

DRAFT
Common control interface
for networked digital audio and video products –
Part 5-1: Transmission over networks – General

Foreword

[EdNote] *To be added; see 5.2.1.3 and 6.1.3 in Directives Part 2*

0 Introduction

0.1 Structure of the family of standards

IEC 62379 specifies the Common Control Interface, a protocol for managing networked audiovisual equipment. It is intended to include the following Parts:

1. General
2. Audio
3. Video
4. Data
5. Transmission over networks

Part 1 specifies aspects which are common to all equipment, and includes an introduction to the Common Control Interface.

Parts 2 to 4 specify control of internal functions specific to equipment carrying particular types of live media. Part 4 ~~does not~~ refers to time-critical data such as commands to automation equipment, but not to packet data such as the control messages themselves.

Part 5 specifies control of transmission of these media over each individual network technology. It includes network specific management interfaces along with network specific control elements that integrate into the control framework.

Within part 5, sub-part 1 (this document) specifies management of aspects which are common to all network technologies. Sub-part 2 specifies protocols which can be used between networking equipment to enable the setting up of calls which are routed across different networking technologies. Sub-parts 3 onwards specify management of aspects which are particular to individual networking technologies.

~~An introduction to the Common Control Interface is given in Part 1.~~

0.2 Support for “future networks”

Most current networks use IP (v4 or v6), but there is a growing recognition that this form of packet switching, or, indeed, packet switching in general, has significant shortcomings for much of the traffic carried on the Internet. ~~Many of the problems with IP are a result of the lack of separation between the tasks of choosing a route and forwarding packets along that route~~ The service provided by IP networks is “best effort” transmission of individual packets submitted at unpredictable intervals, which is inappropriate for continuous media where transmission of a regular stream of packets with defined latency is required. More details are given in the Introduction to IEC62379-5-2.

0.3 Services provided by the network

This family of standards is mainly concerned with two kinds of service, one suitable for live media and one suitable for management messages. A network may well provide other kinds of service as well, but they are outside the scope of IEC 62379.

~~*[We should also standardise a service suitable for "best-effort" traffic other than management messages. That could mean file transfer (particularly of large files) or more generally "streamed" media over a "best effort" channel, as well as protocols such as HTTP.]*~~

[EdNote] The service for live media (including status broadcasts) is intended to be similar to the service provided by a cross-point router. Thus it is suitable for carrying a stream of data at a constant rate from a source to one or more destinations. Important Quality of Service (QoS) parameters include the maximum and minimum delay between source and destination and the likelihood of parts of the data being lost.

The concept of a "call" or virtual circuit is used. (Formal definitions are in clause 3 of Part 1.) Setting up and tearing down calls corresponds to making and breaking routes in a cross-point router. The normal procedure in router control is to "take" a source from a destination, and the model used in this standard is that the management terminal sends commands to the destination unit as described in 0.4.1 below, and the destination unit then asks the network to make the connection.

An interface unit may implement calls in very different ways, depending on the type of network over which the call passes; network-specific details are in the Sub-part of Part 5 which applies to each type of network. If the network does not support multicasting, the source may send a separate copy of the stream to each destination unit.

If the network offers a connection-oriented service, calls map naturally onto connections and the network can be expected to offer guarantees for the QoS parameters when a call is set up. If the network only offers a connectionless service, calls map onto sessions at a higher layer (for instance, RTP flows which are carried over the connectionless service provided by UDP over IP; note that there is still a "call set-up" process, using a protocol such as SIP), in which case it may only be possible to give an estimate of the QoS parameter values.

The service for management messages may be either connection-oriented or connectionless. In either case, it is assumed to be a "best effort" service which may lose messages or buffer them for an unlimited length of time. On IP networks, the connectionless protocol UDP is used (as specified in RFC1157), but on other networking technologies there may well be advantages in setting up a call, for instance if authentication can be done at call set-up time rather than separately for each message.

0.4 Network ports, **callsflows**, and media streams

0.4.1 Connectivity model

The physical connection to the network is described by a "block", as described in 0.3 of Part 1. The block has an input for each media flow going to the network, and an output for each media flow coming from it. The number of inputs and outputs will usually change dynamically as calls are connected and cleared down. This model is used in switches within the network as well as in end equipment.

CallsFlows of the same type are grouped together independently of the network port through which they pass, for instance the "unit destination list" (see 5.4.4) includes all ~~calls with~~ incoming media streams, and the "unit source list" (see 5.4.3) includes all ~~calls with~~ outgoing media streams.

To "take" the media stream from a remote source, a new entry is first created in the "unit destination list" of the unit that is to receive the stream; note that at this stage it may not be

known on which of its network ports the call will be connected, if it has more than one. When the management terminal has written all the necessary information in this entry, including identifying which input to which block is to receive the incoming media stream, it requests the unit to make the connection on the network. If the connection is successfully made, a new output is created (or an existing, currently unused, output is assigned) on the network port block representing the physical port on which the call was made, and the input that is to receive the media stream (on a block which may represent a media output port or an internal processing function) is connected to it.

If the remote source is already transmitting on the network, and the network supports multicasting, the network will simply copy the existing stream to the new unit. Otherwise, the source unit will receive an incoming call requesting a source which it identifies from information in the call set-up (or INVITE) message; it creates (or assigns) a new input to the relevant network port block and connects it to the source.

[EdNote] This subclause talks about "calls", "streams", and "flows" more or less interchangeably; we now have formal definitions for "call" and "flow" and "synchronous flow", but I think in the introduction it is OK to continue to use informal terms such as "media stream". I'm not sure whether it's possible for packets addressed to an IP address to arrive by more than one physical port. Presumably the equipment could choose to stop that happening by only responding to ARP packets on one physical port, but it might want to give the opportunity for load sharing. If it can happen, we need a note here to say that the internal IP-router counts as a separate "port".]

A "call" as defined here only carries one media stream. If the network supports calls which carry a bundle of several media streams, each stream has its own entry in the unit source or destination list. In this case, the "call identifier" will probably consist of a part that identifies the call (i.e. the bundle) and a part that distinguishes individual streams within the bundle. IEC62379-5-2 specifies a call identifier in this form.

0.4.2 Privilege

With each call is associated one of the four privilege levels described in 0.7 of IEC 62379-1. In increasing order of privilege they are listener, operator, supervisor, and maintenance. Calls with a given privilege level cannot be affected by management commands issued by a user, or via a management call, with a lower privilege.

0.4.3 Call identity

"Identity" information may be associated with any call, to provide a user with information that may not be apparent from, say, the network addresses of the endpoints. In a broadcast environment this could include an indication of whether it is part of an on-air programme chain, and if so which programme.

