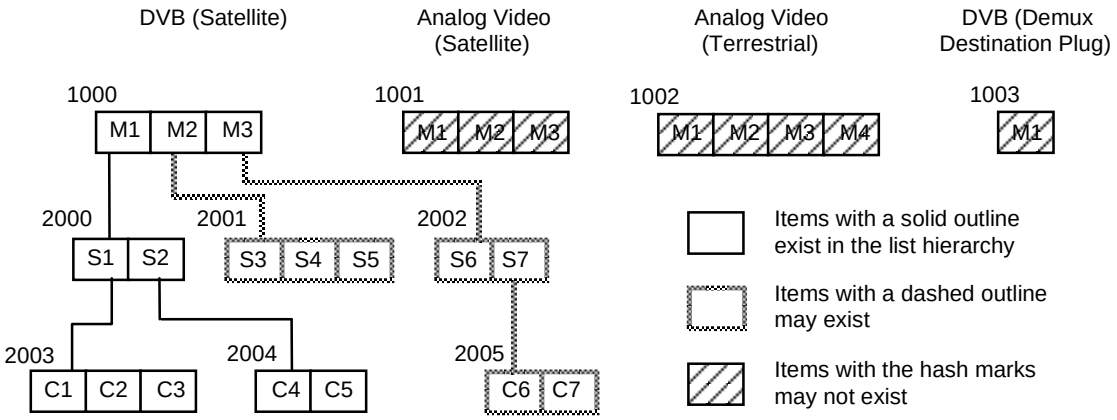
3) Providing information about other Services and Components on non-current Multiplexes is strongly recommended. In some situations, a certain level of information may be available, but other information may be inaccessible while those Multiplexes are not selected.

4) A tuner is strongly recommended to fill in as many as possible of the attributes in the Multiplex, Service, and Component entries. For selection attributes, the tuner shall provide all preferred attributes if they are defined or all mandatory attributes otherwise.

5) There shall be **one hierarchy per system for each receiver antenna and for each system on the tuner demux destination plug**. For example, consider a receiver unit which has a single satellite antenna which receives an analog video (1) and a DVB system (2), a terrestrial antenna which receives analog video (3), and the Tuner Subunit can receive a DVB signal on its demux (4). The Tuner Subunit shall maintain a set of four Multiplex hierarchies as shown in the diagram below.

6) The AV/C Descriptor Mechanism defines a set of rules for descriptor access, some of which shall be supported by all types of Subunits, and others that are optional.

The following diagram illustrates the relationship between these lists in the Multiplex hierarchies. It also illustrates how rule 5 is satisfied for an example receiver unit with a single satellite antenna which can receive both DVB and analog video signals, a terrestrial antenna that can receive analog video signals, and its Tuner Subunit which can accept a DVB signal on its demux destination plug:



1) For this example, multiplex M1 is currently selected (this means that the tuner is tuned to this multiplex - the entire multiplex may or may not be output to a source plug).

2) The solid lines indicate possible objects and lists IF THE MULTIPLEX HIERARCHIES ARE SUPPORTED.

3) The dashed lines indicate optional information that the tuner may provide. Note that for non-current multiplexes, some of this information may not be available.

Figure 5.2 – Example of the Multiplex Hierarchy Relationship

The following figure illustrates the overall Multiplex hierarchy of a Tuner Subunit:

A Tuner Subunit

Supports 3 broadcast systems

1 Multiplex List per system per antenna, and 1 List per system on the demux destination plug

List on Demux has 0 or 1 Entry

Service List

Service Entry

Component List

Component Entry

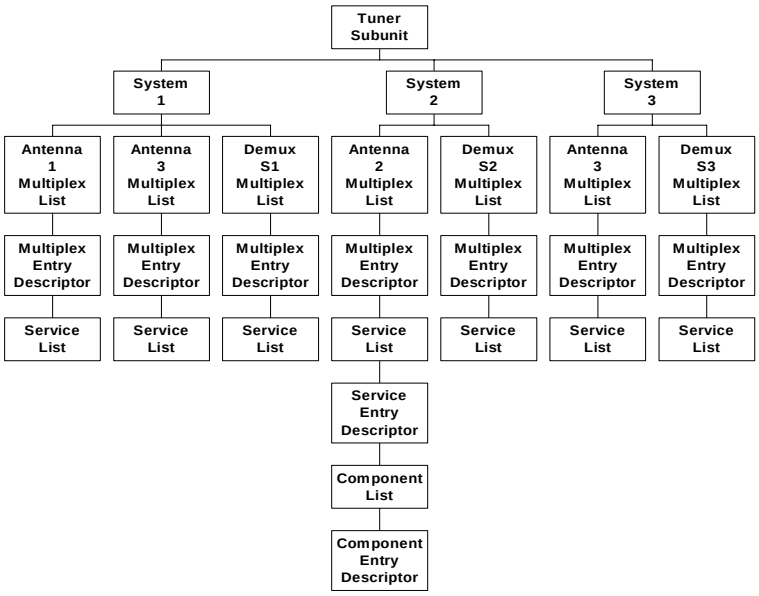


Figure 5.3 – Example of an Overall Multiplex Hierarchy of a Tuner Subunit

5.3 Rules and Guidelines for Tuner Subunit Objects and Object Lists

List Descriptor and Entry Descriptor are used to represent certain attributes or characteristics of technologies related to a particular type of subunit. Because each type of technology has its own individual set of characteristics, not all lists and entries can be assumed to behave in a "standard" way. This section covers rules for Tuner Subunits.

5.3.1 Migrating Objects

In the case of the Tuner Subunit, the Multiplex, Service, and Component lists all represent information that are derived from broadcast signals in the air. Most of the digital broadcast systems are designed to be flexible enough to allow Services to migrate from one transponder (hence, one Multiplex) to another for various reasons. Because we use list descriptors to represent related collections of Multiplexes and Services, it is possible that Service descriptors will also migrate from one list to another. This has implications on the way a controller should behave when dealing with these entities.

5.3.2 Object References

The AV/C Descriptor Mechanism defines two ways of referring to an entry in a list: either by its position in a specified list, or by its unique object ID.

Because the broadcast lists (Multiplex, Service, and Component) are so dynamic, it is not safe for a controller to assume that a given object will always be in the same position in a list. The action of one object leaving a list and joining another may cause a ripple effect that could shift several objects in the process. For this reason, it is much more reliable to refer to objects in the broadcast lists by their unique object IDs, because these IDs will be carried with them as they migrate across lists.

In order to have a consistent reference mechanism for controllers, the construction of object ID's will be defined per broadcasting system. These object ID rules will ensure that object ID's for a type of object (Multiplex, Service, or Component) will be unique across all lists of the same *list_type* within a given hierarchy. This allows controllers to use the OBJECT NUMBER SELECT command to select from these lists, and to use the "don't care" specification if desired.

Some tuner lists are much more static in nature, and will likely only change when the user wants them to. One example is the Preset list(s), of which there may be many supported by a Tuner Subunit (maybe one for each member of the home). Due to the nature of these kinds of lists, it is more convenient or even "natural" to refer to objects by their position. The semantics of selection from a preset list drive this concept; if several preset lists exist, each person that uses one will want to refer to the first preset, second, third, etc. It would be less useful if the first preset list had presets 1 to 10, the second preset list was required to have 11 to 20, and so on.

A Tuner Subunit is required to allow controllers to use both types of object references (by position or by ID). It is up to the controller to know when the appropriate reference type should be used.

5.3.3 Object ID Assignments

Because of the nature of certain types of Tuner Subunit lists, a set of rules have been defined for the numbering of objects within those lists:

1) The assignment of the object ID for the preset and preferred components entry descriptors can be defined by the tuner implementation. These assignments shall be within the general rules defined by the AV/C Descriptor Mechanism.

2) The assignment of the object ID for the Multiplex entry descriptors, the Service entry descriptors, and the Component entry descriptors are defined per *system_id*. For details, please refer to the appropriate AV/C Tuner Broadcast System Specification document.

5.3.3.1 *list_type* and *list_ID* of the List Descriptors

The following table presents the currently defined *list_type* and *list_ID* values for the Tuner Subunit lists:

Table 5.1 – Tuner Subunit *list_type* and *list_ID* values

List	*list_type* value	*list_ID* value	comments
Preferred_Components	90_{16}	1000_{16}	There is only one preferred components list, which is a root list. It shall have this fixed ID.
Multiplex	80_{16}	The Multiplex, Service, Component, and Preset list types all have ID values in the range: 1001_{16} - $2FFF_{16}$	These are root lists (their list_ID values are in the Tuner Subunit identifier descriptor). There shall be one Multiplex list per system on each receiver antenna and for each system on the tuner demux destination plug. For more details, please refer to the example illustrated in the section titled The Tuner Subunit Multiplex Hierarchies which begins on page 25.
Service	82_{16}		There may be many Service and component lists in a hierarchy. These are not root lists.
Component	84_{16}		See comments for the Service list.
Preset	$A0_{16}$		There may be several preset lists, all of which are root lists.
Reserved	all other values in the subunit type-specific range	3000_{16} - $3FFF_{16}$	This range is reserved for future specification for Tuner Subunits.

5.3.3.2 *entry_type* of the Entry Descriptors

The following table contains the *entry_type* definitions for Tuner Subunit-specific objects:

Table 5.2 – Tuner Subunit *entry_type* values

object entry	*entry_type* value	Comments
Multiplex	80_{16}	A Multiplex object entry may exist in either a Multiplex or a Preset list
Service	82_{16}	A Service object entry may exist in either a Service or a Preset list
Service With Specified Components	83_{16}	This object entry may exist in a Preset list
Component	84_{16}	This object entry may exist in a Component list
Preferred Components	90_{16}	This object entry may exist in a Preferred Components list
Reserved	all other values	reserved for future definition

5.3.4 An example of Object ID assignments

The following diagram illustrates the object ID assignment rules, and how object ID's are unique across all lists of the same *list_type* within a given hierarchy. In this Digital Audio Broadcasting (DAB) example, the object IDs for both instances of C1 is C1 because it is the SAME object in both instances. If a controller wanted to select the Alpha 1 radio audio component, then it should use the full path specification. If the controller did not care how the Component was selected by the tuner, then it could specify a "don't care" path:

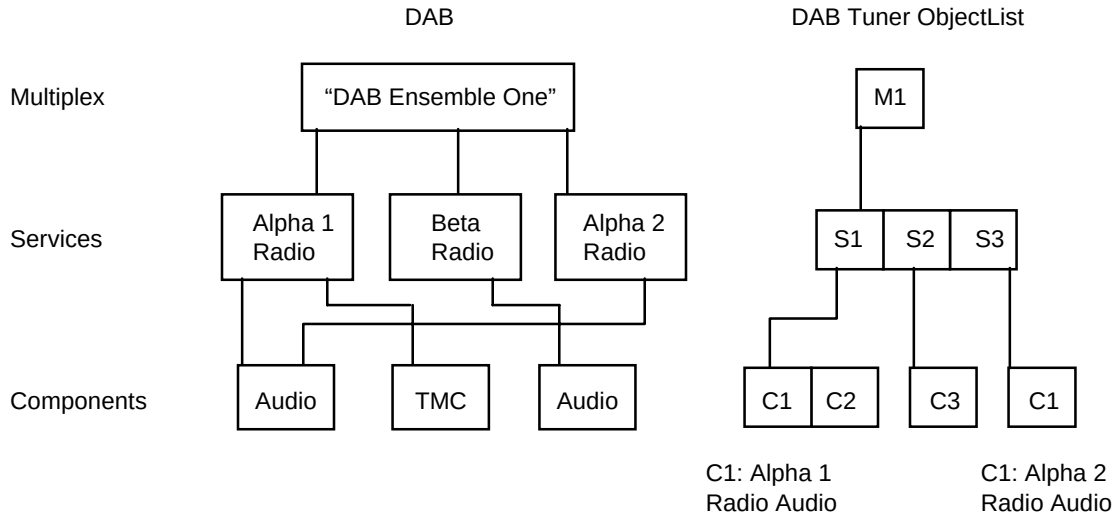


Figure 5.4 – Object ID Assignment Example

5.3.5 Text Field Encoding

All text fields in the descriptors specified in this document (Subunit Identifier Descriptor, Tuner Status descriptor, object lists, and objects) shall be encoded as minimum ASCII text as defined in the IEEE 1212 standard, section 8.1.4 of the 1994-10-05 edition. Each text character is one byte.

The exception to this rule is for those data structure fields which are defined by the particular broadcast system that they represent (for example, in the *entry_specific_information* fields of Multiplex, Service, and Component objects). In those cases, the broadcast system will specify the text field encoding rules.

NOTE — Regional variations of broadcast systems may affect the text field encoding of a structure. For example, the Japanese digital broadcast system is based on the European DVB system, but the Japanese system specifies two-byte character codes that are not in the European DVB specification (they are defined in ARIB STD-B5).

6. Tuner Descriptors

This section describes the definitions and values of the Tuner Subunit specific fields within the descriptors defined in the AV/C Descriptor Mechanism structure. Please refer to references [R3] and [R4] for the definitions and values of the general fields.

6.1 Tuner Subunit Identifier Descriptor

A Tuner Subunit shall have the same basic Subunit Identifier Descriptor structure as presented in the AV/C Descriptor Mechanism.

The Tuner Subunit Identifier Descriptor is READ ONLY. Its contents may change because of some external events such as antenna attachment / removal, CREATE DESCRIPTOR command execution, etc.

It describes the characteristics of the broadcast system(s) supported by the tuner. More than one broadcast system may be supported by a Tuner Subunit; this is called a multi-system tuner.

It is possible that there is no antenna attached for a specific broadcast system supported by the tuner.

It may have several list descriptors associated with it, called root list descriptors, through the link of *root_object_list_id_n* fields of the Subunit Identifier Descriptor.

All of the general fields of the Subunit Identifier Descriptor are as defined in the references [R3] and R4]. This document defines the *subunit_dependent_information* field only.

The *size_of_list_ID* field for Tuner Subunits shall be set to 2, indicating two (2) bytes.

