AV/C Tuner Model and Command Set

Version 1.0
April 15, 1998

Sponsored by:
Audio/Video Working Group of the 1394 Trade Association

Approved for Release by:
This document has been approved for release by the 1394 Trade Association Board of Directors

Abstract: This specification defines a model and command set for analog and digital tuners operating over IEEE 1394-1995. The command set makes use of the Function Control Protocol (FCP) defined by IEC 61883, Digital Interface for Consumer Electric Audio/Video Equipment 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
3925 W. Braker Lane, Austin, TX 78759 USA
http://www.1394TA.org

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 approved 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
3925 W. Braker Lane
Austin, TX 78759
USA

Table of Contents

1. Normative References

The following documents may be useful to the reader interested in learning about the full AV/C protocol and related technologies. All standards are subject to revision; the reader is encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

This AV/C Tuner Model and Command Set specification must be used in conjunction with the general AV/C specification and the appropriate tuner subunit broadcast system specification(s), according to the product being designed. All of these specifications are referenced below.

1.1 Contact Information

The documents referenced herein may be obtained from the following organizations:

1.1.1 1394 Trade Association (1394 TA)

The 1394 Trade Association can be contacted via the references provided on the cover page of this and all AV/C specification documents.

1.1.2 Association of Radio Industries and Business (ARIB)

Nittochi Bld. 14F 1-4-1 Kasumigaseki Chiyoda-ku Tokyo
100-0013 Japan

Phone: +81-3-5510-8590
Fax: +81-3-3592-1103

1.1.3 Advanced Television Systems Committee (ATSC)

Documents from the ATSC can be located on the following WWW site:
http://www.atsc.org

1.1.4 European Telecommunications Standards Institute (ETSI)

ETSI Secretariat
Postal Address: F-06921 Sophia Antipolis Cedex - FRANCE
Office Address:650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE

Phone: +33-4-92-94-42-00
Fax: +33-4-93-65-47-16
Internet: secretariat@etsi.fr
http://www.etsi.fr

1.1.5 International Electrotechnical Commission (IEC) (contact in the United States)

U.S. National Committee of the IEC ANSI
11, West 42nd Street, 13th floor

New York, NY 10036

Phone: +1-212-642-4900
+1-212-642-4980 (sales)
Fax: +1-212-398-0023
Internet: http://www.ansi.org

Documents can be ordered from:

http://www.iec.ch/cs1ord-e.htm
http://www.iec.ch/cs1oi-e.htm

1.1.6 The Institute of Electrical and Electonics Engineers, Inc. (IEEE)

The IEEE can be contacted via their WWW home page: http://www.ieee.org

1.1.7 International Telecommunication Union (ITU)

The ITU can be contacted via their WWW home page:http://www.itu.int

1.2 1394 Trade Association Specifications

AV/C Master Index: Guide to AV/C Specification Documents - this document is available on the 1394 Trade Association web site noted above, and is kept up to date with the latest released versions of AV/C specifications. The reader is encouraged to always consult this document for information on the latest versions of specifications mentioned here, as well as specifications which may be developed in the future.

AV/C Digital Interface Command Set General Specification Version 3.0, April 15, 1998

AV/C Digital Interface Command Set General Specification Version 2.0.1

AV/C Tuner Model Working Specification Version 1.0W

AV/C Tuner DVB Specification Version 1.0, April 15, 1998

AV/C Tuner Analog Video System Specification Version 1.0, April 15, 1998

AV/C Tuner Analog Audio System Specification Version 1.0, April 15, 1998

1.3 Related Technical Specifications

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

ISO/IEC 13213:1994, *Control and Status Register (CSR) Architecture for Microcomputer Buses*

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

FINAL DRAFT prETS 300 401 May 1997 Second Edition, *Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers FINAL DRAFT*

ATSC Standard A/53 (1995), *Digital Television Standard*

Association of Radio Industries and Business ARIB STANDARD - ARIB STD-B5 version 1.0, *DATA MULTIPLEX BROADCASTING SYSTEM FOR THE CONVENTIONAL TELEVISION USING THE VERTICAL BLANKING INTERVAL*

ITU-R BT.470, *Analog Video Transmission*

2. Introduction

This document defines a model and command set for AV/C Tuner subunits. This model supports a variety of different broadcast systems, including both analog and digital; it is possible to create an AV/C tuner subunit which supports several of these systems.

The tuner subunit model makes full use of the AV/C descriptor mechanism, which is currently defined in the normative references below.

2.1 Rules for Reserved Fields

This section clarifies the rules which have always been in effect regarding how reserved fields shall be treated in command parameters and data structures for AV/C.

Unless otherwise specified (see note below), command parameters and data structure fields marked as “reserved” or “reserved for future specification” shall be set to zero by controllers on input to a target, and by targets on output to controllers.

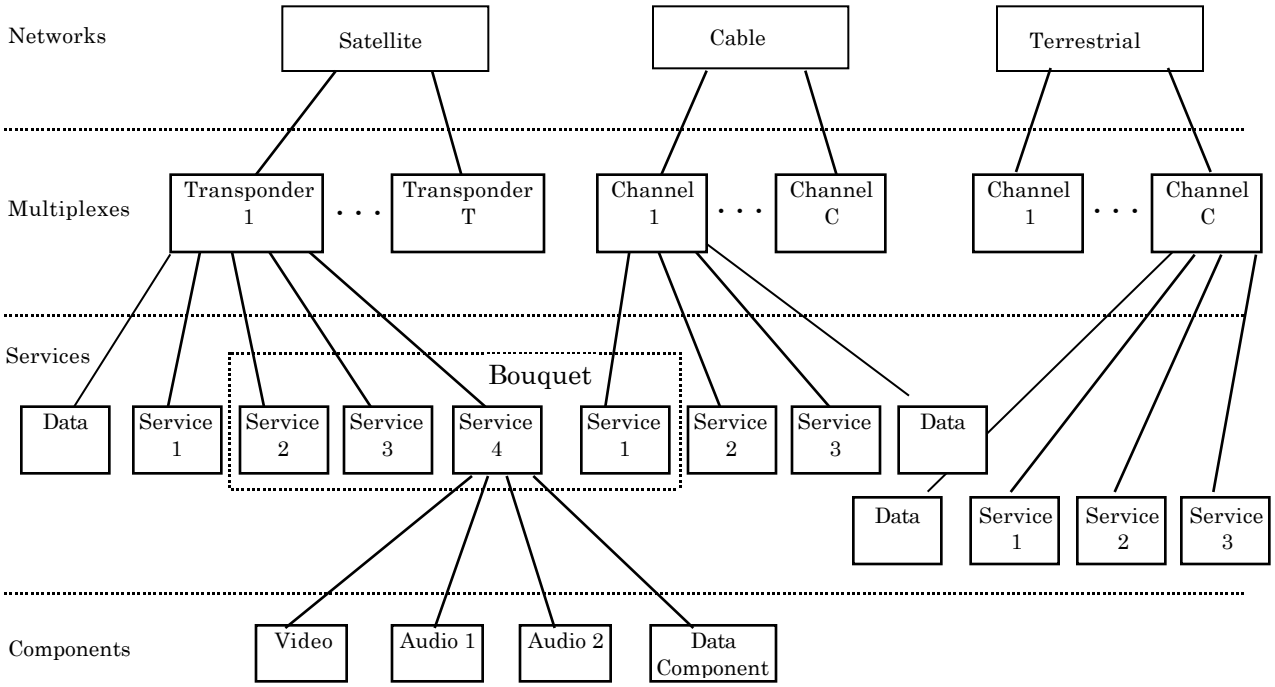
For input operands of commands, targets shall NOT ignore fields that were reserved when the target was implemented. Rather, the target shall examine the reserved fields; if any of them are specified, then the target shall reject the command with a NOT IMPLEMENTED response.