Included in this information are "importance" and "reconnection priority". Importance is associated with a destination, for instance a broadcast transmitter would have the highest importance; a PC on which the programme is being played would have a lower importance. When reporting the identity of a call which has a large number of destinations, only the higher-importance destinations are reported. Reconnection priority controls the order in which calls are reconnected if an equipment failure causes a large number of calls to fail simultaneously.

More details are given in A.4 of IEC 62379-1.

0.5 Control of routing

For some applications, including radio and TV broadcasting, calls are required to exist for very long periods with very high reliability. IEC 62379 includes facilities to assist that.

Where equipment is duplicated to increase reliability, the control system may request that callsroutes between the two sets of equipment follow different paths in the network, so that no part of the network can become a single point of failure. Destination equipment may receive two copies of the media by different routes.

This allows a call to be "replaced". The replacement callroute is connectedset up, then the destination equipment switches from using the data from the original callroute to the replacement, then the original call is torn down. The destination equipment needs to be able to align the two data streams so that there is no discontinuity when switching; ~~this e-means-for-doing-so (such as the timing specified in AES53) are outside the scope of Part 5~~ should be done using timing information in the data packets.

One use for call replacement is to allow networking equipment to be taken out of service for scheduled maintenance or relocation. This uses another facility, whereby the equipment can be "barred" from accepting new callsroutes. The existing callsroutes through it are then "replaced", and the replacement callsroutes will take a different routepath. Once all callsroutes through it have been replaced, the equipment can be taken out of service.

0.6 Scheduled calls

If the network supports it, calls to carry live media can be pre-scheduled. If a programme requires a feed from a remote studio or other remote location, the network can be requested in advance to reserve the necessary resources.

This requires all calls with guaranteed QoS to specify the maximum duration of the call, so that the network knows by when its resources will have been released. If this duration is exceeded, the network may disconnect the call if required to allow a pre-booked call to be made.

The MIB includes information to support this feature. Where the network requires to know the duration and it is not specified by the control system, a suitable default (tailored to the application) should be generated by the interface equipment.

Common Control Interface – Part 5-1: Transmission over networks – General

1 Scope

This International Standard specifies aspects of the Common Control Interface that are common to all network technologies, including setting up and tearing down of sessions and the service provided by the network.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[EdNote] tbd; I guess we need to list 62379-1 and RFCs 1157 and 3411; what else?]

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62379-1 and IEC 62379-5-2, and the following apply.

[EdNote] needs review, including 3.5; note that the above wording imports all the definitions etc in Parts 1 and 5-2; the different kinds of port should probably be replaced by a single definition of a “media port”.]

3.1 —

datagram

~~a message transmitted using a connectionless network service~~

3.2

media port

source or destination of media data in an interface unit; either a physical port (e.g. an external connector on the unit) or a logical port (e.g. an internal connection to another part of the unit)

3.3

link

a connection between two or more units that carries media data and other messages

3.4

switch

a unit which routes media data and other messages between links

3.5 Abbreviations

3.5.1

**application protocol data unit
APDU**

3.5.2

**information element
IE**

3.5.3

**service data unit
SDU**

3.5.4

**transmission control protocol
TCP**

3.5.5

**unspecified bit rate
UBR**

4 Network service specifications

4.1 Service for live media

Live media (including status broadcasts) shall be transmitted using a call for which, if the network supports it, guaranteed levels of throughput, delay, and data loss shall be requested.

4.2 Service for management messages

Management messages ~~may~~should be transmitted in data units on an asynchronous flow as specified in IEC 62379-5-2 using either a call or ~~If no such service is available, a connectionless datagram service such as UDP~~ may be used.

Where ~~calls are~~ connection-oriented service is used, at least one call at each privilege level shall be accepted by a destination unit at any given time. If more calls at one privilege level are accepted, this shall not prevent the acceptance of at least one call at each other privilege level.

5 MIB definitions applicable to all networks

5.1 General

The structure of the MIB shall be as specified in IEC 62379-1.

5.2 Type definitions

[EdNote 1] need to review this list; it should contain the definitions for this document and anything that can be expected to be useful in most of the other sub-parts of Part 5. It should not include anything that gets imported from Part 1.]

[EdNote 2] I have changed "Boolean" to "TruthValue" everywhere, also "DataFormat" to "MediaFormat". I hope that's right!]

The following application-wide types shall be used:

```
NetPortState ::= INTEGER {
    disabled          (1),
    closing           (2),
    linkDown          (3),
    linkUp            (4), -- does not know to what linked
    pointToPoint      (5), -- link is point-to-point
    peerGroup         (6), -- e.g. Ethernet hub
    sharedMedia       (7)  -- with master, e.g. 802.11
} (disabled.. sharedMedia)

PortIdentifier ::= OCTET STRING (SIZE(3))
' octet 1 = port type
' octets 2 and 3 = port number (high byte in octet 2)
```

[EdNote 3] A unit might well have more than 255 ports of the same type, e.g. audio inputs to a mixing desk; I don't think more than 65535 is likely, though. We need to define the "port type" codes.]

```
ConnectionEnd ::= INTEGER {
    source            (1),
    destination       (2)
} (source..destination)
```

CauseCode ::= OBJECT IDENTIFIER

[EdNote 4] Need to define these somewhere; probably need generic ones here or in Sub-part 2 and network-specific ones in the other Sub-parts. Zero length means "normal call clearing".]

```

ConnectionState ::= INTEGER {
  readyToConnect      (1),
  connectionRequested (2),
  terminating         (3),
  active              (4),
  failed              (5),
  disconnected         (6),
  pending             (7),
  inactive            (8),
  finished            (9),
  callProceeding     (10),
  receivedOffer       (11),
  acceptedOffer       (12),
  reservationRequested (13),
  clearing             (14)
} (readyToConnect..clearing )

```

Importance ::= INTEGER (1..255)

Priority ::= INTEGER (1..255)

5.3 Conceptual row type definitions

The following types are used to specify the syntax of managed objects in this standard that represent conceptual table rows.

[These were in the same subclause as the other type definitions in the earlier draft; I have changed names ending in "Info" to "Entry".]