The *size_of_object_ID* field is defined by each supported broadcast system specification.

The *size_of_object_position* field for Tuner Subunits shall be set to 2, indicating two (2) bytes.

The Tuner Subunit shall have the following *subunit_dependent_information* within the subunit identifier descriptor:

address offset	Contents
00_{16}	number_of_systems (n)
01_{16}	system[0]_specification
:	
:	:
:	system[n-1]_specification
:	

Figure 6.1 – Tuner *subunit_dependent_information*

The *number_of_systems* field, a one-byte field, specifies how many broadcast systems are supported by this Tuner Subunit.

For each supported broadcast system, there is one *system[x]_specification* field, with the following format:

address offset	Contents
00_{16}	specification_length
01_{16}	
02_{16}	system_id
03_{16}	implementation_profile_id
04_{16}	number_of_subsystem_labels (m)
05_{16}	sub_system_label_length[0]
06_{16}	sub_system_label[0]
:	
:	
:	:
:	sub_system_label_length[m - 1]
:	sub_system_label[m - 1]
:	
:	
:	multiplex_preferred_selection_flags
:	
:	service_preferred_selection_flags
:	number_of_antennas (n)
:	antenna[0]_specification
:	
:	
:	:
:	antenna[n – 1]_specification
:	
:	
:	system_specific_information_length
:	system_specific_information
:	
:	

Figure 6.2 – *system[x]_specification*

The *specification_length* field contains the number of bytes, which follow in this descriptor structure. The value of this field does *not* include the length field itself.

The *system_id* field indicates a broadcast system that the Tuner Subunit supports. The following broadcast systems are currently defined:

Table 6.1 – *system_id*

system_id	name
10_{16}	analog video
11_{16}	analog audio
20_{16}	DVB (Digital Video Broadcast)
21_{16}	Reserved for DAB (Digital Audio Broadcast)
22_{16}	ATSC DTV
23_{16}	Rec. ITU-R BO. 1294 System B
all others	Reserved for future specification

NOTE —For convenience, the system_id values, which are currently defined, have been listed above. However, the detailed specifications for each broadcast system specification, including its profile definitions, are defined in separate documents. Future broadcast systems, when defined, will ONLY have their system_id values defined in their documentation; this tuner specification document might NOT be updated just to add new system_id values.

The *implementation_profile_id* field specifies the profile ID of the tuner implementation **for this *system_id***. Note that a Tuner Subunit may be implemented with a different profile ID for each of the systems that it supports. There shall be one profile for each supported system. The profile values are defined in the section titled Tuner Subunit Profiles, which begins on page 85.

The *number_of_subsystem_labels* field indicates the number of service providers the tuner is capable of dealing with. If the *number_of_subsystem_labels* field is zero, the tuner is not bound to any particular service provider and the *sub_system_label_length[x]* and *sub_system_label[x]* fields shall not exist.

The *sub_system_label_length[x]* and *sub_system_label[x]* fields can identify a particular service provider. For example, “PerfecTV” or “Canal Plus” are network providers in the DVB system. The label length field defines how many bytes are in the text string.

The presence of the *sub_system_label[x]* does not mean that this tuner is unable to select from OTHER service providers; it simply means to associate the preferred selection flags (defined next) with service providers, to assist in tuning actions. If the *sub_system_label[x]* field is empty, the preferred selection flags must NOT be ignored; it is possible that a service provider name was not available, or there is no particular name associated with a given system.

The *multiplex_preferred_selection_flags* and *service_preferred_selection_flags* indicate the set of preferred selection attributes. There is a 1 to 1 correspondence between the bits in these fields and the corresponding bits in the *system_specific_multiplex_attributes_valid_flags* and *system_specific_service_attributes_valid_flags* for the broadcast system being specified. For the preferred selection flags, only those bits may be set which correspond to mandatory selection attributes. Therefore, the preferred attributes are a subset of the mandatory selection attributes, and are guaranteed to select a single Service *for these particular service providers*.

In some circumstances, a tuner may be designed to supports more than one service provider, but each provider has the same preferred selection attributes. This could lead to selection conflicts if the values that are assigned to those attributes were to overlap among the service providers. To remedy this, the Tuner Subunit may define additional preferred selection flags, which are used to resolve any potential conflicts and to ensure that selections succeed.

As an example of the above, consider two service providers, A and B. To select a Service from provider A, the tuner only needs to know the *service_id*; so, the tuner assigns only the *service_id* preferred selection flag for provider A. Likewise, provider B also can uniquely select Services within its domain using only

service_id, so the tuner also assigns only that preferred selection flag. If the two providers both have a Service with ID 100, then a controller which asked for Service 100 would not be sure which one it was getting: either from provider A or provider B. To fix this, the Tuner Subunit may assign another preferred selection flag, for *original_network_id*. Now the controller knows that it must provide both of these attributes when specifying a selection. Thus, it will specify either {network A, 100} or {network B, 100} when making the selection, and it will get exactly what it wanted.

Mandatory selection attributes for each broadcasting system are presented in the broadcast system specification. For each preferred selection attribute, its corresponding bit shall be set to 1. If there are no special requirements for preferred selection, then these flags must all be set to the value 0.

Regardless of the values of these flags, the sizes of the *multiplex_preferred_selection_flags* and *service_preferred_selection_flags* must match the sizes of the *system_specific_multiplex_attributes_valid_flags* and *system_specific_service_attributes_valid_flags* of the specific broadcast system, respectively.

The *number_of_antennas* field contains the number of receiver unit antennas *available* to this Tuner Subunit for this broadcast system. If there isn't any antenna for this broadcast system available to this tuner, the *number_of_antennas* should be zero and the *antenna[n]_specification* fields shall not exist. Note that there will only be zero or one antenna *connected* to the subunit antenna destination plug at any given time. It is possible that a physical antenna may support multiple broadcast systems, with similar *antenna[n]_specifications* of the same antenna in each of the supported broadcast system. Additionally, it is possible that a broadcast system supports multiple bands of frequencies and different types of signal format, with multiple *antenna[n]_specifications* of the same antenna in a broadcast system.

The *antenna[n]_specification* fields are an array of antenna specifications. All antenna specifications have this format:

<table>
<tr><th>address offset</th><th colspan="4">Contents</th></tr>
<tr><td>00_{16}</td><td>mobile</td><td>movable</td><td>reserved (3 bits)</td><td>transport (3 bits)</td></tr>
<tr><td>01_{16}</td><td colspan="4">external_input_plug_number</td></tr>
<tr><td>02_{16}</td><td colspan="4" rowspan="2">system_specific_antenna_range_specification_length</td></tr>
<tr><td>03_{16}</td></tr>
<tr><td>04_{16}</td><td colspan="4" rowspan="3">system_specific_antenna_range_specification</td></tr>
<tr><td>:</td></tr>
<tr><td>:</td></tr>
</table>

Figure 6.3 – *antenna[n]_specification*

The *mobile* and *movable* flags indicate these characteristics of the antenna mechanism, according to these combinations:

Table 6.2 – mobile / movable flags

mobile/movable flags	meaning
00	fixed – the antenna does not change its orientation
01	movable – the antenna mechanism can be adjusted to different orientations (such as a satellite dish)
10	mobile – the antenna is mounted in a fixed position on a platform which may be changing its location and orientation at any time (such as a car antenna)
11	mobile and movable - the antenna has both of these characteristics

The *transport* field is defined as follows:

Table 6.3 – *transport*

transport (binary)	meaning
001	satellite
010	cable
011	terrestrial
all others	reserved for future definition

The *external_input_plug_number* field specifies the receiver unit's external input plug number, which is an external antenna input plug that corresponds with this antenna specification. Notice that the external antenna input plug is simply one type of the external input plugs.

The *system_specific_antenna_range_specification_length* and *system_specific_antenna_range_specification* fields contain information that is specific to the broadcast system.

The *system_specific_antenna_range_specification* consists of several *selection_attribute_range_specification* structures. It has this basic format (the selection attribute labels A, B...X correspond to labels used in the system specific attribute range specifications - their purpose is to identify each attribute in the range specification structure):

system_specific_antenna_range_specification structure
selection_attribute_range_specification (for selection attribute A)
selection_attribute_range_specification (for selection attribute B)
: :
selection_attribute_range_specification (for selection attribute X)

Figure 6.4 – *system_specific_antenna_range_specification*

The collection of *selection_attribute_range_specification* structures are stored sequentially. There is no count field at the beginning of the collection, because there will be one structure per selection attribute which has a range specification.

The *selection_attribute_range_specification* presents the range of values for one of the selection attributes supported by tuner for this **antenna**. The collection of selection attributes for which ranges are defined is broadcast system specific. There is no particular relationship between the set of selection attributes with range specifications, and the mandatory or preferred selection attributes. For details on which selection attributes have such definitions, refer to the tuner broadcast system specification.

The units of measure for each attribute are also defined by the particular broadcast system. A controller will have to understand the broadcast system in order to make use of these detailed specifications.

The basic format of a *selection_attribute_range_specification* is as follows:

address offset	Contents		
00_{16}	list_flag	range_flag	size_of_attribute (s) (6 bits)
01_{16}	list_and_range_flag_specific_contents		
:			
:			

Figure 6.5 – *selection_attribute_range_specification*

The *list_flag* bit indicates whether the range contents are specified as a list of values, or as a single value.

The *range_flag* bit indicates whether the range contents are specified as a discrete or continuous range of values.

The *size_of_attribute* field specifies the size, in bytes, of the selection attribute for which the range is specified.

The *list_and_range_flag_specific_contents* field will have a format which depends on the four combinations of *list_flag* and *range_flag*. The four combinations have the following definitions:

Address offset	combination 0,0 (no range specified)		
00_{16}	0	0	size_of_attribute (s) = 0

Figure 6.6 – *list_flag* and *range_flag (0,0)*

When both flags are zeroes, there is no range specification for this selection attribute. The *size_of_attribute* field shall be set to zero.

Address offset	combination 0,1 (single value range)		
00_{16}	0	1	size_of_attribute (s)
01_{16}	starting_value		
:			
s			
s + 1	ending_value		
:			
2s			

Figure 6.7 – *list_flag* and *range_flag (0,1)*

When the two flags have the value {0,1}, then the attribute range specification indicates a single value range.

The *size_of_attribute* field indicates the number of bytes used for each value.

The *starting_value* field contains the beginning value of the range.

The *ending_value* field contains the ending value of the range.

address offset	combination 1,0 (list of item values)		
00_{16}	1	0	size_of_attribute (s)
01_{16}	number_of_values (n)		
02_{16}	value[0]		
:			
s + 1			
:	:		
:	value[n – 1]		
:			
s * n + 1			

Figure 6.8 – *list_flag* and *range_flag (1,0)*

When the *list_flag* and *range_flag* bits are {1,0} then the attribute range specification holds a collection of discrete values for this attribute.

The number of bytes used for each of these entries is equal to *size_of_attribute*.

The *number_of_values* field indicates how many of these discrete values are specified in the structure.

The *value[x]* fields each hold one of the values.

Address offset	Combination 1,1 (list of item ranges)		
00_{16}	1	1	size_of_attribute (s)
01_{16}	number_of_ranges (n)		
02_{16}	starting_point[0]		
:			
s + 1			
:	ending_point[0]		
:			
2s + 1			
:	:		
:	:		
:	starting_point[n - 1]		
:			
:			
:	ending_point[n - 1]		
:			
2 * s * n + 1			

Figure 6.9 – *list_flag* and *range_flag (1,1)*

When the *list_flag* and *range_flag* bits are {1,1} then the attribute range specification holds a collection of range values for this attribute.

The *size_of_attribute* field indicates the number of bytes used for each value.

The *number_of_ranges* field indicates how many of these range values are specified in the structure.

Each range is specified by the pair of *starting_point[x]* and *ending_point[x]* entries.

The *system_specific_information_length* and *system_specific_information* fields contain information that is specific to the broadcast system.

6.2 Tuner Status Descriptor

The tuner status descriptor structure is specific to Tuner Subunits. It holds information about the Tuner Subunit in general, and about what information are on each of its source plugs. In contrast to the subunit identifier descriptor, the information in this structure is dynamic, and is kept up to date by the tuner. This

structure is READ ONLY and may be examined by a controller in order to determine the operational status of the tuner and its source plugs. The controller may also ask for notification of changes to this descriptor. For details, please refer to the TUNER STATUS command.

The general format of the tuner status descriptor is as follows:

Address	Contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	general_tuner_status
:	
:	
:	number_of_source_plugs (n)
:	source_plug_status[0]
:	
:	
:	:
:	source_plug_status[n - 1]
:	
:	

Figure 6.10 – Tuner Status Descriptor

The *descriptor_length* field indicates the number of bytes which follow in this descriptor structure. The value of this field does *not* include the length field itself.

The *general_tuner_status* field contains status information about the Tuner Subunit which is not specific to a particular source plug. The *general_tuner_status* field has this format:

address offset	Contents
00_{16}	antenna_input_info_length
01_{16}	antenna_input_info
:	
:	
:	system_specific_multiplex_selection_length
:	system_specific_multiplex_selection
:	
:	
:	demux_input_info_length
:	demux_input_info
:	
:	

Figure 6.11 – *general_tuner_status* field format

The *antenna_input_info_length* field contains the number of bytes for the following *antenna_input_info* field. If the Subunit's antenna destination plug is not connected to any external antenna input plug, the *antenna_input_info_length* should be zero and there is no *antenna_input_info* field.

The *antenna_input_info* field gives the current status information about the antenna destination plug of the Tuner Subunit and has the following format:

Address offset	msb							lsb
00_{16}	active_system_id							
01_{16}	searching	moving	no_RF	reserved				
02_{16}	input = 0	selected_antenna						
:	antenna_general_system_info							

Figure 6.12 – *antenna_input_info*

The *active_system_id* field specifies the *system_id* of the currently active system. It is one of the *system_id* values as contained in the subunit identifier descriptor.

The *searching* field indicates whether the tuner is searching for a Broadcast system, a Multiplex, or a Service. A value of 1 indicates searching.