On output data structures or parameters of commands, controllers shall ignore fields that were reserved when the controller was implemented. These rules exist to allow future extension of the specification while retaining compatibility with existing products.

> **NOTE:** In some instances, reserved command operands or data structure fields may be specified as non-zero values. These cases will be clearly indicated in the specification. Controllers and targets shall deal with them in the same manner as defined above.

3. Broadcasting and Tuning Concepts

The following diagram and definitions illustrate the digital video broadcasting service delivery model as defined in the **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:



Multiplex: A stream of all the digital data carrying one or more services within a single physical channel.

Service: A sequence of programs under the control of a broadcaster which can be broadcast as part of a schedule. *

Bouquet: A Bouquet is a collection of services marketed as a single entity. As shown in the diagram above, these services may span more than one multiplex.

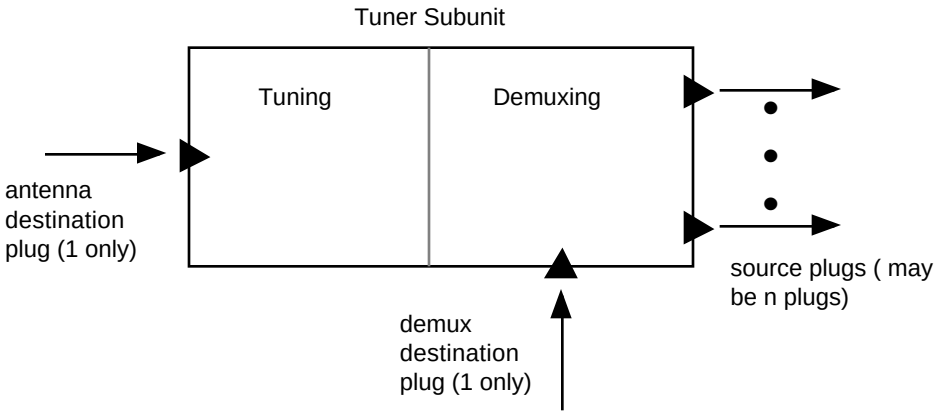
Components: One or more entities which together make up an event, e.g. video, audio, teletext.

* Note: When a service is provided by the tuner, legal and least necessary navigation data 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 PAT and PMT tables 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 **EIT** document referenced above.

The analog video broadcasting model would be slightly different. Each transponder has only a single service. Hence, an analog "multiplex" consists of a single service. Each analog service may have several components, similar to the diagram above.

3.1 The Tuner Subunit Model

The conceptual model of a tuner subunit consists of two functional stages: the tuner and the demultiplexer (also called demuxer or demux). Input to the tuning functionality is via a single antenna destination plug. The demuxer also has a single demux destination plug; output from the tuner subunit is from the demuxer, which has one or more source plugs. The following diagram illustrates this model:



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, set top box, VCR, etc.

3.2 Antenna Destination Plug

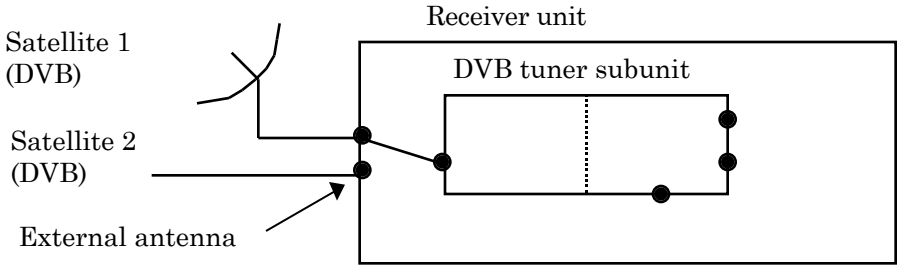
The receiver unit has two types of input plugs: external input, and serial bus (1394) input. The external input can either be an antenna input, or some other type of input. In addition, the receiver unit has two types of output plugs: external output, and serial bus output. The external output plug of the receiver is not attached to an antenna, but it may be some other type of physical connection.

Input to the tuner 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 receives its input from antennas. Any attempt to connect a serial bus plug to the tuner antenna destination plug shall be REJECTED.

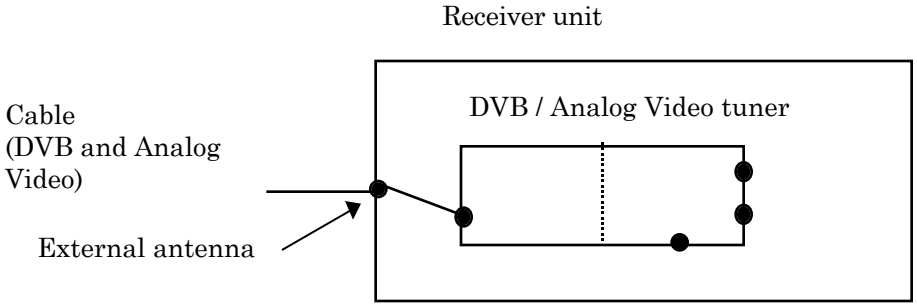
There may be many different antenna connections on the outside of a receiver unit. Each of these physical connectors must be reported as external input plugs from the PLUG INFO command when issued to the receiver unit.

These different inputs may all carry the same **type** of signal (**analog video, DVB**, etc.), or each input may carry a different type of signal. A tuner subunit can be designed to handle more than one type of signal, but it can only handle one type of signal at a time. If a receiver unit is designed to handle more than one 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.

The following diagrams illustrate the relationship of the receiver unit external antenna input plugs and the tuner subunit antenna destination plug:



In the above diagram there are two antenna inputs that provide a DVB signal, but there is only one tuner subunit that can handle DVB. To select a DVB service, it will be necessary to establish a connection between one of these two external inputs and the antenna destination plug of the tuner subunit.



In this diagram there is a single antenna input which provides both a DVB and 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 which carries the signals. However, the individual plugs (external input, subunit antenna destination) must still be reported via the appropriate PLUG INFO command.

3.3 Demux Destination Plug

The receiver unit may have one or more serial bus input and output plugs. The demux part of a digital tuner subunit has one demux destination plug, and one or more source plugs. The demux part of an analog tuner subunit also has one demux destination plug and one or more source plugs, but demuxing is different between digital and analog tuner subunits. For either kind of subunit, restrictions on making connections to these plugs are implementation dependent.

The demux destination plug may be connected to another subunit within the receiver unit.

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. Although the analog tuner cannot demultiplex multiple services from a stream, it can select certain components of a single analog stream such as closed captioned text. As a result, the analog tuner demux destination plug exists to receive input from the non-antenna input plug of the receiver unit. Please refer to the section titled Tuner Demuxing and Output for details on the

nature of multiplexed digital streams, and the multiplexing that can be performed on analog streams.

3.4 Connections

The main purpose of the tuner subunit is to allow a controller to select a service. When the receiver unit's external antenna input plugs are used, then the connection to the tuner subunit's antenna destination plug shall be made automatically. This connection shall be made transparently as part of service selection via the subunit antenna destination plug.

If it is necessary to establish a connection between the receiver's external (non-antenna) input plugs or serial bus input plugs, or another subunit, and the tuner subunit's demux destination plug, then this connection MUST be established manually using a CONNECT command. This connection must be made before issuing the corresponding service selection command.

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 CONNECT command.

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:

Destination Plug	Plug Number Value
Antenna Destination	0
Demux Destination	1

If a service selection command is issued to the tuner subunit which requires connection 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.