[EdNote] Needs updating to match new versions of Tables- 1-4]

```

NodeRouteEntry ::= SEQUENCE {
  nrAddrType      TDomain,
  nrAddress        TAddress,
  nrLocal          TruthValue,
  nrBarred         TruthValue
}

UnitPrivilegeEntry ::= SEQUENCE {
  upLevel          PrivilegeLevel,
  upCallPoolSize  CardinalNumber
}

UnitConnectionEntry ::= SEQUENCE {
  ucCallIdentifier OCTET STRING,
  ucType           ConnectionType,
  ucEnd            ConnectionEnd,
  ucRemoteAddrType TDomain,
  ucRemoteAddress  TAddress,
  ucLocalPort      PortIdentifier,
  ucMediaFormat    MediaFormat,
  ucPackageSize    CardinalNumber,
  ucPackageRate    CardinalNumber,
  ucPrivilege      PrivilegeLevel,

```

```

ucState           ConnectionState,
ucRemembered      TruthValue,
ucSource           Utf8String,
ucDestination     Utf8String,
ucService         Utf8String,
ucImportance      Importance,
ucPriority         Priority,
ucStartTime       DateTime,
ucEndTime         DateTime,
ucConnectTime     CardinalNumber,
ucLastCause       OCTET STRING
    }
    
```

5.4 MIB object definitions

~~[I have deleted the original 5.4.1. It was a relic from the ATM-specific system; the function which was described in the Note to 5.4.1.6 is now covered by 5.4.2, and the PNNI-related aspects were network-specific.]~~

5.4.1 Information about management agents

~~[EdNote] This really ought to have been in Part 1; do we want to leave it out altogether for now and add it to Part 1 at the next revision? Is there any switch-specific information that we should include here instead?]~~

The group of objects in Table 1 should be implemented by all pieces of equipment that accept management calls. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 network(5) general(1) generalMIB(1) management(1) }
```

~~NOTE: Equipment that requires management messages to be delivered by a connectionless datagram service need not implement these objects.~~

Table 1: Managed objects conveying information about management agents

| Identifier | Syntax | Index | Readable | Writable | Volatile | Syntax |
|----------------------------|-----------------------------------|-------|----------|------------|----------|-----------|
| managementCallPoolSize (1) | IndexNumber | | listener | none | no | mandatory |
| unitPrivilegeList (2) | SEQUENCE OF UnitPrivilegeEntry | | none | none | no | mandatory |
| ↳unitPrivilegeEntry (1) | UnitPrivilegeEntry | | none | none | no | mandatory |
| ↳upLevel (1) | PrivilegeLevel | yes | listener | none | no | mandatory |
| ↳upCallPoolSize (2) | IndexNumber | | listener | supervisor | no | mandatory |
| ↳upCallsPresent (3) | CardinalNumber | | listener | none | yes | mandatory |

5.4.1.1 managementCallPoolSize

The maximum number of simultaneous management calls the unit can support.

5.4.1.2 unitPrivilegeList

The list of privilege level descriptors for this unit. There is one entry for each possible privilege level.

5.4.1.3 unitPrivilegeEntry

An entry in the list of privilege level descriptors for this unit.

5.4.1.4 upLevel

The privilege level described by this entry.

5.4.1.5 upCallPoolSize

The number of simultaneous management calls reserved for this privilege level. If the sum of the values in this object for all privilege levels is less than the value in `managementCallPoolSize`, additional calls can be accepted at any level. Changes to this value shall only affect new connection requests; existing calls shall be maintained.

5.4.1.6 upCallsPresent

The number of management calls currently connected at this privilege level.

5.4.2 Network ports

Each physical connection to a network, in a switch or end equipment, shall be represented using a network port block. A network port block shall have one input for each call that is transmitting a media stream and one output for each call that is receiving a media stream.

The group of objects in table 2 shall be implemented by all switches. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 network(5) general (1) generalMIB(1) networkPorts(2) }
```

This node shall be used as the block type identifier for network port blocks.

Table 2: Managed objects for network ports

| Identifier | Syntax | Index | Readable | Writable | Volatile | Status |
|-----------------------|------------------------|-------|----------|------------|----------|--------|
| netPortTable (1) | SEQUENCE OF APortEntry | | none | none | yes | m |
| ↳netPortEntry (1) | APortEntry | | none | none | yes | m |
| ↳nPortBlockId (1) | BlockId | yes | none | none | no | m |
| ↳nPortName (2) | Utf8String | | listener | supervisor | no | m |
| ↳nPortState (3) | NetPortState | | listener | supervisor | yes | m |
| ↳nPortAddressType (4) | TDomain | | listener | none | no | m |
| ↳nPortAddress (5) | TAddress | | listener | none | no | m |
| ↳nPortPAddrType (6) | TDomain | | listener | none | yes | m |
| ↳nPortPartnerAddr (7) | TAddress | | listener | none | yes | m |
| ↳nPortBarred (8) | TruthValue | | listener | supervisor | no | m |

5.4.2.1 nPortTable

A table of network port descriptors for this unit. Each physical network port on the unit shall have a corresponding entry in this table. There may also be entries for "virtual" network ports.

5.4.2.2 nPortEntry

An entry in the network port table.

5.4.2.3 nPortBlockId

The block identifier for this port. Used as an index when accessing the network port table.

5.4.2.4 nPortName

The name assigned to this port. This is an arbitrary text string assigned by the system manager. Such assignment should persist across resets of the unit.

Until a name has been assigned, this object shall have a value that relates to a visible marking associated with the port's physical connector.

EXAMPLE: The name of a port on a network switch whose connector is labelled "2" on the unit's enclosure should default to a value such as "Port 2" or "Ethernet port 2" or "Front panel port 2". If it is connected to a network socket in, say, studio 6, a supervisor may then rename it as "Studio 6".

5.4.2.5 nPortState

The current link-layer state of the port's network connection.

If a management terminal sets this object to `closing` or `linkDown`, the managed unit shall reroute or, if that is not possible, gracefully close down all calls that pass through the port. In the case of `closing`, the port shall then enter the `disabled` state.

For as long as the port is in `linkDown` state, the managed unit shall attempt to establish a network connection on the port.

5.4.2.6 nPortAddressType

The type of network address used for `nPortAddress`.

5.4.2.7 nPortAddress

An address which identifies the port.

NOTE: This will normally be the 48-bit MAC address of the interface. An IP address may be used if it is permanently assigned, but not if it is acquired via DHCP~~identifies the equipment, not the interface, so cannot be used if the equipment has more than one interface.~~

5.4.2.8 nPortPAddrType

The type of network address used for `nPortPartnerAddr`.

5.4.2.9 nPortPartnerAddr

In `pointToPoint` state, the address of the unit to which the port is connected.

In `sharedMedia` state, the address of the unit which controls the local network to which the port is connected, e.g. wireless base station or clock master.

In other states, the `nPortPartnerAddr` value is not defined by this standard.

NOTE 1: This object is intended to allow a management terminal to "crawl" a network to discover its topology and what resources are present. The address allows it to make a management connection to the link partner in the case of a point-to-point link, or, in the case of a shared-media network segment, to a unit which may be able to supply a list of all the units on the segment. In contrast to `nPortAddress`, the `nPortPartnerAddr` value should identify the unit rather than its interface, so an EUI-64 ~~or an IP address~~ is appropriate.