The *moving* field indicates whether the antenna is moving (in the case of a movable antenna). A value of 1 indicates moving.

The *no_RF* field indicates whether there is currently a RF (Radio Frequency) carrier present (e.g. while searching). A value of 0 means the RF carrier is present. If the Tuner Subunit is not capable of detecting the presence of the RF carrier, it shall set this field to 0.

If either *searching*, *moving*, or *no_RF* is set, the *antenna_general_system_info* field may contain values that are invalid.

The *input* field indicates that the Multiplex is being received on the antenna destination plug (the input value is 0).

The *selected_antenna* field specifies the receiver antenna number which is currently connected to the Tuner Subunit's antenna destination plug. The *selected_antenna* value is the index (0 based) of an *antenna_specification* in the *system[x]_specification* for the subunit, as described in the Subunit Identifier Descriptor data structure, *subunit_dependent_information* field. The actual external input plug number is in the *external_input_plug_number* field of the *antenna_specification*.

The *antenna_general_system_info* field contains status information about the currently active broadcast system; this information is not related to a particular source plug of the Tuner Subunit, so it is called "antenna general system info". The format and contents of this field is of course system specific, so the details can be found in the appropriate AV/C Tuner Broadcast System Specification document.

The *system_specific_multiplex_selection_length* field specifies the number of bytes for the following *system_specific_multiplex_selection* field, which is described in details in section 7.3 Multiplex Selection and Information Fields on page 59. A special note on the *selected* flag field of the selection attributes: since this Multiplex is the currently selected Multiplex within the Tuner Subunit, the *selected* flag shall be set to reflect the selection.

The *demux_input_info_length* field contains the number of bytes for the following *demux_input_info* field.

The *demux_input_info* field gives the current status information about the demux destination plug of the Tuner Subunit and has the following format:

Address offset	msb							lsb
00_{16}	active_system_id							
01_{16}	searching	reserved						
02_{16}	input = 1	reserved						
:	demux_general_system_info							

Figure 6.13 – *demux_input_info*

The *active_system_id* field specifies the *system_id* of the currently active system on the demux destination plug. It is identical to one of the *system_id* values as contained in the subunit identifier descriptor.

The *searching* field indicates whether the tuner is searching. The Demuxing function may be searching for a Service. A value of 1 indicates searching.

The *input* field indicates that the Multiplex is being received on the demux destination plug (the input value is 1).

The *demux_general_system_info* field contains status information about the currently active broadcast system; this information is not related to a particular source plug of the Tuner Subunit, so it is called "demux general system info". The format and contents of this field is of course system specific, so the details can be found in the appropriate AV/C Tuner Broadcast System Specification document.

The *number_of_source_plugs* field contains the number of source plugs on the Tuner Subunit, and hence it indicates the number of *source_plug_status[x]* fields that are in this status structure.

The *source_plug_status[x]* fields each contain status information for their respective source plugs. There shall be one of these structures for each source plug on the tuner, even if the plug currently has no status information to report. These fields are each split into two general areas, as shown below:

address offset	Contents	
00_{16}	source_plug_number	
01_{16}	attributes	
02_{16}	Input	reserved
03_{16}	data_status_length	
04_{16}	data_status	
:		
:		
:	info_type_status_length	
:	info_type_status	
:		
:		

Figure 6.14 – *source_plug_status[x] format*

The *source_plug_number* field indicates the actual source plug number.

The *attributes* field has the following format:

Table 6.4 – *attributes* value

Attribute value	Name	Meaning
1xxx xxxx	has_more_attributes	If this bit is set to 1, then the next byte is also an attributes byte. If this bit is 0, then the next byte is as defined for this structure.
xxxx xxx1	has_an_additional_field	If this bit is set to 1, then an additional field is attached at the end of the currently defined structure. If this bit is 0, then there is no additional field for this structure.
all others		Reserved for future specification.

The *input* bit indicates which of the Tuner Subunit *destination* plugs is receiving the signal that is going to this source plug. The values are 0 = antenna destination plug, 1 = demux destination plug.

The *data_status_length* field contains the number of bytes for the following *data_status* field.

The *data_status* field holds the status information for all data that has been placed on this plug by the DIRECT SELECT DATA command (see page 80). The format of this field is as follows:

address offset	msb							lsb
00_{16}	status							
01_{16}	number_of_selection_specifications (n)							
02_{16}	flowfunction[0]							
03_{16}	dsd_selection_specification[0]							
:								
:								
:	:							
	flowfunction[n - 1]							
:	dsd_selection_specification[n - 1]							
:								
:								

Figure 6.15 – *data_status*

The *status* field describes the current situation of the specified data, as defined in the table below:

Table 6.5 – *status*

value	status description
00_{16}	None of the information instances listed in the following fields are currently on the specified source plug, or no information instances are selected.
10_{16}	All information instances listed in the following fields are currently on the specified source plug.
20_{16}	Some information instances listed in the following fields are on the specified source plug (examine the *currently_available* flags in the *data_status* and *info_type_status* fields).

All other operands are as described for the DIRECT SELECT DATA control command (see page 80).

The *info_type_status_length* field contains the number of bytes for the following *info_type_status* field.

The *info_type_status* field holds the status information for all of the information instances which have been placed on this plug by the DIRECT SELECT INFORMATION TYPE and OBJECT NUMBER SELECT control commands. The format of this field is as follows:

address offset	msb							lsb
00_{16}	status							
01_{16}	number_of_selection_specifications (n)							
02_{16}	dsit_selection_specification [0]							
:								
:								
:	:							
:	:							
:	:							
:	dsit_selection_specification [n - 1]							
:								
:								

Figure 6.16 – *info_type_status*

The *status* field is as described above for the *data_status* field.

All other operands are as described for the DIRECT SELECT INFORMATION TYPE control command (see page 64).

If a complete Multiplex is selected, the value of the *number_of_selection_specifications* shall be zero and there will be no *dsit_selection_specification[x]* operands. If the complete Multiplex is on the specified source plug, the value of the *status* field shall be 10_{16}.

Note that when an OBJECT NUMBER SELECT control command is performed by the Tuner Subunit, it shall use the appropriate detailed selection specification information from the selected object(s) to create the corresponding *dsit_selection_specification[x]* fields in this status structure. Hence, there are no *ons_selection_specification[x]* fields added to this structure. To determine which *objects* are on a plug, the

OBJECT NUMBER SELECT status command may be used (this also applies as a general solution for all Subunits which support the OBJECT NUMBER SELECT command).

For both *data_status* and *info_type_status* , if the plug number indicated by the *source_plug_number* field is not currently used to output data or information instances, then the corresponding *status* field shall be 00_{16}. In this case, none of the fields beyond *status* shall exist in the *data_status* or *info_type_status* fields. The *data_status_length* and/or *info_type_status_length* fields shall be set accordingly.

6.2.1 Descriptor Identifier for the Tuner Status Descriptor

The tuner status descriptor is specific to the Tuner Subunit type; it has the following type value from the range of subunit-specific descriptor types (for details on the general descriptor types, please refer to the explanation of the OPEN DESCRIPTOR command, which can be found in the Reference [R3]:

Table 6.6 – descriptor_type of the Tuner Status Descriptor

descriptor_type	Meaning
80_{16}	Tuner Status Descriptor

The *descriptor_type_specific_reference* field does not exist (the *descriptor_identifier* consists of only the *descriptor_type* field) because there can be only one tuner status descriptor for a Tuner Subunit.

6.3 Tuner Multiplex List Descriptor

The tuner Multiplex List Descriptor is a root list descriptor. There is a Multiplex List Descriptor per broadcast system per antenna, and per supported broadcast system on the tuner demux destination plug.

The Multiplex List Descriptor exists regardless if there is any entry descriptor in its list. In addition, it contains Multiplex Entry Descriptor only.

The Multiplex list contains one or more Multiplex objects, each of which describes one of the currently accessible Multiplexes. When a tuner is “tuned” into a Multiplex, then we say that the Multiplex is selected. There can be at most one Multiplex selected at any give time. The entries in the Multiplex list shall be Multiplex descriptors, as defined below. The exact contents of these descriptors will vary by the type of broadcast system, so the details can be found in the appropriate AV/C Tuner Broadcast System Specification document.

The Multiplex List Descriptor is as follows:

Address	Contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	list_type = 80_{16}
00 03_{16}	attributes = xx01 xxxx
00 04_{16}	size_of_list_specific_information
00 05_{16}	
:	number_of_entries (n)
:	
:	object_entry[0]
:	
:	
:	:
:	object_entry[n-1]
:	
:	

Figure 6.17 – Tuner Multiplex List Descriptor

The *attributes* for the Multiplex List indicate that all entries in the list have *object_ID* values. The bit which indicates *has_child_ID* is for object entries only, so it is required to be set to 0 for all lists. Currently, there is only one byte defined for list attributes, so the msb would be 0 also.

Note that currently there is no *list_specific_information* for the Multiplex List, so the *size_of_list_specific_information* field shall be 00_{16}.

The *object_entry[x]* field is the Multiplex Entry Descriptor described below.

6.4 Tuner Multiplex Entry Descriptor

The tuner Multiplex Entry Descriptor is the entry descriptor of the tuner Multiplex List Descriptor.

The Multiplex object specifies one broadcast Multiplex. The format of the object entry is as follows:

address offset	Contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 80_{16}
03_{16}	attributes = xxx0 xxxx
:	child_list_ID (if specified by the has_child_ID flag)
:	
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information (MSB)
:	(LSB)
:	entry_specific_information
:	
:	

Figure 6.18 – Multiplex Entry Descriptor

The *child_list_ID* field contains the ID of a Service List Descriptor, if specified by the *has_child_ID* flag in the *attributes* field.

The Multiplex may or may not have a Service list associated with it, depending on the broadcast system and the status of the tuner. The *has_child_ID* attribute bit will be set accordingly.

The *size_of_entry_specific_information* field specifies the number of bytes used for the following *entry_specific_information* field. The size field is two bytes, and is NOT included in this calculation.

The *entry_specific_information* field of the Tuner Subunit Multiplex Entry Descriptor shares the same structure for all broadcast systems:

Address Offset	msb							lsb
00_{16}	system_id							
01_{16}	input	antenna_number						
02_{16}	system_specific_multiplex_selection							
:								
:								
:	system_specific_multiplex_information							
:								
:								

Figure 6.19 – Multiplex Entry Descriptor *entry_specific_information*

The *system_id* field identifies the type of system (e.g. DVB, analog video) described by this tuner object. The values are defined in the table of *system_id* values presented earlier.

The *input* field indicates which input, either the antenna or demux destination plug, the subunit should use to get the requested service(s). The following table illustrates the values defined for this field:

Table 6.7 – General Multiplex *input* Field

Value for input	Meaning
0	Take input from the antenna destination plug.
1	Take input from the demux destination plug.

The *antenna_number* field is the index of an antenna specifier for the subunit, as described in the subunit identifier descriptor data structure. This is a zero-based value. If the demux destination plug is selected, then the *antenna_number* field has no meaning.

The *system_specific_multiplex_selection* field specifies the selection attributes that uniquely identify a Multiplex and the *system_specific_multiplex_information* field specifies the information attributes that describe the characteristics of the Multiplex, but are not used for selection purpose. Please see section 7.3 Multiplex Selection and Information Fields on page 59 for detailed descriptions.

6.5 Tuner Service List Descriptor

Within a given Multiplex, there may be zero, one, or more Services available (zero Services means that those which used to exist have since disappeared). A Service list holds Service descriptor objects, each of which describes a Service in the Multiplex. The entries in the list shall all describe Services from the **SAME** Multiplex. The number of Services may change from time to time, so it cannot be assumed that this list will be fixed. Each Multiplex object in the Multiplex list will have at most one Service list. The currently selected Multiplex shall have a Service list; for others, non-selected Multiplexes, it MAY be possible to create a Service list, but this is not guaranteed. It is dependent on the broadcast system capabilities, and on the broadcast system implementation. For example, the DVB system has the potential to carry service information about Services on other Multiplexes. However, broadcasters are not required to provide this information in the air. When a new Multiplex is selected, then the Tuner Subunit shall make sure that its Service list is updated (or created) and kept up to date for as long as that Multiplex is selected. The subunit may choose to keep the Service lists of the formerly selected Multiplex(es) around, but it is not required to do so.

The Tuner Service List Descriptor contains Service Entry Descriptor only.

The Service List Descriptor is as follows:

Address	Contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	list_type = 82_{16}
00 03_{16}	attributes = xx01 xxxx
00 04_{16}	size_of_list_specific_information
00 05_{16}	
00 06_{16}	number_of_entries (n)
00 07_{16}	
00 08_{16}	object_entry[0]
:	
:	
:	:
:	object_entry[n-1]
:	
:	

Figure 6.20 – Tuner Service List Descriptor

The *attributes* for the Service List indicate that all objects in the list have *object_ID* values. The bit which indicates *has_child_ID* is for object entries only, so it is required to be set to 0 for all lists. Currently there is only one byte defined for list attributes, so the msb would be 0 also.

Note that currently there is no *list_specific_information* for the Service List, so the *size_of_list_specific_information* field shall be 00_{16}.

The *object_entry[x]* field is the Service Entry Descriptor described below.

6.6 Tuner Service Entry Descriptor

The Service object specifies one complete Service within a Multiplex. The format of the object entry is as follows:

address offset	Contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 82_{16}
03_{16}	attributes = xxx0 xxxx
:	child_list_ID (if specified by the has_child_ID flag)
:	
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information
:	
:	

Figure 6.21 – Service Entry Descriptor

The *child_list_ID* field contains the ID of a Component List Descriptor, if specified by the *has_child_ID* flag in the *attributes* field.

The Service object may or may not have a child list, so its *has_child_ID* flag will be set accordingly.

The *size_of_entry_specific_information* field specifies the number of bytes used for the following *entry_specific_information* field. The size field is two bytes, and is NOT included in this calculation.