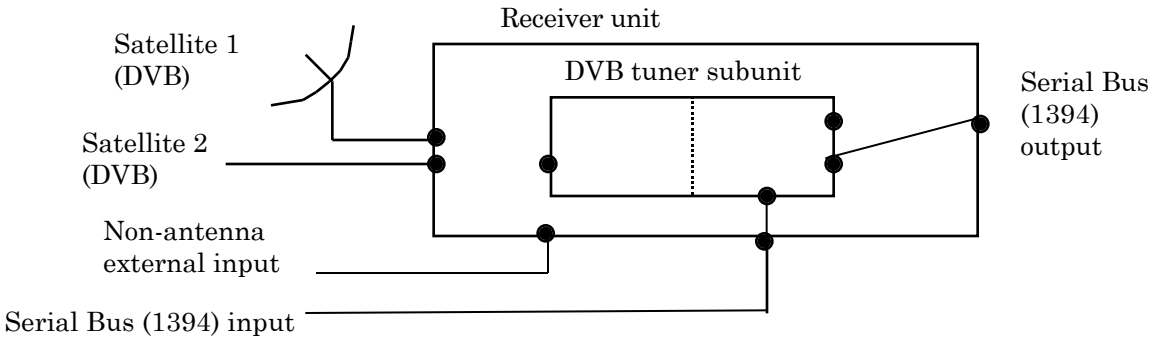
All current connections of tuner subunits shall be reported by the CONNECT status or CONNECTIONS status commands. This includes all permanent connections as described above.

For the source plugs of the tuner subunit, it shall be necessary to use the CONNECT command to establish a connection between them and the destination plugs of another subunit within the receiver unit or the serial bus 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 required if the connection between the tuner subunit source plug and a receiver unit serial bus output plug is not 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.

The following table illustrates the various combinations of connections between receiver unit and tuner subunit plugs, and which ones are valid or not. All invalid connections shall generate a response of NOT IMPLEMENTED. Unless otherwise noted, all descriptions apply to both analog and digital tuner subunits:

Non-Tuner Subunit Plug	Tuner Subunit Plug	Connection Valid?	Comments
External antenna input plug	Antenna destination plug	Yes	Made automatically as part of a service selection command.
External antenna input plug	Demux destination plug	No	X
External antenna input plug	Source (output) plug	No	X
External (non-antenna) input plug such as video input line	Antenna 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 CONNECT command, or it may be a permanent connection. This connection is for analog tuners.
External (non-antenna) input plug such as video input line	Source (output) plug	No	X
External output plug	Antenna destination plug	No	X
External output plug	Demux destination 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 CONNECT command, or it may be a permanent connection.
Serial bus input plug	Antenna 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 CONNECT command, or it may be a permanent connection. This connection is for digital tuners.
Serial bus input plug	Source (output) plug	No	X
Serial bus output plug	Antenna destination plug	No	X
Serial bus output plug	Demux destination 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 CONNECT command, or it may be a permanent connection.
Subunit source plug	Antenna destination plug	No	X
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 CONNECT command, or it may be a permanent connection. Valid for both digital and analog tuners.
Subunit source plug	Source (output) plug	No	X
Subunit destination plug	Antenna destination plug	No	X
Subunit destination plug	Demux destination 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 CONNECT command, or it may be a permanent connection. Valid for both digital and analog tuners.

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



3.5 Tuner Demuxing and Output

The following definitions apply to the demuxing and output operations of a tuner subunit:

Information Type There are four possible types of information that can be individually controlled with a tuner subunit: Multiplex, Service, Component or Data.

A *service* is as described above, at the beginning of the Broadcasting and Tuning section. For illustration purposes, we can consider a service to be the thing that people normally watch, such as CNN, BBC, Disney, etc.

A *component* is one 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, and audio 2.

In addition to audio and video components, a service may have associated *data components*. Examples of such data would be closed caption text and teletext. We use the term "data component" to distinguish between this kind of data and the kind that is described next.

A *multiplex* is the complete collection of services and data from one transponder.

As shown in the diagram at the beginning of the Broadcasting and Tuning Concepts section, there is also some data that is sent with the collection of services in an input signal (the term input signal is defined below). This kind of data is not a component of a service. Examples of this kind of data, for DVB broadcasting, would be the Network Information Table (NIT), or the Event Information Table (EIT).

Multiplex The physical waveform that contains one or a multiplex of services, components or data. The antenna input plug carries a "set" of Multiplexes from which only one can be selected 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.

Information Instance This is an actual piece of information in a Multiplex, including the Multiplex itself. An Information Instance may be of any defined Information Type (multiplex, service, component, data).

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

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.

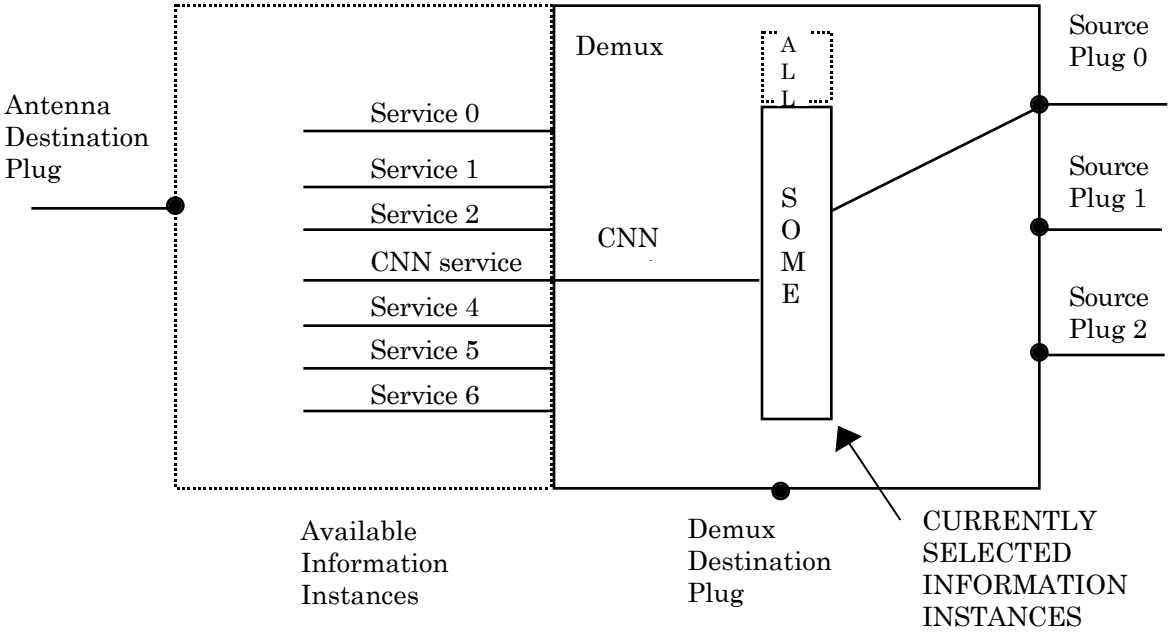
Currently Selected Information Instances A tuner subunit may have many Information Instances available at its selected destination plug. The tuner will be instructed by a controller to select one or more, or possibly all, of these Information Instances, so that they can be made available on the tuner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

Demuxing The selection of individual information instances from the input signal. This could be one or more services, individual components, data, or any combination of Information Types.

Multiplex Selection Under some circumstances, it may be desirable to select all of the available information instances in a multiplexed stream, and make them available on the tuner subunit source plug(s). This is a special case of tuning, in which the demuxing functionality is not used. When this happens, the demuxer is available to perform its function as a separate operation.

Consider the following diagram which shows the selection of only some services via the antenna destination plug of the tuner subunit:

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.