NOTE 2: For some kinds of network segment, such as an Ethernet segment using CSMA/CD, there is no straightforward method for enumerating all the units present on the segment.

5.4.2.10 nPortBarred

`false` (the default) if the unit is allowed to connect a ~~call, or forward a datagram, route~~ via the port; `true` if forbidden.

5.4.3 List of ~~live~~ media sources ~~calls~~

The “list of sources” has an entry for each synchronous flow transmitted by the managed unit.

The group of objects in Table 3 shall be implemented by all end equipment that can be the source for ~~live media~~ a synchronous calls flow, and by all switches. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 network(5) general(1) generalMIB(1) callSources(3) }
```

NOTE: Calls are always connected by the destination, so this table is read-only, apart from the ability to clear down a call from the source end. Management calls are not included in this list.

[EdNote] We assume The assumption is that incoming calls specify a source in some way that is may be at least partially network-dependent, and whenever a new connection is made an entry appears in this table, disappearing again when the call is released.]

Table 3: Managed objects conveying the list of sources

| Identifier | Syntax | Index | Readable | Writable | Volatile | Syntax |
|---|------------------------------------|-------|----------|----------|----------|-----------|
| unit Call SourceList (1) | SEQUENCE OF UnitCallSourceEntry | | none | none | yes | mandatory |
| Unit Call SourceEntry (1) | UnitCallSourceEntry | | none | none | yes | mandatory |
| f usFlow Call Identifier (1) | OCTET STRING | yes | none | none | yes | mandatory |
| f usBlockId (2) | SourceBlockId | yes | listener | none | yes | mandatory |
| f usBlockInput (3) | IndexNumber | | listener | none | yes | mandatory |
| f usPackageSize (4) | CardinalNumber | | listener | none | yes | mandatory |
| f usPackageRate (5) | CardinalNumber | | listener | none | yes | mandatory |
| f usPrivilege (6) | PrivilegeLevel | | listener | none | yes | mandatory |
| f usState (7) | ConnectionState | | listener | see 6.3 | yes | mandatory |
| f usCause (8) | CauseCode | | listener | see 6.3 | yes | mandatory |
| f usSource (9) | Utf8String | | listener | none | yes | mandatory |
| f usDestination (10) | Utf8String | | listener | none | yes | mandatory |
| f usService (11) | Utf8String | | listener | none | yes | mandatory |
| f usImportance (12) | Importance | | listener | none | yes | mandatory |
| f usPriority (13) | Priority | | listener | none | yes | mandatory |
| f usStartTime (14) | DateTime | | listener | none | yes | mandatory |
| f usEndTime (15) | DateTime | | listener | none | yes | mandatory |
| f usConnectTime (16) | CardinalNumber | | listener | none | yes | mandatory |
| f usFlowIdStandard | TruthValue | | listener | none | yes | mandatory |

5.4.3.1 unit~~Call~~SourceList

The list of ~~calls~~ flows for which this unit transmits media data.

NOTE: In the case of end equipment, the table list ~~calls~~ flows for which the unit is the source. In the case of a switch, it lists information relating to onwards transmission towards the destination(s), for all ~~media~~ ~~calls~~ synchronous flows passing through the unit.

5.4.3.2 unit~~Call~~SourceEntry

An entry in the list of calls for which this unit transmits media data.

5.4.3.3 **usCallFlowIdentifier**

An octet string which identifies the **callflow**. The format ~~is outside the scope of this document~~ specified in IEC 62379-5-2 should be used if available; see 5.4.3.19.

~~NOTE: The management terminal should regard the call identifier simply as an index to this table, chosen by the managed unit. The managed unit may use the format specified in IEC62379-5-2, in which case a call has the same identifier in every piece of equipment through which it passes; it may also use a format related to the underlying network technology, in which case a call can have different identifiers in different units, or on different ports of the same unit, and the same identifier on different ports or in different units might refer to different calls. Thus the management terminal should not assume any relationship between calls on different ports in this table, or between calls in this table and in the table specified in 5.4.4, based on a comparison of their identifiers; relationships should be established via the connector table specified in 4.2.5 of IEC62379-1.~~

~~[EdNote] This table was originally defined for ATM networks, on which a call can only carry one flow; now we have the possibility of a call carrying multiple flows, and it is the flows that this table should list. I have changed “call” to “flow” in the object names; this is a good time to do that because we don’t have text for Annexes A and B yet. [We should maybe provide an indication whether the value is a call (or flow) identifier as specified in IEC62379-5-2, so that the management terminal can know whether its significance is global or merely local.]~~

5.4.3.4 **usBlockId**

The identifier of the network port block for the unit's network port through which this **callflow** passes.

5.4.3.5 **usBlockInput**

The input number, to the network port identified by **usBlockId**, through which this **callflow** passes.

NOTE: The entry associated with this input in the connector table shows the block output which is the source of the media stream. The entry associated with that output in the mode table shows the media format being transmitted,

5.4.3.6 **usPackageSize**

The maximum number of payload octets that may be transmitted in a single data unit on the **callflow**.

~~NOTE This is the size negotiated for the call, which when multiplied by the **usPackageRate** value defines the bandwidth required; it is not the maximum transmission unit size for the links over which the flow will be transmitted.~~

~~EXAMPLE 1: For an ATM link this would be fixed at the value 48, the number of payload octets in a cell.~~

~~EXAMPLE 2: For an RTP stream this would be the maximum payload size for the RTP packets.~~

5.4.3.7 **usPackageRate**

The number of data units per second that may be transmitted on the **callflow**.

5.4.3.8 **usPrivilege**

The privilege level associated with this **callflow**, which is the highest of the privilege levels associated with its destinations if the network provides that information, `supervisor` otherwise.

5.4.3.9 **usState**

The current state of this **callflow**. The `callProceeding` state shall indicate that an incoming connection request has been accepted and confirmation from the caller is awaited.

5.4.3.10 **usCause**

This object is initialised to “normal call clearing” and is set by the managed unit when it changes `usState` to `failed` or `disconnected`. See also 6.3.

5.4.3.11 usSource

The source name for this [callflow](#).

NOTE: In the case of end equipment, the source name is specified as part of the definition of one of the blocks through which the signal passes on its way to the network port. In the case of a switch, it is inherited from the call's `udSource` object (see 5.4.4.15).

[EdNote] In the case of audio connections it ought to appear somewhere in Part 2 but I don't see it there; we ought to fix that.]

5.4.3.12 usDestination

The destination name for the most important destination of the [callflow](#) reached via this network port.

5.4.3.13 usService

The service name for the most important destination of the [callflow](#) reached via this network port.

5.4.3.14 usImportance

The importance of the most important destination of the [callflow](#) reached via this network port.

5.4.3.15 usPriority

The priority of the part of the [callflow](#) connected via this network port.

5.4.3.16 usStartTime

The time at which the state is expected to change from `pending` to `reservationRequested` or `active`.

5.4.3.17 usEndTime

The time after which the network may release the resources reserved for the [callflow](#) on this port.

5.4.3.18 usConnectTime

The number of seconds for which this [callflow](#) has been `active` on this network port or the maximum `CardinalNumber` value, whichever is the less; zero if the state is not `active`.

5.4.3.19 [usFlowIdStandard](#)

`true` if `usFlowIdentifier` is in the format specified in IEC62379-5-2, `false` otherwise.

NOTE 1: If the format specified in IEC62379-5-2 is used, a flow has the same identifier in every piece of equipment through which it passes. Thus records in which `usFlowIdStandard` or (in the list of destinations, see 5.4.4.27) `udFlowIdStandard` is `true`, even in different managed units, refer to the same call if, and only if, they have the same `usFlowIdentifier` or `udFlowIdentifier` value.

NOTE 2: If `usFlowIdStandard` is `false`, the management terminal should regard the flow identifier simply as a value chosen by the managed unit which, when combined with `usBlockId`, forms an index to this table. Thus the management terminal should not assume any relationship between calls on different ports in this table, or between calls in this table and in the table specified in 5.4.4, based on a comparison of their identifiers; relationships should be established via the connector table specified in 4.2.5 of IEC62379-1.

5.4.4 List of live media destination-calls

The “list of destinations” has an entry for each synchronous flow received by the managed unit.

The group of objects in Table 4 shall be implemented by all end equipment that can be the destination for live-media synchronous callsflows, and by all switches. The root node for these objects shall be