The *entry_specific_information* field of the Tuner Subunit Service Entry Descriptor shares the same structure for all broadcast systems:

Address Offset	msb							lsb
00_{16}	system_id							
01_{16}	system_specific_service_selection							
:								
:								
:	system_specific_service_information							
:								
:								

Figure 6.22 – Service Entry Descriptor *entry_specific_information*

The fields for this common structure are very similar to the Multiplex entry descriptor above, but Service entry descriptors do not have (or need) the input and antenna field. The reason for this is that the selection process for a given type of object (Multiplex, Service, or Component) requires the specification of the appropriate objects higher in the hierarchy.

The *system_id* field has the same meaning as described in Multiplex Entry Descriptor.

The *system_specific_service_selection* field specifies the selection attributes that uniquely identify a Service and the *system_specific_service_information* field specifies the information attributes that describe the characteristics of the Service, but are not used for selection purpose. Please see section 7.4 Service Selection and Information Fields on page 61 for detailed descriptions.

6.7 Tuner Component List Descriptor

Within a given Service, there may be zero, one, or more Components available (zero Components means that those which used to exist have since disappeared). A Component list holds the Component descriptor objects of exactly ONE Service. Each object describes exactly one Component. There will be one Component list for each Service in a Multiplex. If it is not possible to select an individual Component (as is the case with most analog video systems), then the Service may not have such a list.

As with the number of Services in a Multiplex, the number of Components in a Service may vary over time, so this list can not be assumed to be fixed. In most cases, it will probably not be possible to construct or reliably maintain a Component list for Services, which are not on selected Multiplexes. This is because the tuner would normally be required to select that Multiplex and analyze the signals in order to construct the Component list. As with the Service lists, when a new Multiplex is selected, the Tuner Subunit shall update (or create) the Component list for each Service and shall keep them up to date. It has the option of retaining Component lists for Services on non-selected Multiplexes.

The Tuner Component List Descriptor contains Component Entry Descriptor only.

The Component List Descriptor is as follows:

Address	Contents
$00\ 00_{16}$	descriptor_length
$00\ 01_{16}$	
$00\ 02_{16}$	list_type = 84_{16}
$00\ 03_{16}$	attributes = xx01 xxxx
$00\ 04_{16}$	size_of_list_specific_information
$00\ 05_{16}$	
$00\ 06_{16}$	number_of_entries (n)
$00\ 07_{16}$	
$00\ 08_{16}$	object_entry[0]
:	
:	
:	:
:	object_entry[n-1]
:	
:	

Figure 6.23 – Component List Descriptor

The *attributes* for the Component List indicate that all objects in the list have ID values. The bit which indicates *has_child_ID* is for object entries only, so it is required to be set to 0 for all lists. Currently, there is only one byte defined for list attributes, so the msb would be 0 also.

Note that currently there is no *list_specific_information* for the Component List, so the *size_of_list_specific_information* field shall be 00_{16}.

The *object_entry[x]* field is the Component Entry Descriptor described below.

6.8 Tuner Component Entry Descriptor

The format of the Component object entry is as follows:

Address offset	Contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 84_{16}
03_{16}	attributes = xx00 xxxx
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information
:	
:	
:	

Figure 6.24 – Component Entry Descriptor

Note that currently, Component objects do not have child lists. If future system definitions do support child lists of Component objects, then the *has_child_ID* attribute bit will be set accordingly.

The *size_of_entry_specific_information* field specifies the number of bytes used for the following *entry_specific_information* field. The size field is two bytes, and is NOT included in this calculation.

The *entry_specific_information* field of the Tuner Subunit Component Entry Descriptor shares the same structure for all broadcast systems:

Address Offset	msb							lsb
00_{16}	system_id							
01_{16}	system_specific_component_selection							
:								
:								
:	system_specific_component_information							
:								
:								

Figure 6.25 – Component Entry Descriptor *entry_specific_information*

The fields for this common structure are very similar to the Multiplex entry descriptor above, but Component entry descriptors do not have (or need) the input and antenna field. The reason for this is that the selection process for a given type of object (Multiplex, Service, or Component) requires the specification of the appropriate objects higher in the hierarchy.

The *system_id* field has the same meaning as described in Multiplex Entry Descriptor.

The *system_specific_component_selection* field specifies the selection attributes that uniquely identify a Component and the *system_specific_component_information* field specifies the information attributes that describe the characteristics of the Component, but are not used for selection purpose. Please see section 7.5 Component Selection and Information Fields on page 62 for detailed descriptions.

6.9 Tuner Preferred Components List Descriptor

The Preferred Components List Descriptor is a root list descriptor. There is only one Preferred Components List Descriptor in a Tuner Subunit.

The preferred components object allows the specification of a collection of audio and subtitling component references to be used when selecting a Service. This object is normally used as part of a Service selection action, either using the OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE commands. A reference to a preferred components object may also be included in a preset object entry.

A preferred components list can specify several preferred components objects. One practical use of this list is to support such preferred components for many users; there could be one or more entries per user.

The Preferred Components List Descriptor is as follows:

Address	Contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	list_type = 90_{16}
00 03_{16}	attributes = xx01 xxxx
00 04_{16}	size_of_list_specific_information
00 05_{16}	
00 06_{16}	number_of_entries (n)
00 07_{16}	
00 08_{16}	object_entry[0]
:	
:	
:	:
:	object_entry[n-1]
:	
:	

Figure 6.26 – Preferred Components List Descriptor

In the *attributes* for the Preferred Components List, the bit which indicates *has_child_ID* is for object entries only, so it is required to be set to 0 for all lists. Currently, there is only one byte defined for list attributes, so the msb would be 0 also.

Note that currently there is no *list_specific_information* for the Preferred Components List, so the *size_of_list_specific_information* field shall be 00_{16}.

The *object_entry[x]* field is the Preferred Components Entry Descriptor described below.

Note that this type of list is required to have object IDs for its objects.

6.10 Tuner Preferred Components Entry Descriptor

A Tuner Subunit manufacturer may have a default Preferred Components Entry Descriptor that matches the regional variation of the implementation for preferred audio and subtitling languages.

In addition, a Tuner Subunit may optionally allow controllers to create Preferred Components Entry Descriptors to suit the users' preferences.

The format of the preferred components object entry is as follows:

address offset	Contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 90_{16}
03_{16}	attributes = xx00 xxxx
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information
:	
:	

Figure 6.27 – Preferred Components Entry Descriptor

The *attributes* of the Component entry specify that it has no *has_child_ID*.

The *size_of_entry_specific_information* field specifies the number of bytes used for the following *entry_specific_information* field. The size field is two bytes, and is NOT included in this calculation.

The *entry_specific_information* field of a preferred components entry descriptor is a preferred components descriptor, which appears as follows:

address offset	Contents
00_{16}	preferred_components_name_length
01_{16}	preferred_components_name
:	
:	
:	number_of_audio_language_preferences (n)
:	audio_language_preference[0]
:	
:	
:	:
:	audio_language_preference[n - 1]
:	
:	
:	number_of_subtitling_language_preferences (m)
:	subtitling_language_preference[0]
:	
:	
:	:
:	subtitling_language_preference[m – 1]
:	
:	

Figure 6.28 – Preferred Components Entry Descriptor *entry_specific_information*

The *preferred_components_name_length* field specifies the number of bytes in the following *preferred_components_name* field.

The *preferred_components_name* fields specify the name of this object. If there is no name, then the length field shall have the value zero and the name field shall not exist. One use of this field could be to allow the person who created these preferences object to give it a meaningful name for future use when setting up selection presets.

The *number_of_audio_language_preferences* field indicates how many audio language preferences are in this entry.

The *audio_language_preference[x]* field holds the specifier for this language preference.

The audio language preference specifiers are arranged in priority order, with the first entry having the highest priority. If the audio language component specified by item [0] exists, then that audio component shall be selected. If it does not exist, then the tuner shall look for audio language item [1], and so on. If none of the preferred audio components can be found, then the results are defined by the tuner implementation.

The *number_of_subtitling_language_preferences* and *subtitling_language_preference[x]* fields are used in the same manner as the audio language fields described above.

The audio and subtitling specifiers are 24-bit values. For AV/C tuner model, audio and subtitling component references are specified the same way for all broadcast systems, using the

ISO_639_language_code definitions referred to by the Digital Video Broadcast system. For details, please refer to the appropriate AV/C Tuner Broadcast System Specification document.

6.11 Tuner Preset List Descriptor

The Preset List Descriptor is a root list descriptor.

The preset object defines a preset combination of Components, a Service, or a Multiplex, to select. The preset entries contain detailed specifications of the items to be selected. The preset entry can include a reference to a preferred components object.

A preset list is primarily updated by a controller. However, it is allowed and even advised for the tuner to update the entries such that they contain the most recent information regarding the Multiplexes, Services and Components.

The Preset List Descriptor is as follows:

address	Contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	list_type = $A0_{16}$
00 03_{16}	attributes = xx0x xxxx
00 04_{16}	size_of_list_specific_information
00 05_{16}	
00 06_{16}	list_specific_information
:	
:	
:	number_of_entries (n)
:	
:	object_entry[0]
:	
:	
:	:
:	object_entry[n-1]
:	
:	

Figure 6.29 – Preset List Descriptor

In the *attributes* for this Preset List, the bit which indicates *has_child_ID* is for object entries only, so it is required to be set to 0 for all lists. Currently, there is only one byte defined for list attributes, so the msb would be 0 also. Supporting object IDs in this list is optional; the *has_object_ID* flag shall be set accordingly.

The *list_specific_information* field of the preset list is currently defined as follows:

address offset	Contents
00_{16}	list_name_length
01_{16}	list_name
:	
:	

Figure 6.30 – Preset *list_specific_information*

The *list_name_length* field indicates the number of bytes used for the list_name field. One practical use of this field would be to hold the name of the user which defined this preset list. If this list does not have a name, then the length field shall be set to 0.

The *list_name* field contains the name of this preset list.

The *object_entry[x]* field is the Preset Entry Descriptor described below.

6.12 Tuner Preset Entry Descriptor

The format of the preset object entry is as follows:

address offset	Contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = <see description>
03_{16}	attributes = xx0x xxxx
:	object_ID (if specified by has_object_ID in the list attribute flags)
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information (depends on the value of entry_type)
:	
:	

Figure 6.31 – Preset Entry Descriptor

The *entry_type* can be a Multiplex, a Service, or a Service with specified components, as described in the DIRECT SELECT INFORMATION TYPE command.

The *attributes* of this preset entry specify that it has no has_*child_ID*.

The *size_of_entry_specific_information* field specifies the number of bytes used for the following *entry_specific_information* field. The size field is two bytes, and is NOT included in this calculation.

The *entry_specific_information* of a preset object depends on the *entry_type*. In addition to a preset name, it may specify a Multiplex, a Service, or it may specify a "Service with specified components."

The *entry_specific_information* field is defined as follows:

Address offset	Contents	
00_{16}	preset_name_length	
01_{16}	preset_name	
:		
:		
:	system_id	
:	input	antenna_number
:	system_specific_multiplex_selection_length	
:	system_specific_multiplex_selection	
:		
:		
:	dsit_selection_specification	
:		
:		
:	creator_specific_preset_info_length	
:	creator_specific_preset_info	
:		
:		

Figure 6.32 – Preset Entry Descriptor *entry_specific_information*

The *preset_name_length* field specifies the number of bytes used for the following *preset_name* field.

The *preset_name* field contains the name of this preset object. This could be used by a controller, for example, to associate the name of the television station with this entry.

When the preset object is specifying only a Multiplex (when entry_type = multiplex), the fields from *system_id* to *system_specific_multiplex_selection* have the same meaning as those used for the DIRECT SELECT INFORMATION TYPE command as described on page 64. In this case, the *dsit_selection_specification* field does not exist.

When the preset object is specifying either a Service or Service with specified components (based on the *entry_type* value), then the fields from *system_id* to *dsit_selection_specification* have the same meaning as those for the DIRECT SELECT INFORMATION TYPE command with a single *dsit_selection_specification* operand.

The *creator_specific_preset_info_length* and *creator_specific_preset_info* fields contain the length and data of information that is specific to the creator of this preset object. The creator may not necessarily be the vendor of the Tuner Subunit.

7. General Broadcast System Selections and Information Fields

This section describes the general data structure frame used by all Tuner Subunits to represent broadcast system selections and information fields.

7.1 Text Field Encoding

The format of all text fields in various read-only broadcast system selections and information fields which are maintained by the Tuner Subunit shall be defined according to the particular broadcast system being represented. The exception is for regional variations which are based on a given broadcast system.

7.2 Selection Attributes and Information Attributes

The term “Selection Attributes” or “Information Attributes” does not refer to the “*attribute*” field of any descriptor.

Selection Attributes and Information Attributes are the collection of fields that describe a specific type of tuner’s selections and information fields of a broadcast system, such as DVB.

The following definitions are useful when reading about Tuner Subunit objects:

Selection Attributes: When we discuss the attributes or parameters necessary to describe a given type of system, such as DVB, we define Selection Attributes to be those attributes that can be used to perform a tuning action. This tuning action may be a Multiplex selection, or the selection of a particular Service, Component, or data.

Mandatory Selection Attributes: A subset of the Selection Attributes; we define these attributes such that when specified, they are **guaranteed** to select a **specific** information instance. Selection using these attributes shall be supported by the tuner, and the selection operation shall not be rejected as long as the attributes are valid and other tuner-specific constraints are satisfied (such as having the necessary resources available to satisfy the selection request). These attributes are specified per broadcast system.

Information Attributes: These are attributes that are used for informational purposes only (such as the name of a Service), and are normally obtained from the air. These attributes are NOT used for selection purposes.

Attributes Valid Flags: These flags are a set of bits, which indicate the validity of Selection Attributes **and** Information Attributes within an entry descriptor. When the corresponding attribute is valid, then its valid flag is set to 1. When the corresponding attribute is not valid, then its valid flag is set to 0. Either way, the field for that attribute exists within the selections and information fields.