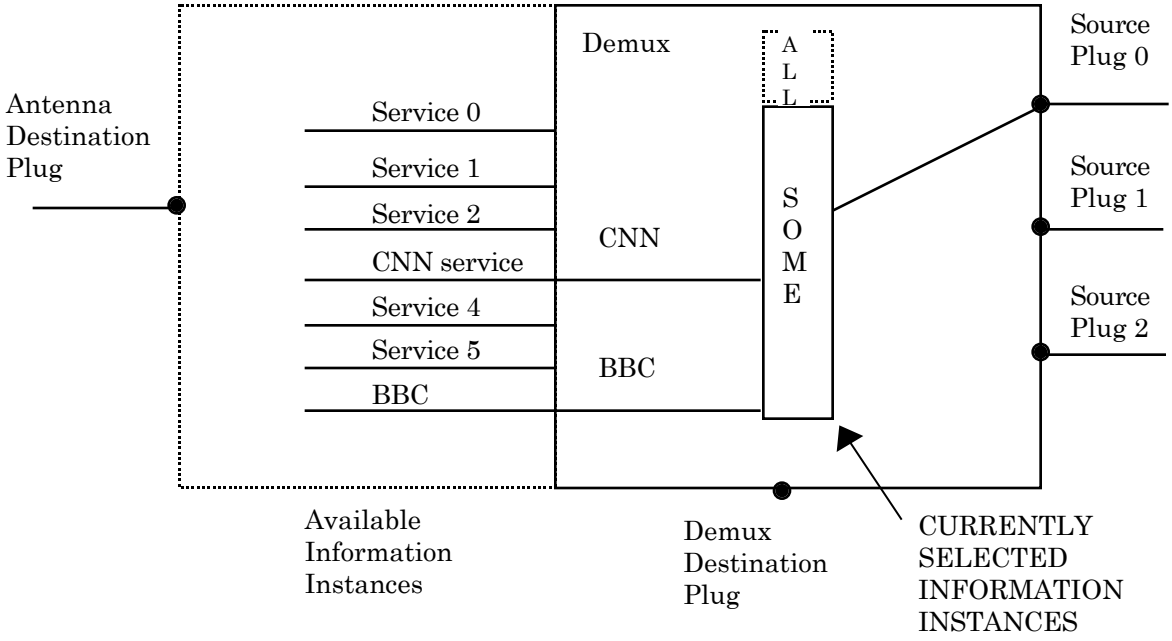


Notes:
1) The input is from the antenna destination plug. There are 7 services available at the antenna destination plug.

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.

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

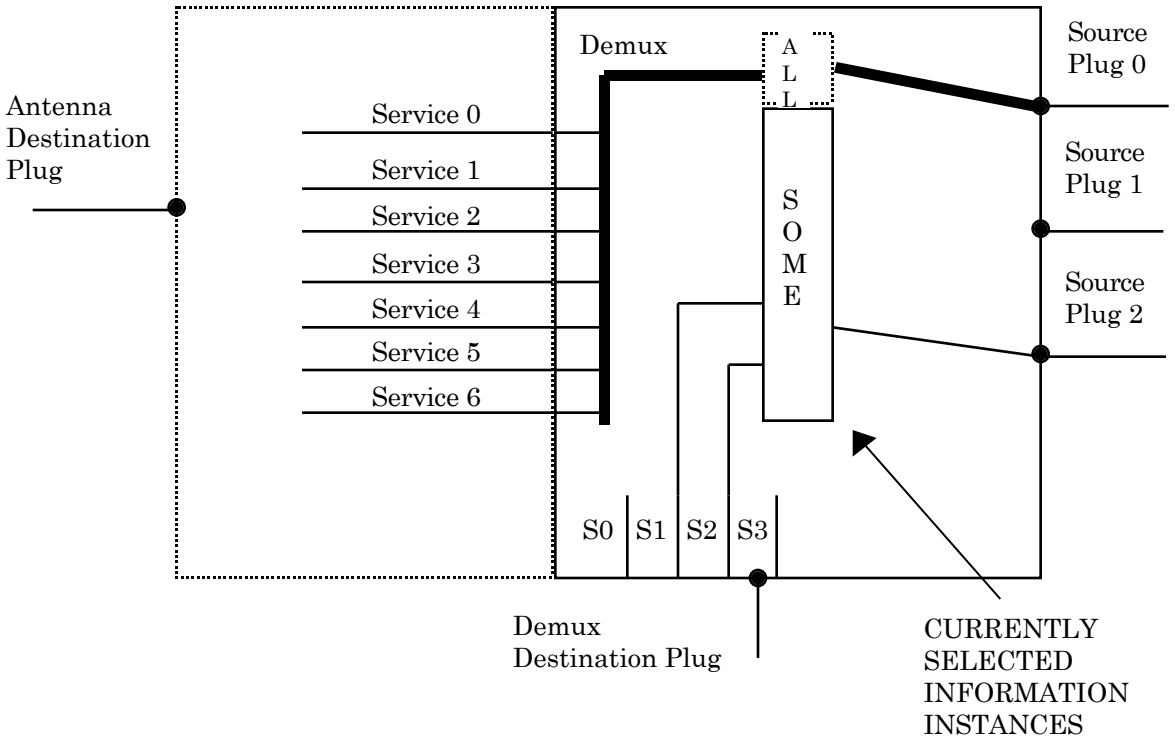


Notes:
1) The input is from the antenna destination plug. There are 7 services available at the antenna destination plug.

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.

The above diagram shows the selection of only two services from those that are available at the antenna destination plug. In some cases, it is possible to use both the antenna destination and demux destination plugs at the same time. Under these circumstances, one of the plugs must be selecting all of the information instances that are available (multiplex selection), and the other must be selecting only some of the information instances that are available (demuxing). The following diagram illustrates this concept:

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.

Depending on the type of tuner subunit, and the types of signals supported by the tuner subunit, the output from a plug may be any collection of these information types:

A multiplexed stream (a collection of services such as CNN, HBO, Disney, NHK1 and associated data)

A single demultiplexed service (such as CNN only)

A single component of a service (such as Audio2 of CNN)

The data (such as EIT - Event Information Table)

Certain types of broadcast systems, such as analog video, do not consist of multiplexed services, so the output of an analog video tuner subunit will be a combination of three types:

A single service (CNN)

A component of a service (the language audio)

The data (such as closed caption text)

4. Tuner Subunit Identifier Descriptor, Status Descriptor, Objects and Object Lists

4.1 Text Field Encoding

All text fields in the descriptors specified in this section (subunit identifier descriptor, status descriptor, objects and object lists) shall be encoded as printable English 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 which are not in the European DVB specification (they are defined in ARIB STD-B5).

4.2 Tuner Subunit Identifier Descriptor

In the case of a tuner subunit, the SUBUNIT IDENTIFIER describes the characteristics of the broadcast system(s) supported by the tuner. More than one broadcasting system may be supported by a tuner subunit; this is called a multi-system tuner. It may have several object lists associated with it. Please refer to the section titled Tuner Model Objects and Object Lists for details on the objects and object lists defined for the tuner model.

A tuner subunit shall have the same basic SUBUNIT IDENTIFIER DESCRIPTOR structure as presented in the AV/C Descriptor Mechanism.

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
:	

The *number_of_systems* field specifies how many broadcast systems are supported by this tuner subunit.

The *system[x]_specification* fields each describe a broadcast system; each has 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
:	
:	

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:

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}	reserved for DTV
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 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 Universal Profile ID Assignments, which begins on page 62.

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.

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 subsystem label does not mean that this tuner is unable to select from OTHER service providers; it is simply a means to associate the preferred selection flags (defined next) with service providers, to assist in tuning actions. If the *sub_system_label* 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 *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 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 which 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 for 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.