```
{ iso(1) standard(0) iec62379 network(5) general(1) generalMIB(1) callDestinations(4) }
```

NOTE 1: Management calls are not included in this list. Media-calls Synchronous flows are always connected by the destination, so in the case of end equipment every entry in this list is a callflow which has been initiated by a management terminal through the process detailed in clause 6.

Table 4: Managed objects conveying the list of destinations

| Identifier | Syntax | Index | Readable | Writable | Volatile | Syntax |
|------------------------------------|----------------------------------|-------|-----------------|----------------|--------------|--------------------|
| unitNextCallFlowId (1) | OCTET STRING | | listener | none | yes | see note 3 |
| <u>unitNextCallId (2)</u> | <u>OCTET STRING</u> | | <u>listener</u> | <u>none</u> | <u>yes</u> | <u>see 5.4.4.2</u> |
| unitCallDestList (3 ₂) | SEQUENCE OF UnitCallDestEntry | | none | none | yes | mandatory |
| UnitCallDestEntry (1) | UnitCallDestEntry | | none | none | yes | mandatory |
| FlowCallIdentifier (1) | OCTET STRING | yes | none | none | maybe | mandatory |
| udNetBlockId(2) | SourceBlockId | | listener | see 6.1 | yes | mandatory |
| udNetBlockOutput(3) | IndexNumber | | listener | none | yes | mandatory |
| udSourceAddrType (4) | TDomain | | listener | see 6.1 | maybe | mandatory |
| udSourceAddress (5) | TAddress | | listener | see 6.1 | maybe | mandatory |
| udPackageSize (6) | CardinalNumber | | listener | none | maybe | mandatory |
| udPackageRate (7) | CardinalNumber | | listener | none | maybe | mandatory |
| udPrivilege (8) | PrivilegeLevel | | listener | see 6.1 | maybe | mandatory |
| udState (9) | ConnectionState | | listener | see 6.1 | yes | mandatory |
| udCause (10) | CauseCode | | listener | see 6.1 | yes | mandatory |
| udSource (11) | Utf8String | | listener | none | maybe | mandatory |
| udDestination (12) | Utf8String | | listener | see 6.1 | maybe | mandatory |
| udService (13) | Utf8String | | listener | see 6.1 | maybe | mandatory |
| udImportance (14) | Importance | | listener | see 6.1 | maybe | mandatory |
| udPriority (15) | Priority | | listener | see 6.1 | maybe | mandatory |
| udStartTime (16) | DateTime | | listener | see 6.1 | maybe | mandatory |
| udEndTime (17) | DateTime | | listener | see 6.1 | maybe | mandatory |
| udConnectTime (18) | CardinalNumber | | listener | none | maybe | mandatory |
| udConnectCount (19) | CardinalNumber | | listener | supervisor | maybe | mandatory |
| udRemembered (20) | TruthValue | | listener | see 6.1 | no | see note 3 |
| udDestBlockId(21) | SourceBlockId | | listener | see 6.1 | maybe | see note 3 |
| <u>udDestBlockInput(22)</u> | <u>IndexNumber</u> | | <u>listener</u> | <u>see 6.1</u> | <u>maybe</u> | <u>see note 3</u> |
| <u>udFlowIdStandard</u> | <u>TruthValue</u> | | <u>listener</u> | <u>none</u> | <u>maybe</u> | <u>mandatory</u> |

NOTE 2: Where the volatility is shown as "maybe", the object is volatile if, and only if, udRemembered is false.

NOTE 3: Objects shown as “see note 3” are mandatory for end equipment but should return noSuchName in units that cannot be the destination of a media callflow.

5.4.4.1 unitNextFlowCallId

A callflow identifier value, not in the form specified in IEC 62379-5-2, for which a new record is created in the table as described in 6.1. Consecutive Get requests for this object shall return different values; values should be chosen in a way that maximises the time before re-use of any particular value. A GetNext request for this object shall not create a new record; the value returned (if any) is not defined by this standard.

NOTE 1 Whereas the port number is an index into the list of sources, so source flow numbers only need to be unique for each port, in the list of destinations the flow number is the only index, and therefore needs to be unique across all ports of the managed unit.

Managed units that support flow identifiers in the format specified in IEC 62379-5-2 should internally allocate a call identifier as if `unitNextCallId` had been requested. The value returned may be the call identifier, or a flow identifier based on it, or some other value which refers to it. In any case, the record that is created shall have `udFlowIdStandard` set to `false`, and the value returned shall be different from any flow identifier in the format specified in IEC 62379-5-2 that could be created using a call identifier returned by reading `unitNextCallId`.

NOTE 2 To meet this requirement, it is sufficient for the octet string to be a different length from the format specified in IEC 62379-5-2, or to begin with the internally allocated call identifier.

5.4.4.2 `unitNextCallId`

A new call identifier value in the form specified in IEC 62379-5-2, from which flow identifiers may be derived as described in 6.1. Consecutive Get requests for this object shall return different values; values should be chosen in a way that maximises the time before re-use of any particular value. The value (if any) returned for a GetNext request for this object is not defined by this standard.

This object is mandatory for end equipment that supports flow identifiers in the format specified in IEC 62379-5-2, but shall return `noSuchName` in units that do not support that format and in units that return `noSuchName` for `unitNextFlowId`.

5.4.4.3 `unitCallDestList`

The list of `callsynchronous flows` for which this unit receives ~~media data~~.

NOTE: In the case of end equipment, the table lists `callsflows` for which the unit is the destination. In the case of a switch, it lists information relating to reception from the source, for all media `callsflows` passing through the unit.

5.4.4.4 `unitCallDestEntry`

An entry in the list of ~~live media~~ `callsynchronous flows` for which this unit receives ~~media data~~.

5.4.4.5 `udCallFlowIdentifier`

An octet string which identifies the `callflow` uniquely within the unit. ~~The format is outside the scope of this document~~ See also 5.4.4.27.

NOTE: This is the only index field, so must be unique within the unit, not merely for each port. ~~Apart from that provision, the Note to 5.4.3.3 applies.~~

5.4.4.6 `udNetBlockId`

In `inactive` and `readyToConnect` states, the identifier of the network port block for the unit's network port through which this `callflow` should be connected, or `nullBlock` if the network port is to be chosen by the managed unit. The managed unit may ignore this value even if it is not `nullBlock`.

In other states, the identifier of the network port block for the unit's network port through which this `callflow` passes, or zero if the `callflow` is not associated with a specific network port.

5.4.4.7 `udNetBlockOutput`

The output number, of the network port identified by `udNetBlockId`, through which this `callflow` passes, or zero if `udNetBlockId` is zero or the output has not yet been chosen.

NOTE: The entry associated with this output in the mode table shows the media format being received. The connector table specified in 4.2.5 of IEC62379-1 shows to which block inputs (if any) the received media stream is being routed internally. In the case of a switch, these blocks will be the network ports on which the call is being output.

5.4.4.8 udSourceAddrType