7.3 Multiplex Selection and Information Fields

The following figure describes the system specific Multiplex selection and information fields:

Address Offset	msb							lsb
	system_specific_multiplex_selection							
02_{16}	system_specific_multiplex_attributes_valid_flags							
03_{16}								
04_{16}	currently_ available	selected	reserved					
05_{16}	system_specific_multiplex_selection_attributes							
:								
:								
	system_specific_multiplex_information							
:	system_specific_multiplex_information_attributes							
:								
:								

Figure 7.1 – Multiplex Selection and Information Attributes

The *system_specific_multiplex_attributes_valid_flags* are defined per broadcast system, and indicate the validity of the entries in both of the following fields (selection and information attributes). Only the valid parameters shall be used by the tuner for selection purposes. The parameters whose valid flags are set to 0 shall not be used. The size of this field shall be two (2) bytes. When making a selection using *system_specific_multiplex_selection*, the validity flags of the information attributes are not used.

While the *system_specific_multiplex_attributes_valid_flags* are defined per *system_id*, all system definitions share one common flag as shown here:

Table 7.1 – General Multiplex Attribute Valid Flags

Flags	Meaning
1xxx xxxx (MSB)	*reserved_field* – Set to 0. The most significant bit of the Multiplex attributes valid flags indicates whether the reserved fields are present or not. When this flag is 1, then the fields exist in both the selection and information attributes fields. If the flag is 0, then they do not exist. (This flag is no longer used in version 2.0 and later versions)
xxxx xxxx (LSB)	All other flags are defined per *system_id*.

The *currently_available* flag indicates whether this Multiplex is actually available at this time. In some situations, it is possible that the Multiplex (or a certain part of it, such as a Service) may not be available even though it is selected. If this bit is set to 1, then it is available. This flag is used to report the status, and therefore, is not used for selections.

The *selected* flag indicates whether this Multiplex is currently selected or not. The value 1 means it is selected. This flag is used to report the status, and therefore, is not used for selections.

The *system_specific_multiplex_selection_attributes* field will contain the various attributes that specify a Multiplex in the given *system_id*. These attributes are used for selection purposes. If the input is via the demux destination plug of the Tuner Subunit, then this field shall be zeroes, and the validity flags that corresponding to the attributes shall not be set.

The *system_specific_multiplex_information_attributes* field will contain the various attributes that provide useful information about a Multiplex in the given *system_id*. These attributes are NOT used for selection purposes.

7.4 Service Selection and Information Fields

The following figure describes the system specific Service selection and information fields:

Address Offset	msb							lsb
	system_specific_service_selection							
01_{16}	system_specific_service_attributes_valid_flags							
02_{16}	currently_ available	reserved						
03_{16}	system_specific_service_selection_attributes							
:								
:								
	system_specific_service_information							
:	system_specific_service_information_attributes							
:								
:								

Figure 7.2 – Service Selection and Information Attributes

The *system_specific_service_attributes_valid_flags* are defined per broadcast system, and indicate the validity of the entries in both of the following fields (selection and information attributes). The size of this field is broadcast system specific. The size of this field shall be one (1) byte. When making a selection using the *system_specific_service_selection*, the validity flags of the information attributes are not used.

While the *system_specific_service_attributes_valid_flags* are defined per *system_id*, all system definitions share one common flag as shown here:

Table 7.2 – General Service Attribute Valid Flags

Flags	Meaning
1xxx xxxx (MSB)	*Reserved_field* – Set to 0. The most significant bit of the Service attributes valid flags indicates whether the reserved fields are present or not. When this flag is 1, then the fields exist in both the selection and information attributes fields. If the flag is 0, then they do not exist. (This flag is no longer used in version 2.0 and later versions)
xxxx xxxx (LSB)	All other flags are defined per *system_id*.

The *currently_available* flag indicates whether this Service is actually available at this time. In some situations, it is possible that the Service (or a certain part of it, such as a Component) may not be available even though it is selected. If this bit is set to 1, then it is available. This flag is used to report the status, and therefore, is not used for selections.

The *system_specific_service_selection_attributes* field will contain the various attributes that specify a Service in the given *system_id*. These attributes are used for selection purposes. If the input is via the demux destination plug of the Tuner Subunit, then this field shall be zeroes, and the validity flags that corresponding to the attributes shall not be set.

The *system_specific_service_information_attributes* field will contain the various attributes that provide useful information about a Service in the given *system_id*. These attributes are NOT used for selection purposes.

7.5 Component Selection and Information Fields

The following figure describes the system specific Component selection and information fields:

Address Offset	msb							lsb
	system_specific_component_selection							
01_{16}	system_specific_component_attributes_valid_flags							
02_{16}	currently_ available	reserved						
03_{16}	system_specific_component_selection_attributes							
:								
:								
	system_specific_component_information							
:	system_specific_component_information_attributes							
:								
:								

Figure 7.3 – Component Selection and Information Attributes

The *system_specific_component_attributes_valid_flags* are defined per broadcast system, and indicate the validity of the entries in both of the following fields (selection and information attributes). The size of this field is broadcast system specific. The size of this field shall be one (1) byte. When making a selection using the *system_specific_component_selection*, the validity flags of the information attributes are not used.

While the *system_specific_component_attributes_valid_flags* are defined per *system_id*, all system definitions share one common flag as shown here:

Table 7.3 – General Component Attribute Valid Flags

Flags	Meaning
1xxx xxxx (MSB)	*Reserved_field* – Set to 0. The most significant bit of the Component attributes valid flags indicates whether the reserved fields are present or not. When this flag is 1, then the fields exist in both the selection and information attributes fields. If the flag is 0, then they do not exist. (This flag is no longer used in version 2.0 and later versions)
xxxx xxxx (LSB)	All other flags are defined per *system_id*.

The *currently_available* flag indicates whether this Component is actually available at this time. In some situations, it is possible that the Component may not be available even though it is selected. If this bit is set to 1, then it is available. This flag is used to report the status, and therefore, is not used for selections.

The *system_specific_component_selection_attributes* field will contain the various attributes that specify a Component in the given *system_id*. These attributes are used for selection purposes. If the input is via the demux destination plug of the Tuner Subunit, then this field shall be zeroes, and the validity flags that corresponding to the attributes shall not be set.

The *system_specific_component_information_attributes* field will contain the various attributes that provide useful information about a Component in the given *system_id*. These attributes are NOT used for selection purposes.

8. Tuner Subunit Commands

Tuner Subunit commands are identified by a *subunit_type* value of 05_{16}. This section describes the commands that are defined specifically for Tuner Subunits. This section also describes how some of the common subunit commands are applied to the Tuner Subunits. Please refer to the section titled Tuner Subunit Profiles on page 85 for a detailed table which illustrates various commands and how they apply to certain kinds of Tuner Subunits.

Selection commands in which operands are specified with incorrect values will result in a REJECTED or NOT IMPLEMENTED response. However, there may be cases in which the specified values are correct, but the tuner cannot perform a straightforward selection because:

1) Incomplete or non-unique specifications (more than 1 selection satisfying the specified selection attributes is possible)

2) Contradictory specifications (a selection is not possible unless one or more specified selection attributes are ignored)

For both cases, a Tuner Subunit shall return an ACCEPTED response with an appropriate indication in the response frame of the selection command, and shall perform only one of the above mentioned possible selections. The choice of which item to select is up to the tuner implementation.

Table 8.1 – Summary of Tuner Subunit Commands

Opcode	Value	Support level (by ctype)			Comments
		C	S	N	
DIRECT SELECT INFORMATION TYPE	$C8_{16}$	O	-	-	Select information instances using detailed specifications
DIRECT SELECT DATA	CB_{16}	O	-	-	Select data (not data Components) from an entire stream
CA ENABLE	CC_{16}	O	O	O	Enable the conditional access system
TUNER STATUS	CD_{16}	-	-	O	Notify controllers when the Tuner Status Descriptor changes

8.1 DIRECT SELECT INFORMATION TYPE (DSIT)

For Tuner Subunits, the DIRECT SELECT INFORMATION TYPE command deals with the information types of *Multiplex*, *Service*, and *Component*. It is NOT used to select the *data* information type.

The selection is performed by providing a detailed specification; this command does not deal with objects and object lists. For the Tuner Subunit, the DIRECT SELECT INFORMATION TYPE control command has three functions:

1) Select the Multiplex for the Tuner Subunit, and the source plug to receive that Multiplex; the input will be one of the antennas on the receiver unit, an input stream from 1394, an external (non-

antenna) input, or an input from another subunit in the receiver unit. The subunit destination plug to receive the signal can be either the antenna destination plug or the demux destination plug.

2) Modify the *currently selected information instances*. It does this in two ways:

   a) by selecting one or more information instances from the *available information instances* and putting it into the *currently selected information instances*

   b) by removing one or more elements from the *currently selected information instances*

3) Modify the output of the Tuner Subunit source plugs. It does this in two ways:

   a) by establishing an output flow of one or more of the *currently selected information instances* to a subunit source plug

   b) by removing one or more information instances from a subunit source plug

The information instances that are selected for addition or removal are specified by *dsit_selection_specifications*. The format of the DIRECT SELECT INFORMATION TYPE control command for the Tuner Subunit is illustrated by the figure below:

	msb							lsb
Opcode	DIRECT SELECT INFORMATION TYPE ($C8_{16}$)							
operand[0]	source_plug							
operand[1]	subfunction							
operand[2]	status							
operand[3]	system_id							
operand[4]	input	antenna_number						
operand[5]	system_specific_search_flags							
operand[6]	system_specific_multiplex_selection_length							
operand[7]	system_specific_multiplex_selection							
:								
:								
:	number_of_dsit_selection_specifications (n)							
:	dsit_selection_specification[0]							
:								
:								
:	:							
:	:							
:	:							
:	dsit_selection_specification[n - 1]							
:								
:								

Figure 8.1 – DIRECT SELECT INFORMATION TYPE command

The *source_plug* field indicates the subunit source plug number for output, and can be any value as specified in the table of subunit source plugs in the description of the CONNECT command in Reference [R3].

The *subfunction* field specifies the operation of the control command and is encoded according to the table of subfunctions specified for the OBJECT NUMBER SELECT command, as documented in the Reference [R3]. However, for the DIRECT SELECT INFORMATION TYPE command, the tuner model does not use the "non-output" plug value FE_{16}.

The following rules shall be adhered to when implementing the subfunctions:

> Generating a mix of information instances (with replace, append, etc.) shall be REJECTED if all objects can not be found on the same transponder **or** if they can not be found on the same antenna.
>
> The remove subfunction shall be REJECTED if the specified information instances are not present on the specified source plug.
>
> A service selection from the demux destination plug (*input* = 1) shall ignore any specified *system_specific_multiplex_selection_attributes*. These attributes are defined for each broadcast system in the appropriate AV/C Tuner Broadcast System Specification document.

The rule about ignoring Multiplex selection attributes when selecting from the demux destination plug exists because the demux input of the Tuner Subunit is NOT used for tuning operations; it is only for demuxing operations. The Multiplex selection attributes are used for tuning operations, and therefore apply only when selecting via the antenna destination plug.

The *status* field shall be set to FF_{16} on input of the control command. In the ACCEPTED response frame, this field shall be updated with the appropriate value as defined here:

Table 8.2 – DSIT *status* definition

status value	Meaning
00_{16}	The selection specification indicated a unique item, which was selected.
01_{16}	The selection specification was ambiguous, so the subunit selected one. The controller should examine the tuner status descriptor to determine exactly what is on the plug.
all others	reserved for future specification

For all responses, except NOT IMPLEMENTED response, the response frame shall consist of only the first three fields (up to the *status* field). All other fields of the control command frame shall not be returned. In case of the INTERIM or REJECTED responses, the contents of the *status* field shall be set to FF_{16}.

The *system_id* field identifies the type of system (e.g. DVB, analog video) from which this selection is being made. The values are defined in the table of *system_id* values presented in the section titled Tuner Subunit Identifier Descriptor on page 31.

The *input* flag and *antenna_number* field are described in Multiplex Selection and *Information* Fields on page 59.

The *system_specific_search_flags* field is defined by the broadcast system specification and allows the controller to specify a search action to be taken during the selection. The flags should be defined in a way that there is a collection of selection attributes used for the search and they correspond to the search flags specified. To perform a search, the following three conditions must be met:

1) Only one of the search flags is allowed to be set (equal to 1)
2) The DSIT subfunction is *append* , *replace*, or *new*
3) There is at most one *dsit_selection_specification* in the DIRECT SELECT INFORMATION TYPE command

In this case, the selection attribute for which the search flag is set will be used as a "wild card", and the tuner will start searching for a "next" value for this selection attribute. The searching starts at the specified value of the corresponding selection attribute of the *system_specific_multiplex_selection* or the *dsit_selection_specification*. If the *system_specific_multiplex_selection* is not specified (the *system_specific_multiplex_selection_length* = 0), the current Multiplex is assumed. If the *dsit_selection_specification* is not specified (the *number_of_dsit_selection_specifications* = 0), the current Service is assumed. The combination of the *system_specific_multiplex_selection* and the *dsit_selection_specification* form the starting point of the search. The end point of the search depends on the settings of the search flag. When the searching is ultimately successful, the DIRECT SELECT INFORMATION TYPE control command ends by selection of the found Information Instance. The found Information Instance has to satisfy the other specified selection attributes as in the non-searching case. Those attributes whose corresponding valid flags are set to 1 shall be satisfied.

When performing a search, the subunit may return an initial response of INTERIM because the search might take some time. If the requested information instances are found, then the subunit shall return a response of ACCEPTED, otherwise it shall return REJECTED.

After an INTERIM response has been sent, a controller may issue the DIRECT SELECT INFORMATION TYPE command specifying a value of FF_{16} for the *system_specific_search_flags* operand. In this case, the current search operation will be terminated. Upon receiving this command, the Tuner Subunit shall stop searching for the specified selection, and find the nearest valid item which corresponds to the search flag which was set for the previous search operation. For example, if the tuner had been searching for a specified frequency and was then told to stop searching, it will stop at the nearest valid frequency it can find. The meaning of "nearest" is defined by the target subunit.

The *system_specific_multiplex_selection_length* field specifies the number of bytes for the *system_specific_multiplex_selection* field, which is described in detail in section 7.3 Multiplex Selection and Information Fields on page 59.