The details of the mandatory selection attributes for each broadcasting system are presented in the system-specific sections of this document. 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.

The *number_of_antennas* field contains the number of receiver unit antennas *available* to this tuner subunit for this broadcast system. Note that there will only be zero or one antenna *connected* to the subunit at any given time.

The *antenna specification* fields are an array of antenna specifications. All antenna specifications have this format:

address offset	contents			
00_{16}	mobile	movable	reserved (3 bits)	transport (3 bits)
01_{16}	input_plug			
02_{16}	system_specific_antenna_range_specification_length			
03_{16}				
04_{16}	system_specific_antenna_range_specification			
:				
:				

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

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
11	mobile and movable - the antenna has both of these characteristics

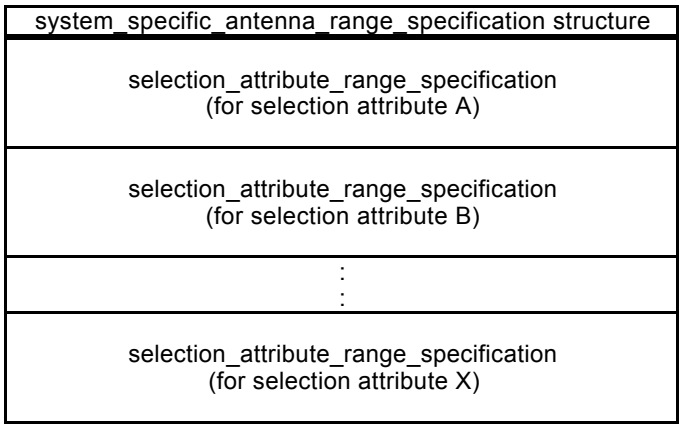
The *transport* field is defined as follows:

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

The *input_plug* field specifies the receiver unit's external antenna input plug number which corresponds with this antenna specification.

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

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):



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 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 system specific sections of this document.

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		
:			
:			

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	0

When both flags are zero, then there is no range specification for this selection attribute.

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			

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

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]		
:			
:			
:	:		
:	value[n - 1]		
:			
:			

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

The *value[x]* fields each hold one of the values. The number of bytes used for each of these entries is equal to *size_of_attribute.*

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]		
:			
:			
:	ending_point[0]		
:			
:			
:	:		
:			
:	starting_point[n - 1]		
:			
:			
:	ending_point[n - 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 *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 number of bytes used for each entry is equal to *size_of_attribute.*

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

4.3 Tuner Status Descriptor (a subunit type-dependent descriptor)

The tuner status descriptor structure is specific to tuner subunits. It holds information about the tuner subunit in general, and about what information is 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 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:

Tuner Status Descriptor	
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]
:	
:	

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:

general_tuner_status 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
:	
:	

The *antenna_input_info_length* field contains the number of bytes for the following *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
antenna_input_info								
00_{16}	active_system							
01_{16}	searching	moving	no_RF	reserved				
02_{16}	input = 0	selected_antenna						
:	antenna_general_system_info							

The *active_system* field specifies the *system_id* of the currently active system. 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. 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 an RF carrier present (e.g. while searching). A value of 0 means an RF carrier is present.

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 antenna destination plug.

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 (see the normative references for details).

The *system_specific_multiplex_selection_length* field specifies the number of bytes for the following *system_specific_multiplex_selection* field. That field has the following format:

system_specific_multiplex_selection	
address offset	contents
00_{16}	system_specific_multiplex_attributes_valid_flags
:	
:	system_specific_multiplex_selection_attributes
:	
:	

The *system_specific_multiplex_attributes_valid_flags* and *system_specific_multiplex_selection_attributes* fields are as described for the DIRECT SELECT INFORMATION TYPE control command (see page 47).

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
demux_input_info								
00_{16}	active_system							
01_{16}	searching	reserved						
02_{16}	input = 1	reserved						
:	demux_general_system_info							

The *searching* field indicates whether the tuner is searching. 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 (see the normative references for details).

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:

source_plug_status[x] format	
address offset	contents
00_{16}	source_plug
01_{16}	attributes
02_{16}	input | reserved
03_{16}	data_status_length
04_{16}	data_status
:	
:	
:	info_type_status_length
:	info_type_status
:	
:	

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

The *attributes* field has the following format:

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 which is coming out on 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 59). 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]							
:								
:								

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

value	status description
00_{16}	No information instances are on the specified source plug.
10_{16}	The information instances listed in the following fields are currently on the specified source plug.
20_{16}	The information instances 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).

All other operands are as described for the DIRECT SELECT DATA control command (see page 59).

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]							
:								
:								

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 47).

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 indicated by the *source_plug* 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.

4.3.1 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 AV/C Descriptor Mechanism):

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.

4.4 Tuner Model Objects and Object Lists

This section describes the basic concepts in the objects and object lists defined specifically for the tuner subunit model. Because there are different types of tuner subunits (such as DVB, analog video, etc.), the objects associated with each one will have different formats and contents for the technology-specific areas. For detailed information about each of the objects and object lists defined for the tuner subunit model, please refer to the system-specific sections of this document.

The list hierarchy defined for the AV/C tuner model is currently defined for a set of three levels (multiplex, services, components), and only one child list for an object. However, the flexible nature of the AV/C object list model allows future growth to support other broadcast systems which may have different configurations.

4.4.1 Some Definitions

The following definitions will be useful when reading about tuner subunit objects and object lists:

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 multiplex selection, or it may be the selection of a particular service, component or data.

Mandatory Selection Attributes These are 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.

Valid Flags Valid flags are a set of bits which indicate the validity of attributes within a descriptor. When the corresponding attribute is valid, then its valid flag is set to 1.

The following table presents the currently defined list_type and ID values for the tuner subunit lists:

list	list_type value	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 37.
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.

The following table contains the object entry_type definitions for tuner subunit-specific objects:

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 Component	90_{16}	This object entry may exist in a Preferred Components list
Reserved	all other values	reserved for future definition

4.4.2 Multiplex List and Multiplex Objects

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 (see the normative references for details).

The Multiplex List Descriptor is as follows:

Multiplex List Descriptor	
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$_{16}$	
00 06$_{16}$	number_of_entries (n)
00 07$_{16}$	
00 08$_{16}$	object_entry[0]
:	
:	
:	:
:	object_entry[n-1]
:	
:	

The *descriptor_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 *attributes* for the Multiplex 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.

All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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

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

Multiplex Object Entry Descriptor	
address offset	contents
00$_{16}$	descriptor_length
01$_{16}$	
02$_{16}$	entry_type = 80$_{16}$
03$_{16}$	attributes = xxxx xxxx
:	child_list_ID (if specified by the has_child_list_ID flag)
:	
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information (MSB)
:	(LSB)
:	entry_specific_information (multiplex descriptor)
:	
:	

The *descriptor_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 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_list_ID* attribute bit will be set accordingly. All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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 multiplex entry is a multiplex descriptor. The format of this field depends on the *system_id* for which this multiplex entry is being defined. For details, refer to the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

4.4.3 Service Lists and Service Objects

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 other, 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 Service List Descriptor is as follows:

Service List Descriptor	
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]
:	
:	

The *descriptor_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 *attributes* for the Service 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.

All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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

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

Service Object Entry Descriptor	
address offset	contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 82_{16}
03_{16}	attributes = xxxx xxxx
:	child_list_ID (if specified by the has_child_list_ID flag)
:	
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information (service descriptor)
:	
:	

The *descriptor_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 service object may or may not have a child list, so its *has_child_list_ID* flag will be set accordingly. All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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 service entry is a service descriptor. The format of this field depends on the *system_id* for which this service entry is being defined. For details, refer to the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

4.4.4 Component Lists and Component Objects

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 on non-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 Component List Descriptor is as follows:

Component List Descriptor	
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]
:	
:	

The *descriptor_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 *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.