The type of network address used for `ucSourceAddress`.

5.4.4.9 udSourceAddress

The network address of the source.

5.4.4.10 udPackageSize

The maximum number of payload octets that may be transmitted in a single data unit on the call.

NOTE: See examples in 5.4.3.6.

5.4.4.11 udPackageRate

The number of data units per second that may be transmitted *on*for the *call*flow.

5.4.4.12 udPrivilege

The privilege level associated with this *call*flow.

5.4.4.13 udState

The current state of this *call*flow.

5.4.4.14 udCause

This object is initialised to “normal call clearing” and is set by the managed unit when it changes `udState` to `failed` or `disconnected`. See also 6.3.

5.4.4.15 udSource

The source name for this *call*flow.

NOTE: this object is not writable by the management terminal. The source name is inherited from the source of the *call*flow.

5.4.4.16 udDestination

The destination name for this *call*flow.

[EdNote] Maybe this (and the service name) ought to be read-only here, and inherited from the downstream block. In the case of audio connections it ought to appear somewhere in Part 2 but I don't see it there, as with usSource.]

5.4.4.17 udService

The service name for this *call*flow.

5.4.4.18 udImportance

The importance for this *call*flow.

5.4.4.19 udPriority

The reconnection priority for this [callflow](#).

5.4.4.20 udStartTime

This object shall be initialised to a zero-length octet string. The managed unit may set it to the time at which the [callflow](#) should be connected; see 6.2.

5.4.4.21 udEndTime

The time after which the network may release the resources reserved for this [callflow](#). A zero-length octet string indicates that the [callflow](#) is to remain connected indefinitely.

5.4.4.22 udConnectTime

The number of seconds for which this [callflow](#) has been `active` or the maximum `CardinalNumber` value, whichever is the less; zero if the state is not `active`.

[EdNote] Maybe we don't need "or the maximum CardinalNumber value, whichever is the less", because it will take about 68 years to reach the limit.

5.4.4.23 udConnectCount

The number of times this [callflow](#) has been `madeconnected` (includes both rerouting and automatic reconnection). ~~A `callConnection` of a flow~~ is only counted when it has been `active` for at least one minute. The count may be reset by writing the value zero; writing any other value shall be rejected with the `badValue` error code.

[EdNote] This was on the wish list for the Radio 4 project; is it really a requirement or can we delete it? Or change it's status from "mandatory" to "optional"?

5.4.4.24 udRemembered

If `true`, indicates this is a "remembered" [callflow](#) which should be reconnected after a disconnection by the network (for instance as a result of a link failure) or a reset of the managed unit.

[EdNote] ~~deleted note re switches because for them this object should not exist~~ NOTE: ~~In the case of a switch, there is no mechanism to set this object to anything other than false.~~

5.4.4.25 udDestBlockId

When `udState` is `readyToConnect` or `active`, the identifier of the block to which this [callflow](#) should be internally connected, or `nullBlock` if no internal connection is to be made. The value in other states is not defined by this standard.

NOTE: There are two ways to connect an input to a block to an external source. The input can be identified by `udDestBlockId` and `udDestBlockInput`, in which case the managed unit makes the internal connection, or these objects can be left as zero and the internal connection made explicitly after the managed unit has set `udNetBlockOutput` to a non-zero value. The former is a simpler process for the management terminal; the latter allows the management terminal to check the format being received before making the internal connection and, if the destination block already has a connection, to wait until a stream is being received on the new connection before making the switch.

5.4.4.26 udDestBlockInput

The input number, of the block identified by `udDestBlockId`, to which this call should be internally connected.

The value of this object shall be ignored by the managed unit when connecting a **callflow** if `udDestBlockId` is `nullBlock`, also if the block identified by `udDestBlockId` only has one input.

5.4.4.27 **udFlowIdStandard**

true if `udFlowIdentifier` is in the format specified in IEC62379-5-2, false otherwise.

NOTE 1: If the format specified in IEC62379-5-2 is used, a flow has the same identifier in every piece of equipment through which it passes. Thus records in which `udFlowIdStandard` or (in the list of sources, see 5.4.3.19) `usFlowIdStandard` is true, even in different managed units, refer to the same call if, and only if, they have the same `udFlowIdentifier` or `usFlowIdentifier` value.

NOTE 2: If `udFlowIdStandard` is false, the management terminal should regard the flow identifier simply as a value chosen by the managed unit which forms an index to this table. Thus the management terminal should not assume any relationship between calls on different ports in this table, or between calls in this table and in the table specified in 5.4.3, based on a comparison of their identifiers; relationships should be established via the connector table specified in 4.2.5 of IEC62379-1.

[EdNote] Other calls: I had in mind there should also be a table of the "best effort" bidirectional point-to-point calls passing through a switch. Do we want these calls to have a "call identity"? If not, and assuming the node won't want to remember the calling and called addresses, the only information that might go in a table entry would be the call identifier and internal routing, which is mostly network-specific so the table should be specified in the network-specific sub-parts. The original idea of call identity was to identify programme streams as in "source = studio B3, service = Radio 4 LW, destination = Droitwich transmitter", so would only apply to "live media" streams.]

6 Calls

6.1 List of destinations in end equipment

When a unit sends a `GetResponse` which includes a `unitNextCallFlowId` value in reply to a `Get` request, it shall create a "new" entry in its `unitCallDestList` for that value in which with `udFlowIdStandard` set to false.

In this subclause, a "new" entry is one in which all numeric objects not otherwise specified are zero and all octet-string and object-identifier objects are zero-length, except that with `udPrivilege` shall be set to the privilege of the sender of the `Get` request, `udState` shall be inactive, and `udRemembered` shall be false.

When a unit receives a `Set` request for a column in an entry in its `unitDestList` which does not exist, but where the index is a valid flow identifier in the form specified in IEC 62379-5-2, and includes a call identifier which has previously been returned as a reply to a `Get` request for `unitNextCallId`, it shall create a "new" entry in its `unitDestList` for that index with `udFlowIdStandard` set to true.

A management terminal shall not send a `Set` request which could create an entry as described in the preceding paragraph unless either it has established that the entry already exists or the index is a flow identifier in the form specified in IEC 62379-5-2 in which the call identifier value is one that it has previously received in reply to a `Get` request for `unitNextCallId`.