If the *system_specific_multiplex_selection* is not specified (the *system_specific_multiplex_selection_length* = 0), or when the *system_specific_multiplex_attributes_valid_flags* are all zeroes, then the currently selected Multiplex shall be used to get the specified data. In this case, the *system_specific_multiplex_selection_attributes* field still exists in the command structure. Furthermore, the *system_id*, *input*, and *antenna_number* fields will be ignored.

The *number_of_dsit_selection_specifications* operand shows the number of tuner selection specifiers that are in the parameter block. To select a Multiplex, a Service, or a Component, the *system_specific_multiplex_selection_attributes* must specify the selection criteria for the Multiplex according to the broadcast technology. The presence of other selection information (Service and Component) will depend on what is being selected. In order to select a complete Multiplex, the value of *number_of_dsit_selection_specifications* shall be zero and there shall be no *dsit_selection_specification[x]* operands.

> The *dsit_selection_specification[x]* operands contain precise descriptions of the information types, and therefore information instances, to be tuned and output. These descriptions will be specific to a given information type for a given broadcast system, so details are presented in the appropriate AV/C Tuner Broadcast System Specification document.

The general format of this operand is as follows:

Address offset	Contents
00_{16}	specification_length
01_{16}	information_type
02_{16}	info_type_selection_specification
:	
:	

Figure 8.2 – *dsit_selection_specification*

The *specification_length* field contains the number of bytes which follow in this structure. The value of this field does *not* include the length field itself.

The *information_type* field specifies which of the tuner information type is defined by the *info_type_selection_specification* field. The values are defined as follows:

Table 8.3 – *information_type*

Information_type	Value
Service	82_{16}
Service with specified Components	83_{16}
Reserved	all others

The *info_type_selection_specification* field defines the scope of the item(s) to be selected. Depending on the particular broadcast technology and the capabilities of the Tuner Subunit, this scope can be one or more complete Services within a Multiplex, or a combination of Components from a Service.

All specifiers (Service and Component) within a *dsit_selection_specification* structure must be in the same Multiplex of the same broadcast system. It does not make sense to specify an analog video Multiplex specifier and a DVB Service specifier. If the specifications do not match, or if they fail to resolve to an available selection, then the command shall be REJECTED.

8.1.1 Selecting a Complete Service

In order to select a Service using the detailed specifications of the DIRECT SELECT INFORMATION TYPE command, it is necessary to resolve the scope down from the Multiplex to the Service level. This eliminates any ambiguity that may arise when Service specifications might be the same from one Multiplex to another.

The *dsit_selection_specification* causes the tuner to construct a partial transport stream from the desired selection criteria, and to place it on the source plug specified in the command using the specified subfunction. The specification for selecting a complete Service is as follows:

address offset	Contents
00_{16}	specification_length
01_{16}	information_type = service (82_{16})
02_{16}	system_specific_service_selection
:	
:	
:	
:	

Figure 8.3 – *dsit_selection_specification* for a complete Service

The *specification_length* field contains the number of bytes which follow in this structure. The value of this field does *not* include the length field itself.

The *information_type* field defines what kind of entity is being selected. This field indicates the structure of the entire *dsit_selection_specification*. The values for this field are defined in Table 8.3.

The *system_specific_service_selection* is described in details in section 7.4 Service Selection and Information Fields on page 61.

To select more than one Service, it is necessary to specify multiple *dsit_selection_specification* structures within the DIRECT SELECT INFORMATION TYPE command.

8.1.2 Selecting a Service with Specified Components

Instead of selecting a complete Service, it is also possible to select a Service by specifying a subset of the Components of that Service. At times this may be required because in most cases, especially for digital broadcast technologies, there may be many Components which can conflict with each other and don't make sense for the user. One example is the audio language components; when watching a broadcast, a user will normally only want to hear one language component (such as English or Japanese). However, there may be many audio components available from the broadcaster for this Service, so most of them will need to be filtered out. Other examples may include the closed caption text (whether the user wants to see it or not), or any other auxiliary components.

When recording a broadcast, it may be desirable to select some or all of these "conflicting" Components, because they are then available for selection by the user when it is played back.

When selecting a Service with specified Components, we specify the Multiplex, the Service, and the individual Components which compose the desired output Service. The general format of a selection specification for a Service with specified Components is as follows:

Address offset	Contents
00_{16}	specification_length
01_{16}	Information_type = service with specified components (83_{16})
02_{16}	system_specific_service_selection
:	
:	
:	**number_of_components (m) <> FF_{16}**
:	system_specific_component_selection [0]
:	
:	
:	system_specific_component_selection [m-1]
:	
:	

Figure 8.4 – *dsit_selection_specification* for a Service with specified Components

All of the fields down to *system_specific_service_selection* are as described above for Service selection.

The *number_of_components* field specifies how many Component selection specifiers follow in this structure. For the selection of a Service using specified Components, this field shall NOT be set to FF_{16}. When this field is set to 0, then there shall be no *system_specific_component_selection* fields included in the specification.

The *system_specific_component_selection* is described in details in section 7.5 Component Selection and Information Fields on page 62.

8.1.3 Selecting a Service Using Preferred Components

The tuner model defines an object of type *preferred_components*, and a list descriptor to hold several of these objects. The purpose is to define what is essentially a “global” language preference for audio and subtitling component settings, which can be applied to any Service selection. This allows a user to define a set of preferences (“I always want the English audio, and English subtitling”) only once, and the tuner to easily configure any Service selection accordingly.

The list of these preferred objects allows several different preference configurations to be stored for use in different situations by different users. When a Service is selected using the preferences, only one preferred components object is specified, which is then used as a “filter” on the Service specification.

The general format of a Service selection specification using preferred components is as follows:

address offset	Contents
00_{16}	specification_length
01_{16}	information_type = service with specified components (83_{16})
02_{16}	system_specific_service_selection
:	
:	
:	**number_of_components (m) = FF_{16}**
:	selection_indicator
:	preferred_components_object_reference
:	
:	

Figure 8.5 – *dsit_selection_specification* for a Service with preferred Components

All of the fields down to *system_specific_service_selection* are as described above for Service selection.

Note that when we are selecting a Service using preferred Components, we still use the *information_type* "Service with specified Components", which indicates that the format of the selection specification structure will specify which Components to use. In this case, we are referencing a preferred Components object.

The *number_of_components* field specifies how many component selection specifiers follow in this structure. For the selection of a Service using preferred Components, this field SHALL be set to FF_{16}.

The *dsit_selection_specification* for a Service with preferred Components is the only *dsit_selection_specification* which includes both detailed selection attributes (for the Multiplex and Service) and an object reference (for the preferred components object).

The *specifier_type_flag* of the *selection_indicator*, both described in the ONS command in reference [R3], if set to 1, indicates that the preferred components object is referenced by its ID. If it is zero, then the preferred components object is referenced by its position in the preferred components list. When using a position reference, the object reference is 2 bytes, as indicated by *size_of_object_position*. When using an object ID, the number of bytes to use is indicated in the *size_of_object_ID* field of the subunit identifier descriptor.

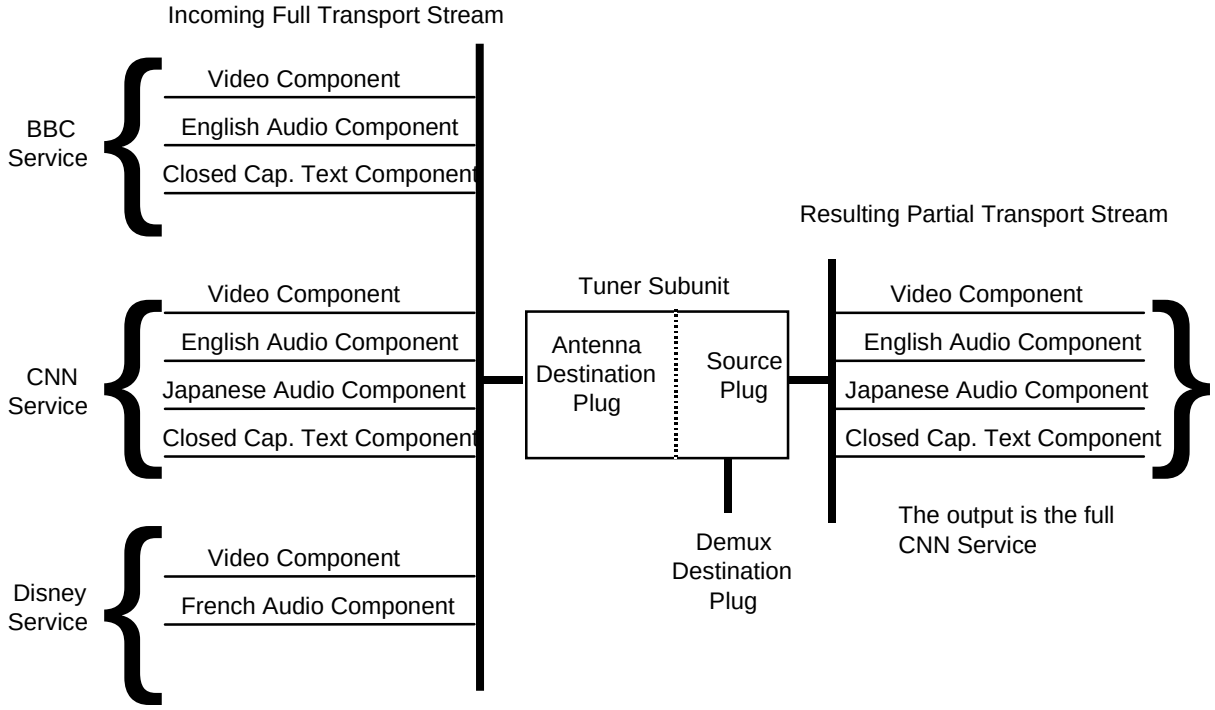
The *preferred_components_object_reference* field contains the reference to the preferred components object. The format of this field is as described above.

8.1.4 Service Construction (Informative)

When the DSIT command is issued to a DVB Tuner Subunit, the resulting output stream will either be a full transport stream (if a Multiplex was selected) or a partial transport stream (in all other cases). When it is a partial transport stream, then the selection process results in the creation of one or more "Services". In this case, a Service may either be a complete Service with all of its Components, or it may be a Service constructed from a subset of its Components. In the latter case, the result is still referred to as a Service.

In all partial transport stream cases, a manufacturer is required to maintain the transport stream requirements specified by the broadcast system standard. These requirements imply that in some broadcast systems, the Tuner Subunit needs to re-construct the selected Services to make the partial transport stream looks like a full transport stream.

The way that these Services are constructed is directly affected by the way selection information is presented to the tuner in the parameters of the DSIT command. Consider the following diagram, which shows a Service being selected from the incoming transport stream:

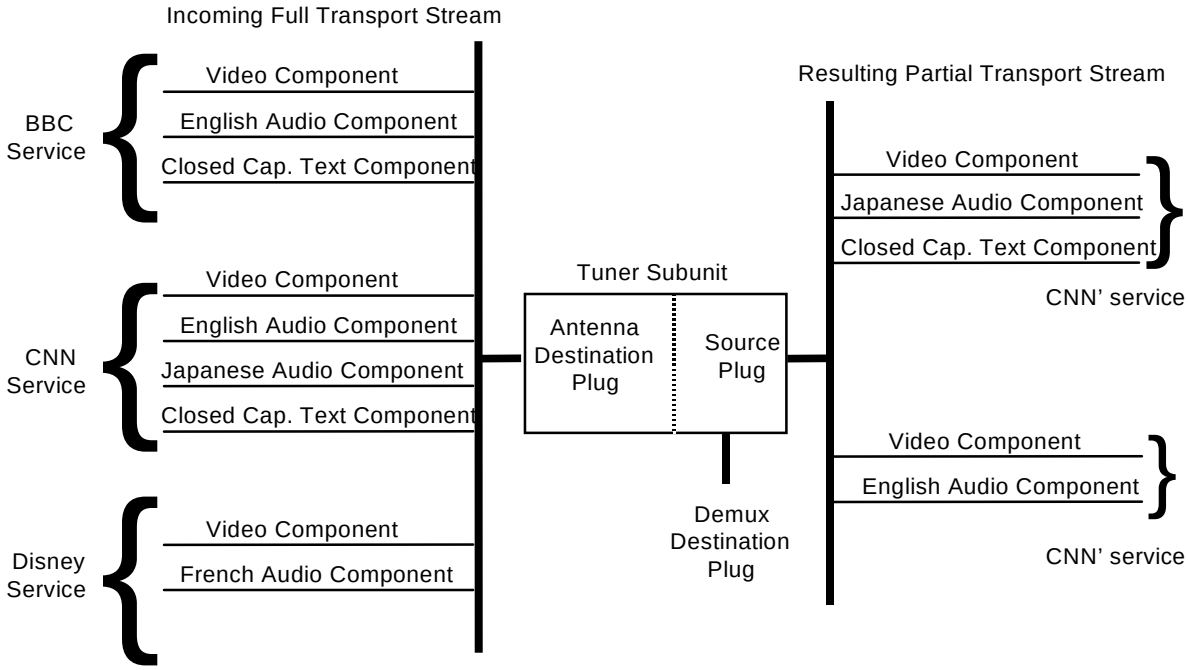


DIRECT SELECT INFORMATION TYPE
source_plug
• • •
system_specific_multiplex_selection_attributes
number_of_dsit_selection_specifications (n) = 1
dsit_selection_specification[0] for the CNN service

Figure 8.6 – Service construction of a complete Service

In the example above, the CNN Service was specified in a single dsit_selection_specification parameter, and the resulting output stream contains the full CNN Service with all of its Components.

Now consider the following diagram:



DIRECT SELECT INFORMATION TYPE
source_plug
• • •
system_specific_multiplex_selection_attributes
number_of_dsit_selection_specifications (n) = 2
dsit_selection_specification[0] for the CNN service with Video, Japanese Audio and CC Text components
dsit_selection_specification[1] for the CNN service with Video and English Audio components

Figure 8.7 – Service construction of multiple Services

In the case of digital systems such as DVB or DAB, in which Services and Components are multiplexed, the output stream can take on two different forms as shown in the two previous diagrams. The second diagram illustrates the *creation* of two separate Services using various Components from an incoming Service. This is implemented by specifying the two desired output Services as two separate *dsit_selection_specification* parameters.