All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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

The format of the component object entry is as follows:

Component Object Entry Descriptor	
address offset	contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 84_{16}
03_{16}	attributes = xx0x xxxx
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information (component descriptor)
:	
:	

The *descriptor_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.

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_list_ID* attribute bit will be set accordingly. All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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 component entry is a component descriptor. The format of this field depends on the *system_id* for which this component entry is being defined. For details, refer to the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

4.4.4.1 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, 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) 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.

2) 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.

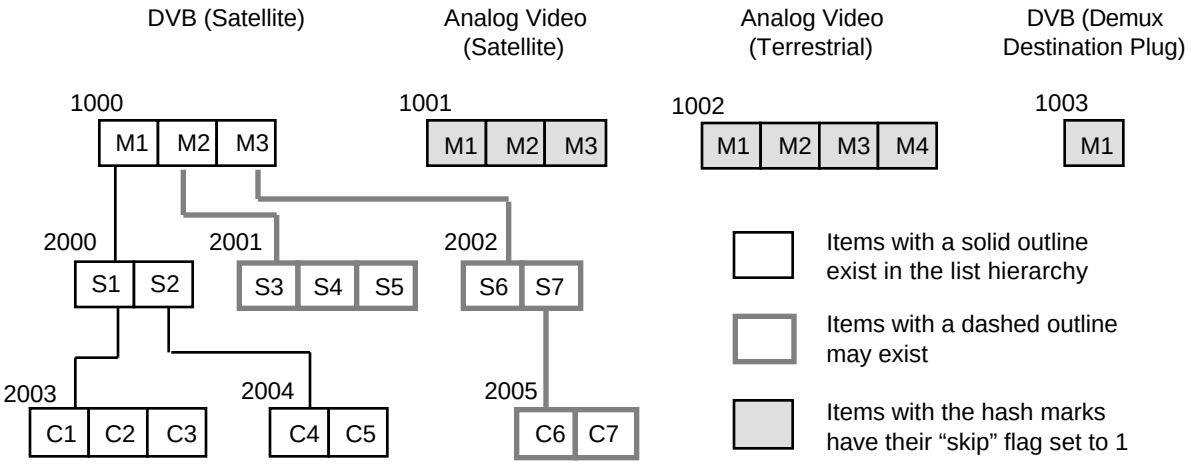
3) All of the lists in this hierarchy are READ ONLY. The subunit shall return a NOT IMPLEMENTED response for any attempt by a controller to modify the lists in any way. The information in these lists is intended to be for status purposes, and is maintained only by the tuner subunit.

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 and a DVB system, and a terrestrial antenna which receives analog video, and the tuner subunit can receive a DVB signal on its demux. 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 which are optional. The tuner subunit SHALL support descriptor access at the object entry level. Also, it shall support the {list_type, object_ID} and {list_ID, object_position} references for the descriptor identifier structure.

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 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 mandatory 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.

4.4.5 Preferred Components List and Preferred Components Objects

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:

Preferred Components List Descriptor	
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]
:	
:	

The *descriptor_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.

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.

All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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

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

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

Preferred Components Object Entry Descriptor	
address offset	contents
00_{16}	descriptor_length
01_{16}	
02_{16}	entry_type = 90_{16}
03_{16}	attributes = xx0x xxxx
:	object_ID
:	
:	
:	
:	size_of_entry_specific_information
:	
:	entry_specific_information (preferred components descriptor)
:	
:	

The *descriptor_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 *attributes* of the component entry specify that it has no *child_list_ID*. All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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 is a preferred components descriptor, which appears as follows:

Preferred Components Object Entry Specific Information	
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]
:	
:	

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 this 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 preference indicators 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 the 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 (see the normative references for details).

4.4.6 Preset Lists and Preset Objects

The preset object defines a preset combination of components, or a service or 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:

Preset List Descriptor	
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_info
00 05_{16}	
00 06_{16}	list_specific_info
:	
:	
:	number_of_entries (n)
:	
:	object_entry[0]
:	
:	:
:	object_entry[n-1]
:	
:	

The *descriptor_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.

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.

All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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

Preset list_specific_info field	
address offset	contents
00_{16}	list_name_length
01_{16}	list_name
:	
:	

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 format of the preset object entry is as follows:

Preset Object Entry Descriptor	
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)
:	
:	

The *descriptor_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 *attributes* of this preset entry specify that it has no *child_list_ID*. All undefined attribute bits shall be treated as reserved, as specified in Rules for Reserved Fields on page 8.

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:

Preset Object entry_specific_information field		
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	
:		
:		

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 47. 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.

4.4.7 Rules and Guidelines for Tuner Subunit Objects and Object Lists

Objects and object lists are used to represent certain attributes or characteristics of technologies related to a particular type of subunit. Because each type technology has its own individual set of characteristics, not all objects and object lists can be assumed to behave in a "standard" way. This section covers rules that are defined by tuner subunits.

4.4.7.1 Migrating Objects, Object References and 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 object ID's for the preset and preferred components objects can be defined by the tuner implementation. These assignments shall be within the general rules defined by the AV/C list and object model.

2) The assignment of object ID's for the multiplex, service and component lists are defined per system_id. For details, please refer to the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

3) The *size_of_object_ID* field in the tuner subunit identifier descriptor specifies the number of bytes used for ALL object ID values within the scope of the subunit. The *size_of_object_ID* field shall be consistent with object ID assignments mentioned in (1) and (2) above.

4) The *size_of_list_ID* field for tuner subunits, also in the subunit identifier descriptor, shall be two (2) bytes.

5) The *size_of_object_position* field for tuner subunits shall be two (2) bytes.

In the case of the tuner subunit, the Multiplex, Service and Component lists all represent information that is 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 object lists to represent related collections of multiplexes and services, it is possible that service objects will also migrate from one list to another. This has implications on the way a controller should behave when dealing with these entities.

For example, the general list model defines two ways of referring to an object in a list: either by its position in a specified list, or by its unique object ID.

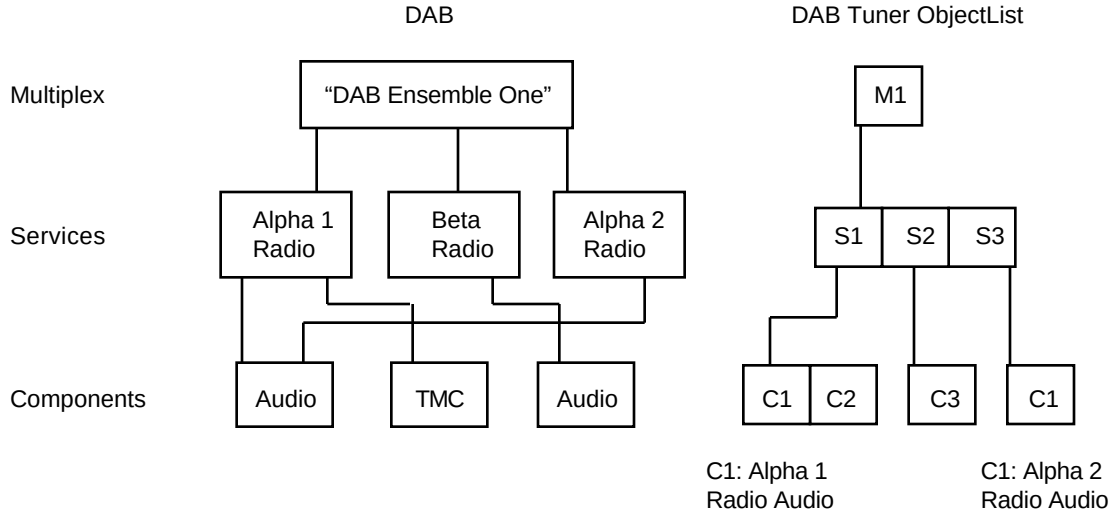
Because the broadcast lists (multiplex, service and component) are so dynamic, it is not safe for a controller to assume that a given object will always be in the same position in a list. The action of one object leaving a list and joining another may cause a ripple effect which could shift several objects 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, 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.