NOTE 1 If the managed unit receives a `Set` request with an index which could be an IEC 62379-5-2 flow identifier which includes a call identifier which has not previously been returned as a reply to a `Get` request for `unitNextCallId`, it should reply `noSuchName`. However, it may also create a new entry without checking the call identifier.

The managed unit shall delete an entry when its `udState` is set to `finished`, and may delete an entry for which `udRemembered` is `false` and `udState` is `failed` or `disconnected` or `inactive` after a time-out which shall be at least 5 minutes.

An end unit shall refuse a Set request for any of the objects whose "writable" privilege is shown in Table 4 as "see 6.1" unless the request has a privilege which is at least as high as the current `udPrivilege` value for the entry. In the case of objects other than `udState`, `udDestination`, `udService`, `udImportance`, and `udPriority`, the request shall also be refused if the current `udState` is not `inactive`. In the case of `udState`, the request shall also be refused unless it changes the value from `inactive` to `readyToConnect`, or from any value to `terminating`, or from `failed` to `inactive` or `readyToConnect`, or from `failed` or `disconnected` or `inactive` to `finished`.

NOTE 24: An operator, for example, can interfere with calls set up by other operators and by listeners but not with those set up by supervisors. It is assumed that an operator in one area will not have access to, and therefore not be able to interfere with, equipment in other areas.

In the case of a unit that cannot be the destination of a media-call synchronous flow (such as a switch, unless it also has media ports), objects whose "writable" privilege is shown in Table 4 as "see 6.1" shall be read-only except as specified in 6.3.

NOTE 32: This means that callflows can only be connected from the destination, not from an intermediate point in the route. It also means that callflows can only be "remembered" at a destination.

6.2 Connecting a callflow

A management terminal requiring to connect a callflow shall first acquire an identifier for the flow, either by reading the destination unit's `unitNextCallFlowId` object to acquire an identifier for the call or by constructing one from a call identifier issued to it by the destination unit. It shall then Set the objects in the `unitCallDestEntry` for that identifier to values appropriate to the call.

In the last such Set request (if there is more than one) the management terminal shall set the `udState` to `readyToConnect`.

When `udState` is set to `readyToConnect`, if `udStartTime` is a zero-length octet string the managed unit shall attempt to connect the callflow. During the connection process, `udState` may take other values such as `callProceeding`, depending on the signalling procedures used; these values may be further specified in IEC62379-5-2 or the sub-part of IEC62379-5 which applies to the network used for the call.

[EdNote] The current draft of 62379-5-2 does not specify any call/flow states.

NOTE If `udFlowIdStandard` is true, the procedures specified in IEC 62379-5-2 should be used: the initial request will be an `AddFlow` if the route is already connected, `FindRoute` else. Otherwise, procedures specific to the network technology should be used.

If the callflow is successfully connected, the managed unit shall set the value of `udState` to `active`. If the call is refused, the unit shall set the value of `udState` to `failed` and the value of `udCause` to the cause code returned by the network. The management unit may then set the value of `udState` to `inactive` and then change any of the other objects in the `unitCallDestEntry` as appropriate to reattempt the call (for instance, to try an alternative address). It may also change `udState` directly to `readyToConnect`, to reattempt with the same parameters. If the cause is indicated to be temporary, and `udState` is still `failed` after a minimum of $(256 - udPriority)/10$ seconds, then the managed unit shall reattempt to establish the call.

If `udStartTime` is a valid `DateTime` object when `udState` is set to `readyToConnect`, if the network supports pre-reservation the managed unit shall request that resources be reserved which will allow the call to be connected at the indicated time. On successful completion of this process it shall set `udState` to `pending`. If the network does not support pre-reservation the managed unit shall set `udState` to `pending` immediately, and then attempt to connect the call at a time such that the latest time the transition to `active` can occur if the call is successfully connected is at the indicated time.

6.3 Terminating a **callflow**

When, as a result of a Set request, the `udState` of an entry in a unit's `unitCallDestList` is set to `terminating`, the managed unit shall request the network to disconnect the corresponding **callflow**, supplying the value in `udCause` as the cause. On completion of the disconnection, the unit shall set `udState` to `failed` if an error was signalled by the network, otherwise to `disconnected`. If there is no corresponding **callflow** (for instance, if the previous state was `inactive`) the unit shall set `udState` either to `failed` (also setting an appropriate value in `udCause`) or to `disconnected`.

A management terminal with privilege level `operator` or above may disconnect a **callflow** at its source, by setting `usCause` (if required) and `usState` in the same way as for a destination, provided its privilege level is also at least as high as the current `usPrivilege` value for the entry. This will disconnect all destinations of the **callflow**.

A management terminal with privilege level `supervisor` or above may disconnect a **callflow** at a switch through which it passes, in the same way as for end equipment, provided its privilege level is also at least as high as the current `udPrivilege` or `usPrivilege` value for the entry. Disconnection by setting `udState` disconnects all destinations reached through the switch; disconnection by setting `usState` for a port disconnects all destinations reached through that port.

[EdNote] Updating call identity: procedures for updating call identity will be network-specific, so the most we could say here is that when any of the components of the call identity are updated the procedure defined in the relevant other Sub-part of Part 5 shall be invoked. I haven't mentioned call identity here at all.

6.4 Maintaining calls

On power down or during a reset operation that causes existing calls to be dropped, a unit shall remove all entries from its `unitCallDestList` where the value of `ucRemembered` is `false`.

On power up, if power has been removed for less than **[needs to be configurable]**, a unit shall set the value of `udState` in all remaining entries in its `unitCallDestList` to `readyToConnect`, otherwise it shall remove all remaining entries from its `unitCallDestList`.