8.2 OBJECT NUMBER SELECT (ONS)

For Tuner Subunits, the OBJECT NUMBER SELECT command deals with the defined information types (Multiplex, Service, and Component). In addition, it deals with the preset lists. For details on the general OBJECT NUMBER SELECT command, please refer to reference [R3].

8.2.1 Selection Using Specified Children

For the Tuner Subunit, the selection of a target by specifying one or more of its child elements is only valid for Service selection (by specifying one or more of its Components). It is not valid to select a Multiplex by specifying one or more of its Services. Please refer to the general description of the OBJECT NUMBER SELECT command for details on this concept.

8.2.2 The Tuner Subunit *ons_selection_specification* Structure

In addition to the general *ons_selection_specification* formats defined by reference [R3], Tuner Subunits support an additional *ons_selection_specification*, which allows the controller to specify a selection using a preferred Components object. All of the fields are as described for the general *ons_selection_specification* in the OBJECT NUMBER SELECT command description.

Supporting the preferred Components list is optional. If the list is not implemented, the response shall be NOT IMPLEMENTED. If the specified object is not in the list, then the command shall be REJECTED.

8.2.2.1 ONS full path specification

The *target* field of the *ons_selection_specification* of a full path specification is specified as follows:

Address offset	Contents
00_{16}	target_object_reference
:	
:	
:	number_of_children = FF_{16}
:	preferred_components_object_reference
:	
:	
:	

Figure 8.8 – *target* (full path specification) with preferred Components descriptor reference

In this case, the *target_object_reference* indicates a Service object which is being selected with the specified preferred Components object indicated by the *preferred_components_object_reference* field.

The Tuner Subunit defines a special meaning when *number_of_children* = FF_{16}, which is to perform the object selection using the specified preferred Components object. Note that the *target_format_flag* in the *selection_indicator* field in the command frame must be set to 1, indicating that the selection is using specified Components. In this case, we specify those Components using a preferred Components object reference.

The *preferred_components_object_reference* indicates which preferred components object to use as a "filter" when selecting the target object specified by *target_object_reference* either by object position or by object ID.

8.2.2.2 ONS "don't care" path specification

The following diagrams illustrate the "don't care" *ons_selection_specification* using preferred components, one using object position references and the other using object ID references (based on the *specifier_type_flag* of the *selection_indicator*).

When the *specifier_type_flag* is zero, then the *target* field shall have the following format:

address offset	Contents
00_{16}	list_ID (MSB)
01_{16}	list_ID (LSB)
02_{16}	target_object_reference
03_{16}	(object_position)
04_{16}	number_of_children = FF_{16}
05_{16}	preferred_components_object_reference
06_{16}	(object_position)

Figure 8.9 – *target* ("don't care" specification) referenced by position with preferred components

When the *specifier_type_flag* is one, then the target field shall have the following format:

address offset	Contents
00_{16}	list_type
01_{16}	target_object_reference
:	
:	(object_ID)
:	number_of_children = FF_{16}
:	preferred_components_object_reference
:	
:	(object_ID)

Figure 8.10 – *target* ("don't care" specification) referenced by ID with preferred components

8.2.3 Service Construction

The same concepts of Service construction apply to OBJECT NUMBER SELECT as they do for DIRECT SELECT INFORMATION TYPE. For more details, please refer to the DSIT command explanation.

8.2.4 Subfunction Implementation Rules

The subfunctions specified by Tuner Subunits for OBJECT NUMBER SELECT are the same as those described in the subfunction table for the general AV/C OBJECT NUMBER SELECT command, EXCEPT for the actions for source plug FE_{16}; the tuner model does not use the non-output signal features of plug FE_{16}.

The implementation rules for the Tuner Subunit OBJECT NUMBER SELECT subfunctions shall be the same as those specified for the tuner-specific DIRECT SELECT INFORMATION TYPE command. These rules are a superset of those specified by the general OBJECT NUMBER SELECT command.

8.2.5 Status Response

The *status* field in the response frame of the OBJECT NUMBER SELECT status command for Tuner Subunits may have one of the following values:

Table 8.4 – ONS *status* definition

value	status description
00_{16}	No information instances, that were selected by ONS commands, are on the specified source plug.
10_{16}	The information instances, that were selected by ONS commands, listed in the following fields are currently on the specified source plug.
20_{16}	The information instances. that were selected by ONS commands, listed in the following fields are supposed to be on the specified source plug, however some of them are currently not on the plug because the information instances are not available (examine the *currently_available* flags in the *data_status* and *info_type_status* fields).

OBJECT NUMBER SELECT status response reports the *ons_selection_specifications* of the currently selected information instances, on the specified tuner subunit source plug, that were selected by previous OBJECT NUMBER SELECT control commands. The information instances that were not selected by OBJECT NUMBER SELECT control commands, such as DIRECT SELECT INFORMATION TYPE commands, will not be reported by OBJECT NUMBER SELECT status response. The controller should examine the tuner status descriptor to get this information.

8.2.6 Summary of Tuner ONS *ons_selection_specification* operands

The following tables summarize the ons_selection_specification operands used in various Tuner selection processes:

Table 8.5 – Tuner ONS Command ons_selection_specification Operands - 1

ONS Operands	Select a Multiplex				Select a Complete Service			
Selection Specifier	Full Path		Don't Care		Full Path		Don't Care	
Root List ID	Multiplex List ID		Multiplex List ID		Multiplex List ID		Multiplex List ID	
Selection Indicator	Position, no children	ID, no children	Position, no children	ID, no children	Position, no children	ID, no children	Position, no children	ID, no children
Target Depth	0	0	FF	FF	1	1	FF	FF
Path specifier					MPX Entry Position	MPX Entry ID		
Target	Multiplex Entry Object Position	Multiplex Entry Object ID	Multiplex List Type	Multiplex List ID	Service Entry Object Position	Service Entry Object ID	Service List ID	Service List Type
			Multiplex Entry Object Position	Multiplex Entry Object ID			Service Entry Object Position	Service Entry Object ID

Table 8.6 – Tuner ONS Command ons_selection_specification Operands - 2

ONS Operands	Select a Service with Selected Components				Select a Service with Preferred Components			
Selection Specifier	Full Path		Don't Care		Full Path		Don't Care	
Root List ID	Multiplex List ID		Multiplex List ID		Multiplex List ID		Multiplex List ID	
Selection Indicator	Position, with children	ID, with children	Position, with children	ID, with children	Position, with children	ID, with children	Position, with children	ID, with children
Target Depth	1	1	FF	FF	1	1	FF	FF
Path specifier	MPX Entry Position	MPX Entry ID			MPX Entry Position	MPX Entry ID		
Target	Service Entry Object Position	Service Entry Object ID	Service List ID	Service List Type	Service Entry Object Position	Service Entry Object ID	Service List ID	Service List Type
			Service Entry Object Position	Service Entry Object ID			Service Entry Object Position	Service Entry Object ID
	N	N	N	N	FF	FF	FF	FF
	Component Position 1	Component ID 1	Component Position 1	Component ID 1	Preferred Component Object Position	Preferred Component Object ID	Preferred Component Object Position	Preferred Component Object ID
	:	:	:	:				
	Component Position N	Component ID N	Component Position N	Component ID N				

Table 8.7 – Tuner ONS Command ons_selection_specification Operands - 3

ONS Operands	Select a Service with Preset Entry				Select a Service with Preset Entry and with preferred Components			
Selection Specifier	Full Path		Don't Care		Full Path		Don't Care	
Root List ID	Preset List ID		Preset List ID		Preset List ID		Preset List ID	
Selection Indicator	Position, no children	ID, no children	Position, no children	ID, no children	Position, with children	ID, with children	Position, with children	ID, with children
Target Depth	0	0	FF	FF	0	0	FF	FF
Path specifier								
Target	Preset Entry Object Position	Preset Entry Object ID	Preset List ID	Preset List Type	Preset Entry Object Position	Preset Entry Object ID	Preset List ID	Preset List Type
			Preset Entry Object Position	Preset Entry Object ID			Preset Entry Object Position	Preset Entry Object ID
					FF	FF	FF	FF
					Preferred Components Entry Position	Preferred Components Entry ID	Preferred Components Entry Position	Preferred Components Entry ID

8.3 DIRECT SELECT DATA (DSD)

The DIRECT SELECT DATA command deals only with the *data* information type, and selects from the specified Multiplex. Note that this *data* is NOT the same as a data Component, as illustrated in the example DVB diagram at the beginning of the section Broadcasting concepts and Service Delivery Model on page 12. The control command has the following format:

	msb							Lsb
Opcode	DIRECT SELECT DATA (CB_{16})							
Operand[0]	source_plug							
Operand[1]	subfunction							
operand[2]	system_id							
operand[3]	input	antenna_number						
operand[4]	system_specific_multiplex_selection_length							
operand[5]	system_specific_multiplex_selection							
:								
:								
:	number_of_dsd_selection_specifiers (n)							
:	flowfunction[0]							
:	dsd_selection_specification[0]							
:								
:								
:	:							
:	flowfunction[n - 1]							
:	dsd_selection_specification[n - 1]							
:								
:								

Figure 8.11 – DIRECT SELECT DATA command

The *source_plug*, *subfunction* and *system_id* through *system_specific_multiplex_selection* operands are as specified for the DIRECT SELECT INFORMATION TYPE command.

When the *system_specific_multiplex_attributes_valid_flags* are all zero, then specified data shall be selected from the currently selected Multiplex. In this case, the *system_specific_multiplex_selection_attributes* field still exists in the command structure. In addition, the *system_id*, *input*, and *antenna_number* fields will be ignored.

The *number_of_dsd_selection_specifiers* field indicates how many pairs of *flowfunction[x]* and *dsd_selection_specification[x]* operands are specified in this command.

The *flowfunction[x]* operands define how the specified data (indicated by the *dsd_selection_specification[x]* operand which follows) should be sent over the *source_plug*. Each of the data specifications in the command can have a different *flowfunction[x]*.

The *flowfunction[x]* allows the controller to maximize overall system performance by only having certain pieces of data sent at certain times. It is defined in the table below:

Table 8.8 – *flowfunction* definition

Value	flowfunction	Action
11_{16}	send	send the required data once, then stop sending
12_{16}	monitor	send once, thereafter send each new version once (Note: support for this flow function is optional)
13_{16}	relay	send every occurrence

The format and contents of the *dsd_selection_specification[x]* operands will be specific to each *system_id*. For details on the data specifications for each system, please refer to the appropriate AV/C Tuner Broadcast System Specification document.

To determine what data has been selected and routed to a particular Tuner Subunit source plug, the controller may examine the Tuner Subunit status descriptor. For details, please refer to the section titled Tuner Status Descriptorwhich begins on page 38.

8.4 CA ENABLE

Support for conditional access (CA) to protected (scrambled) broadcasts is provided through the CA ENABLE control command. At this time, there are no standard definitions for how CA will be implemented, because most systems are using Service provider-specific methods for scrambling Services. The CA ENABLE command has the following format:

address	Contents
opcode	CA ENABLE (CC_{16})
operand[0]	CA_system_specific_length (n)
operand[1]	CA_system_specific
:	
operand[n]	

Figure 8.12 – CA ENABLE command

The *CA_system_specific_length* field contains the number of bytes used for the following conditional access command block *CA_system_specific*. The value of this field does *not* include the length field itself.

The format and contents of the *CA_system_specific* field will depend on the particular CA system being used. This information is used to enable the CA descrambling system.

The CA ENABLE command may be used with a ctype of STATUS. The command has the following format:

	msb							lsb
Opcode	CA ENABLE (CC_{16})							
Operand[0]	FF_{16}							

Figure 8.13 – CA ENABLE status command

The STABLE response frame will have the same format as the control command frame.

The CA ENABLE command can also be used for notification, with a ctype of NOTIFY. The notify command has the following syntax:

	msb							lsb
Opcode	CA ENABLE (CC_{16})							
Operand[0]	FF_{16}							

Figure 8.14 – CA ENABLE notify command

When the tuner selects a Service for which descrambling is needed (such as an impulse pay per view), it will send the notification message. The notify response has the same format as the CA ENABLE control command. The *CA_system_specific* data will contain information used by the controller to establish access for the tuner to the scrambled signal via the CA ENABLE control command described above.

8.5 TUNER STATUS

The TUNER STATUS command allows a controller to request notification messages from the Tuner Subunit when the *tuner status descriptor* changes. This command is only defined for the NOTIFY ctype. Please refer to the definition of the tuner status descriptor on page 38 for details on what information is provided in that structure.

The format of the TUNER STATUS notify command is as follows:

	msb							lsb
Opcode	TUNER STATUS (CD_{16})							
Operand[0]	source_plug							
Operand[1]	FF_{16}							
Operand[2]	FF_{16}							

Figure 8.15 – TUNER STATUS notify command

The *source_plug* operand specifies the *source_plug_status[x]* entry in the tuner status descriptor for which the controller wants change notifications. This value is the same as the source plug number.

To be notified of changes in the general area, *general_tuner_status* field, of the tuner status descriptor, the controller can specify a value of FF_{16} for the *source_plug* operand.

In order to be notified of changes to several different plugs in addition to the general area, the controller may issue several TUNER STATUS commands to the Tuner Subunit, each one specifying the desired source plug (or the general area).

The CHANGED response frame has the same format as the NOTIFY command frame, indicating which *source_plug* (or the general area) has changed. It is up to the controller to examine the tuner status descriptor structure to determine exactly which fields have changed. In the response frame, operand[1] will contain the *system_id* of the broadcasting system which caused the change and operand[2] may contain a value not equal to FF_{16}, indicating the *system_specific_event*. For details on the system specific events, please refer to the system specific sections of the broadcast system specification.