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 DAB example, the object ID for both instances of C1 is not unique 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:



5. Tuner Subunit Commands

Tuner subunit commands are identified by a subunit_type value of 05_{16}. This section describes commands which are defined specifically for tuner subunits. This section also describes how some of the common subunit commands are applied to the tuner subunit. Please refer to the section titled Tuner Subunit Profiles on page 62 for a detailed table which illustrates various commands and how they apply to certain kinds of tuner subunits.

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

(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.

		Support level (by ctype)			
Opcode	Value	C	S	N	Comments
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

5.1 DIRECT SELECT INFORMATION TYPE

For tuner subunits, the DIRECT SELECT INFORMATION TYPE command deals with the information types *multiplex*, *service* and *component*. It is NOT used to select the *data* information type.

Selection of this information 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 plug to receive that multiplex; the source 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 plug to receive the signal can be either the antenna destination or 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]							
:								

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 the General AV/C specification.

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 AV/C Descriptor Mechanism. 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 and 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 (see the normative references for details).

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 to the control command. In the ACCEPTED response frame, this field shall be updated with the appropriate value as defined here:

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, the response frame shall consist only of 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 which begins on page 20.

The *input* flag and *antenna_number* field are described in the broadcast system specification documents (see the normative references).

The *system_specific_search_flags* field allows the controller to specify a search action to be taken during the selection. 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 only 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 (its valid flag will be ignored). 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 all 1's 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 on 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 following *system_specific_multiplex_selection* field. That field has the following format:

system_specific_multiplex_selection	
address offset	contents
00_{16}	system_specific_multiplex_attributes_valid_flags
:	
:	system_specific_multiplex_selection_attributes
:	
:	

The *system_specific_multiplex_attributes_valid_flags* field contains bit flags that are set by the controller, to indicate which of the multiplex attributes in the command parameters are valid. Only the valid parameters shall be used by the tuner for selection purposes. The parameters whose valid flags are set to 0 shall be treated as "don't care". The format and meaning of these flags are defined per *system_id*, in the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

The *system_specific_multiplex_selection_attributes* field contains the selection attributes for the multiplex, according to the format defined by *system_id*.

When the *system_specific_multiplex_attributes_valid_flags* are all zero, 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. Also, the *system_id*, *input*, *antenna_number* and *system_specific_search_flags* 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, service or 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 (see the normative references for details).

The general format of this operand is as follows:

dsit_selection_specification	
address offset	contents
00_{16}	specification_length
01_{16}	information_type
02_{16}	info_type_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 types is defined by the *info_type_selection_specification* field. The values are defined as follows:

information_type	value
service	82_{16}
service with specified components	83_{16}
component	84_{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 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.

5.1.1 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 plug specified in the command using the specified subfunction. The specification for selecting a complete service is as follows:

dsit_selection_specification for a complete service	
address offset	contents
00_{16}	specification_length
01_{16}	information_type = service (82_{16})
02_{16}	system_specific_service_attributes_valid_flags
:	
:	system_specific_service_selection_attributes
:	
:	

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 the *information_type* table above.

The *system_specific_service_attributes_valid_flags* field indicate which of the service selection attributes should be matched by the tuner when performing the selection operation.

The *system_specific_service_selection_attributes* is similar to the multiplex specifier, but of course it provides the detailed specification of a service for the given broadcast technology as defined by the *system_id* field. These specifications are in the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

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

5.1.2 Selecting a Service Using 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 using 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:

dsit_selection_specification for a service with specified components	
address offset	contents
00_{16}	specification_length
01_{16}	information_type = service with specified components (83_{16})
02_{16}	system_specific_service_attributes_valid_flags
:	
:	system_specific_service_selection_attributes
:	
:	
:	**number_of_components (m) <> FF_{16}**
:	system_specific_component_attributes_valid_flags [0]
:	system_specific_component_selection_attributes [0]
:	
:	
:	:
:	system_specific_component_attributes_valid_flags [m - 1]
:	system_specific_component_selection_attributes [m - 1]
:	
:	

All of the fields down to *system_specific_service_selection_attributes* 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_attributes_valid_flags* and *system_specific_component_selection_attributes* fields included in the specification.

The *system_specific_component_attributes_valid_flags [x]* fields indicate which of the component selection attributes should be matched by the tuner when performing the selection operation.

The *system_specific_component_selection_specifier [x]* fields each specify one component of the service, according to the broadcast system specified by *system_id*. These are also defined in the appropriate AV/C Tuner Broadcast System Specification document (see the normative references for details).

5.1.3 Selecting a Service Using Preferred Components

The tuner model defines an object of type *preferred_component*, and a list structure 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:

dsit_selection_specification for a service with preferred components	
address offset	contents
00_{16}	specification_length
01_{16}	information_type = service with specified components (83_{16})
02_{16}	system_specific_service_attributes_valid_flags
:	
:	system_specific_service_selection_attributes
:	
:	
:	**number_of_components (m) = FF_{16}**
:	selection_indicator
:	preferred_components_object_reference
:	
:	

All of the fields down to *system_specific_service_selection_attributes* 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).

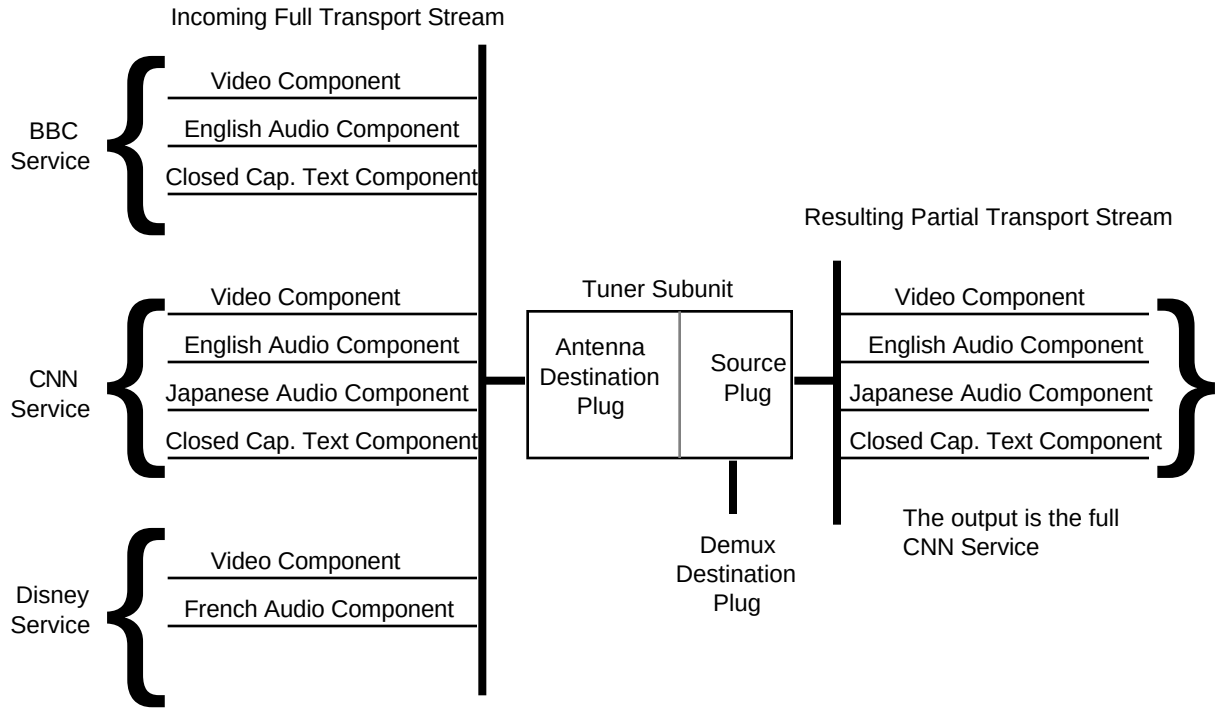
The *specifier_type_flag* of the *selection_indicator*, if set to 1, indicates that the preferred components object is referenced by its ID. If it is zero, then the preferred components object is referenced by its position in the preferred components list. When using a position reference, the object reference is 2 bytes. When using an object ID, the number of bytes to use will be indicated in 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.