The figure below illustrates the TUNER STATUS response frame with a *system_id* of the broadcast system and a *system_specific_event* defined in the broadcast system specification.

	msb							lsb
Opcode	TUNER STATUS (CD_{16})							
Operand[0]	source_plug							
Operand[1]	system_id							
Operand[2]	system_specific_event							

Figure 8.16 – TUNER STATUS response frame with defined *system_specific_event*

The figure below illustrates the TUNER STATUS response frame with a *system_id* of the broadcast system and FF_{16} in Operand[2] when the event is not defined in the broadcast system specification.

	msb							lsb
Opcode	TUNER STATUS (CD_{16})							
Operand[0]	source_plug							
Operand[1]	system_id							
Operand[2]	FF_{16}							

Figure 8.17 – TUNER STATUS response frame with not defined *system_specific_event*

9. Summary of Broadcast System Specific Fields

Each AV/C Tuner broadcast system specification must specify the value of or define the following fields:

- SID:*system_id*.
- SID:*implementation_profile_id*. (if defined)
- SID:*antenna[x]_specification:system_specific_antenna_range_specification:selection_attribute_range_specification.*
- SID:*system_specific_information*.
- SD:*general_tuner_status:antenna_input_info:antenna_general_system_info*.
- SD:*general_tuner_status:demux_input_info:demux_general_system_info*.
- Multiplex *system_specific_multiplex_selection* and *system_specific_multiplex_information*
- Service *system_specific_service_selection* and *system_specific_service_information*
- Component *system_specific_component_selection* and *system_specific_component_information*
- Object_ID size and assignment rules.
- DSIT:*system_specific_search_flag*.
- DSD:*dsd_selection_specification.*
- TUNER STATUS notify command:*system_specific_event*.

10. Tuner Subunit Profiles

The AV/C Tuner Model and Command Set defines a rich collection of commands and data structures which may be supported by various implementations. In order to promote interoperability, it is useful to enable controllers to be built with a certain set of expectations for what kinds of Tuner Subunits they may encounter in an AV network. These expectations are influenced by the kind of broadcast system(s) supported by the tuner, and the commands and data structures that might be needed by controllers. The collection of commands and data structures supported by a given implementation is known as its profile.

In this document, only a few universal profiles are defined. These profile assignments are the same for any broadcast system. In addition to the universal profiles, each AV/C tuner broadcast system specification document may define a set of profile assignments for implementations which support that system. Each of those values, when combined with the *system_id* specified in the *system[x]_specification*, specify a combination of commands and data structures which are supported by the Tuner Subunit for that broadcast system.

A Tuner Subunit shall specify one profile for each broadcast system that it supports. This profile is identified in the *implementation_profile_ID* field of each *system[x]_specification* in the *subunit_dependent_information*.

10.1 Universal Profile ID Assignments

The following table illustrates the universal profile ID values which have been defined:

Table 10.1 – Tuner *implementation_profile_id* definition

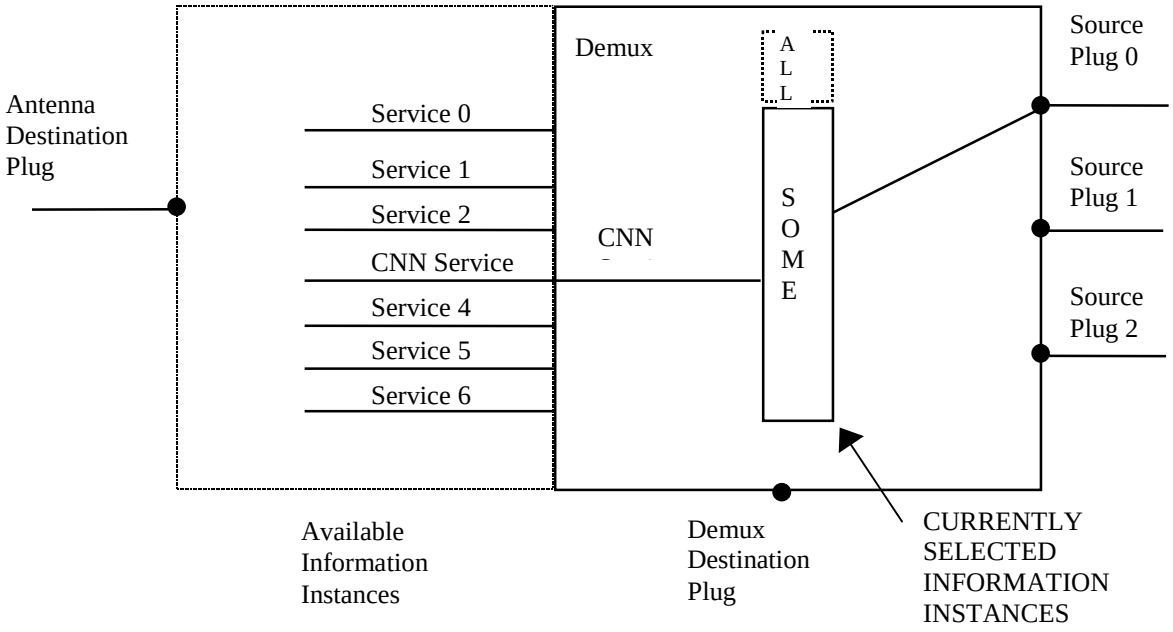
Implementation_profile_id	meaning
$E0_{16}$	conformant_implementation - a Tuner Subunit with this implementation profile ID was created based on the AV/C Tuner Specification version 1.0 and above. The set of features (commands and data structures) supported by this implementation is defined by the manufacturer. This profile ID applies to all broadcast systems.
$E1_{16}$	conformant_full_implementation - a Tuner Subunit with this profile implementation is as described above, but it implements all of the commands and relevant data structures for the specified broadcast system, as defined in the AV/C Tuner Specification version 1.0 and above. This profile ID applies to all broadcast systems.
$F0_{16}$	non_conformant_implementation - a Tuner Subunit with this implementation profile ID was created based on the AV/C Tuner Working Specification version 1.0W. The set of features (commands and data structures) supported by this implementation is defined by the manufacturer. Products with this ID value shall not claim conformance to the AV/C tuner model and command set. This profile ID applies to all broadcast systems.
$F1_{16}$	non_conformant_full_implementation - a Tuner Subunit with this profile implementation is as described above, but it implements all of the commands and relevant data structures for the specified broadcast system, as defined in the AV/C Tuner Working Specification version 1.0W. This profile ID applies to all broadcast systems.
all other values in the range $E0_{16}$ - FF_{16}	reserved for future specification in this AV/C Tuner Specification
values in the range 00_{16} – DF_{16}	reserved for specification in each AV/C Tuner Subunit Broadcast System Specification document (each broadcast system, as defined by its *system_id*, can define values in this range)

Annexes

Annex A: Tuner Subunit Demuxing and Selection Examples

A.1 Single Service Selection

Simple digital tuner example: the user wants to view a single Service, such as CNN. This requires that we select the Service from the antenna. There is no input from the demux destination plug.



Notes:
1) The input is from the antenna destination plug. There are 7 Services available at the antenna destination plug.

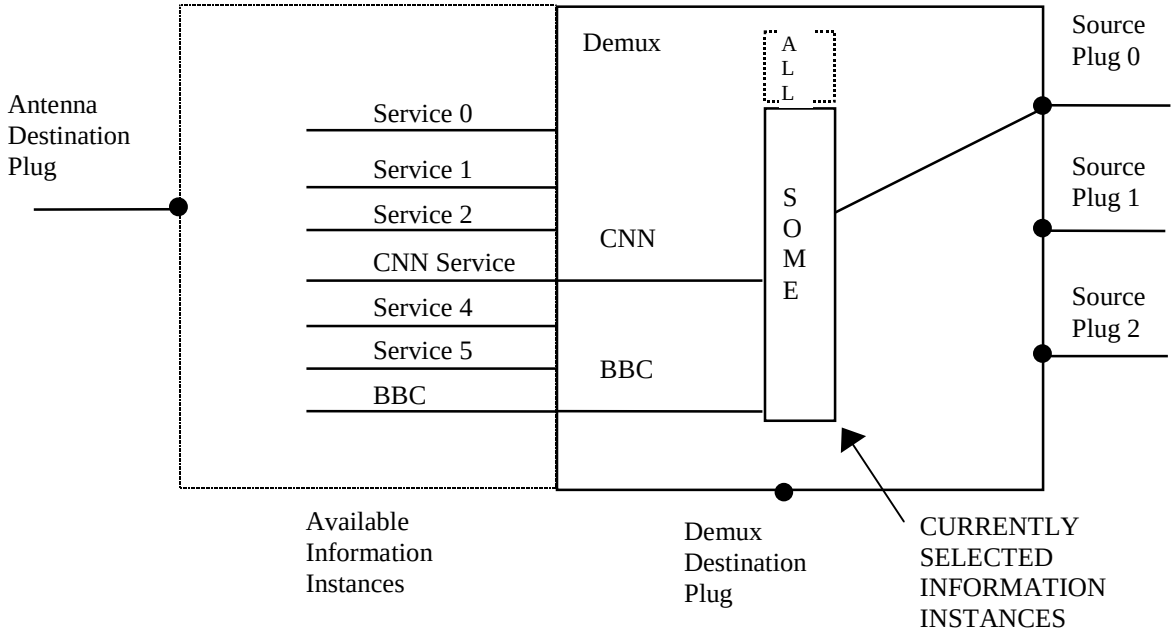
2) The controller has issued either OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE, to cause the CNN Service to become the *currently selected information instance*. It has also caused CNN to be output on source plug 0.

3) If the controller wishes to have only specific components on the source plug, then the above OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE command should also specify the desired components. These commands work from the available information instances.

Figure A.1 – Single Service Selection

A.2 Multiple Services Selection

Another digital tuner example: the user is recording two Services to videotape for later viewing. There is no input from the demux destination plug.



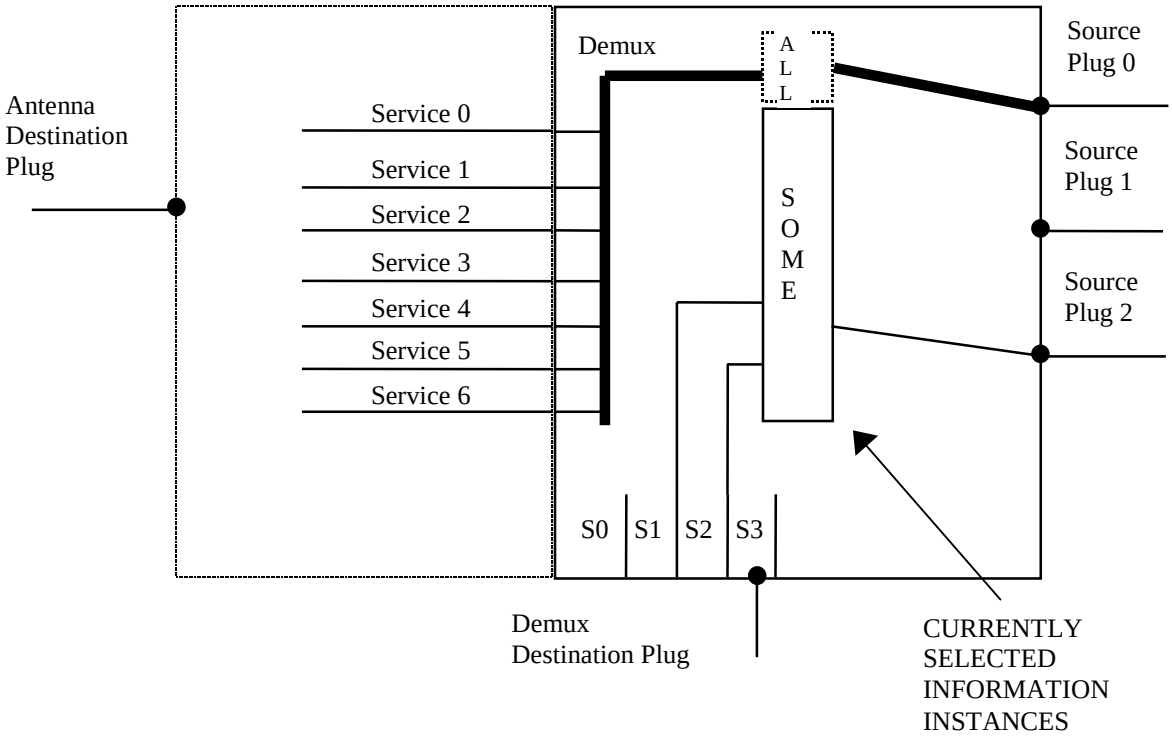
Notes:
1) The input is from the antenna destination plug. There are 7 Services available at the antenna destination plug.

2) The controller has issued either OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE, to cause the CNN and BBC Services to become the *currently selected information instances*. It has also caused them to be output on source plug 0. For this recording example, it is assumed that the appropriate connections to either a VCR subunit or the 1394 serial bus have been made.

Figure A.2 – Multiple Service Selection

A.3 A Special Case: selecting the Multiplex from the antenna and the Services from the serial bus

A more complex digital tuner example: selecting all information instances from the antenna, and some information instances from the demux.



Notes:

1) The input is from both the antenna and demux destination plugs. There are 7 Services available at the antenna destination plug, and 4 Services available at the demux destination plug.

2) The controller has issued either OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE, to cause ALL information instances available at the antenna destination plug to be routed to source plug 0 (the tuner subunit is doing a *pure tuning* operation). IMPORTANT: If there were no other selection activity occurring with the demux destination plug, then all of these information instances would be considered the *currently selected information instances*. However, read on...

3) The controller then issues either OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE to select Services 1 and 2 from the Services available at the demux destination plug, and have them routed to source plug 2. Because there is partial Service selection being performed, this collection becomes the *currently selected information instances*.

4) If the controller wanted to have specific components added to the output of source plug 2 along with Services 1 and 2, then the OBJECT NUMBER SELECT or DIRECT SELECT INFORMATION TYPE commands should specify the desired components. Those commands work from the available information instances.

Figure B.3 – Special Case Selection