5.1.4 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.

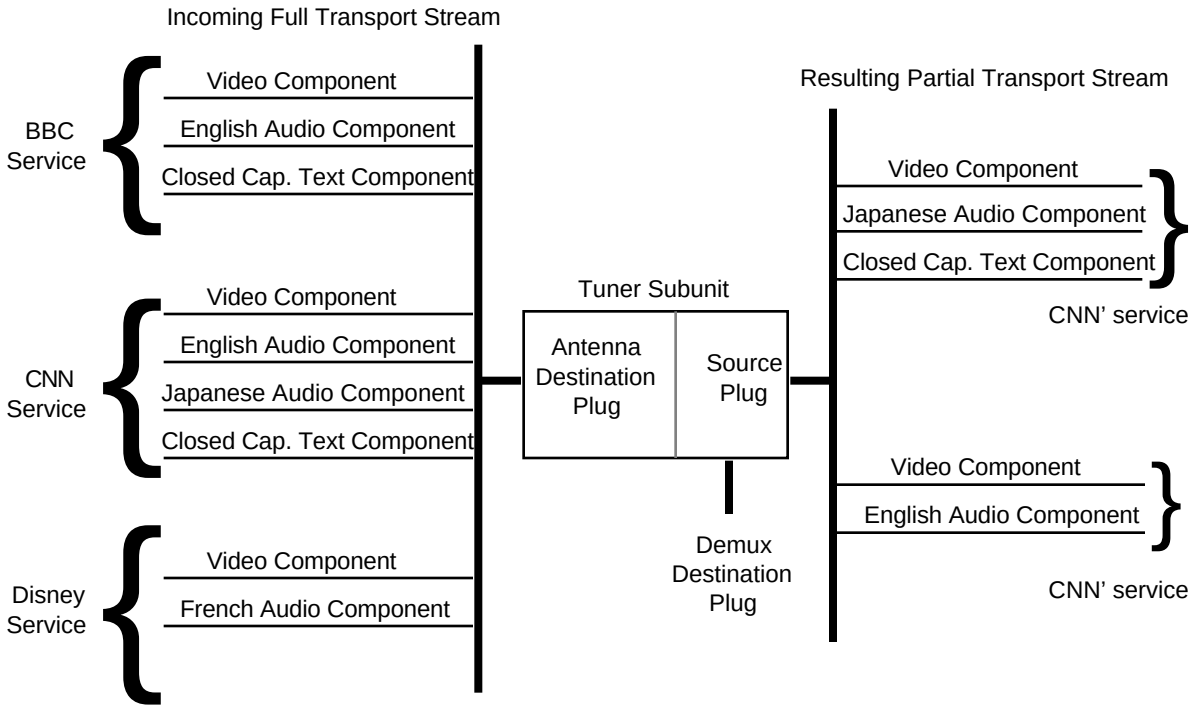
The way that these services are constructed is directly affected by the way selection information is presented to the tuner in the parameters of the DSIT command. Consider the following diagram, which shows a service being selected from the incoming transport stream:



DIRECT SELECT INFORMATION TYPE ($C8_{16}$)
source_plug
• • •
system_specific_multiplex_selection_attributes
number_of_dsit_selection_specifications (n) = 1
dsit_selection_specification[0] for the CNN 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 ($C8_{16}$)
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

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.

5.2 OBJECT NUMBER SELECT

For tuner subunits, the OBJECT NUMBER SELECT command deals with the defined information types lists (multiplex, service, component), and in addition, it deals with the preset lists. For details on the general OBJECT NUMBER SELECT command, please refer to the AV/C Descriptor Mechanism specification.

5.2.1 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.

5.2.2 The Tuner Subunit ons_selection_specification Structure

In addition to the general *ons_selection_specification* formats defined by the AV/C model, tuner subunits support an additional *ons_selection_specification*, which allows the controller to specify 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.

The *target* field of the *ons_selection_specification* can be specified as either a full path, or as a "don't care" path, as follows:

target field (full path specification) using preferred components	
address offset	contents
00_{16}	target_object_reference
:	
:	
:	number_of_children = FF_{16}
:	preferred_components_object_reference
:	
:	
:	

In this case, the *target_object_reference* indicates a service object which is being selected using 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*.

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:

target field ("don't care" specification) using preferred components	
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)

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

target field ("don't care" specification) using preferred components	
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)
:	
:	

Supporting the preferred components list is optional. If the list is not implemented, or if the specified object is not in the list, then the command shall be REJECTED.

5.2.3 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.

5.2.4 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.

5.2.5 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:

value	status description
00_{16}	No information instances are on the specified source plug.
10_{16}	The information instances listed in the following fields are currently on the specified source plug.
20_{16}	The information instances 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).

It is possible that a DIRECT SELECT INFORMATION TYPE command was issued which caused a particular information instance to be appended to the specified source plug, but this information instance is not represented in an object list. In this situation, the information instance will not be reported by the OBJECT NUMBER SELECT status command. The controller should examine the tuner status descriptor to get this information.

5.3 DIRECT SELECT DATA

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 titled Broadcasting and Tuning Concepts on page 9. 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]							
:								
:								

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

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*..

The *flowfunction* 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:

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 (see the normative references for details).

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. Also, the *system_id*, *input* and *antenna_number* fields will be ignored.

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 Descriptor (a subunit type-dependent descriptor) which begins on page 26.

5.4 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]	

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}							

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}							

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.

5.5 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 26 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}							

The *source_plug* operand specifies the 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 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 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 case an 'event' occurred, operand[1] will contain the *system_id* of the broadcasting system which caused the event and operand[2] will 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 this document.

6. 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*.

6.1 Universal Profile ID Assignments

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

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. 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. 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)

6.2 Notes on the command execution model for tuner subunits

Under certain conditions, it shall be possible for the tuner subunit to perform both tuning and demuxing functions independently from each other; this is illustrated by the diagram which shows a multiplex selection and a separate demuxing operation, near the beginning of this document. In this case, the following rules shall apply:

A tuner subunit is allowed to support input *only* from the demux destination plug. This avoids the definition of a separate subunit for those situations where a unit handles the demuxing of signals internally, e.g. during the playback of a DVB recording. However, tuner subunits must support at least one of the signal types defined for the broadcast systems specified by the AV/C Tuner Model, whether in this document or in future AV/C Tuner Broadcast System Specification documents.

Essentially, these rules state that if the tuner subunit is performing a multiplex selection operation which does not require the use of the demux, then it should also allow a controller to explicitly use the demuxing function as a separate operation. Additionally, a tuner subunit must not be created for the purpose of *only* performing non-broadcast system signal demuxing.