If a **callflow** is **droppedterminated** due to normal disconnection, the managed unit shall set its state to `disconnected`. If a **callflow** is **droppedterminated** for any other reason then, if the value of `ucRemembered` is `true`, the unit shall set the value of `udState` in the corresponding entry in its `unitCallDestList` to `readyToConnect`, otherwise it shall set it to `failed`.

After setting the value of `ucState` to `readyToConnect` (for any reason other than a Set request), a unit shall wait for a minimum of $(256 - ucPriority)/10$ seconds, then attempt to ~~establish a new call corresponding to connect the flow, as if~~ the associated entry in its `unitCallDestList`, ~~as if that entry~~ had just been created.

[EdNote 1] Rerouting a call: In the AVA spec, rerouting was done from the source end because of the way ATM does multicasts. Here, maybe it should be done from the destination end, or maybe there should be a duplicate source to which the replacement calls get connected. What is rerouting used for? Is it just to avoid equipment that is to be taken out of service? In that case there ought to be some way the network can orchestrate it. I am not convinced it belongs in here at all.

[EdNote 2] Call replacement: as with updating call identity, I don't think there is anything we can usefully say here.

7 Status broadcasts

7.1 General

[EdNote] This is very different from the format specified in Part 1 and also used in Part 2, which I do not think is right; it differs unnecessarily from the way objects are reported in a GetResponse and doesn't cope with variable-length fields. I don't think it ever got implemented in the Radio 4 project, and I would be surprised if it has been implemented anywhere else. We should maybe put a Note here saying that we aim to migrate to this form. We should also change Part 1 when it comes up for review.

A status broadcast shall consist of a sequence of “reports”. Each report shall show the current value of a MIB object, and shall be classified as either an “in-cycle report” or a “change report” for the object.

Each status broadcast stream shall report a defined set of objects, and shall be divided into “cycles”; each cycle shall contain at least one report for each object in the set as described below.

NOTE 1 The set should not be assumed to be static; for instance, it may consist of certain columns in the list of destinations, in which case objects will be added or removed as records are created or destroyed.

Each object in the set shall be in one of the following states:

- a) pending: value has not been reported in the current cycle
- b) changed: value has changed since it was last reported
- c) reported: value has been reported in the current cycle

An object added to the set is initially in “changed” state. Whenever the value of an object in the set changes, it enters the “changed” state, except that the transition to “changed” state may be delayed until a defined time has elapsed since the previous occasion on which it entered the “changed” state. At the start of a cycle, all objects enter the “pending” state.

Whenever there is an opportunity to transmit a report, if there are any objects in “changed” state a change report is transmitted for one of them, otherwise if there are any objects in “pending” state an in-cycle report is transmitted for one of them, otherwise nothing is transmitted. When a report is transmitted for an object, it enters the “reported” state.

NOTE 2 A change report shows the object's current value; if an object changes several times in quick succession the intermediate values will not necessarily be reported. Implementers should ensure that an object will be in “changed” state if its value is different from that in the most recent report, and also if it has been different at some time since that report. For objects that may change repeatedly, a limit may be placed on the frequency of change reports in order to allow reports of other objects' values to be transmitted. The delay may be specified in relation to other reports, e.g. additional state may be defined such that there cannot be more than one change report for an object between any two adjacent in-cycle reports, except when there are no objects in “pending” state.

A new cycle shall begin when all objects in the set are in “reported” state and a specified time has elapsed since the start of the previous cycle.

NOTE 3 If a cycle takes less than the allotted time, there will be a period during which nothing is transmitted (except change reports, if any changes occur); if it takes more than the allotted time, all subsequent cycles will be delayed.

7.2 Coding and encapsulation of reports

The coding of each report shall consist of: a header octet; the object's identifier (OID); the object's value; and a checksum.

Each report should be transmitted in a separate data unit. If a report does not fit in a single data unit, it shall be divided into fragments, with each fragment other than the first having a header octet (coded for a “continuation fragment”) prepended to it.

NOTE 1 This allows the stream to use small data units, for instance to limit the bandwidth requested if the minimum package rate (see 5.4.3.7 and 5.4.4.11) is 8kHz.

Each header octet shall be encoded as follows. The most significant two bits shall be coded as:

00 in-cycle report, not the last in a cycle

01 last in-cycle report in a cycle

10 change report

11 continuation fragment

and the least significant six bits shall contain a sequence number which shall be one more (modulo 64) than in the previous data unit.

NOTE 2 The sequence number allows the recipient to know whether any data units have been lost. The “length” fields in the BER encoding show whether a continuation is expected. In-cycle reports allow updating of the values of objects for which change reports have been lost.

The checksum shall be two octets containing a nonzero value calculated as follows. If there are an even number of octets in the coding of the report, each pair of octets shall be interpreted as a 16-bit unsigned integer, with the most significant half in the first octet of the pair, and the checksum shall be such that the sum of all the 16-bit integers is a multiple of 65535. If there are an odd number of octets, the checksum shall have the value it would have had if an additional octet containing the value zero had been inserted immediately before the checksum field.

NOTE 3 This calculation is the same as for the checksums used in TCP and UDP.

[EdNote] Well, it's supposed to be; the description is different and I think it is more rigorous.

NOTE 4 The header octet on the first data unit of a fragmented report (including its sequence number) is considered to be part of the report and is included in the checksum; the header octets on the continuation fragments are not. Thus the checksum for each report should be calculated when it is ready to be transmitted, but before fragmentation.

If transmission of each report in a separate data unit would be inefficient, multiple reports may be packed into a single data unit. When this encapsulation is used, each report shall have its own header and checksum.

NOTE 5 It follows from the definition of the sequence number that all reports in a data unit will have the same sequence number.

7.3 Standard report groups

7.3.1 General

The following subclauses list “standard” status reports. Units may also implement other groups of objects to be reported, or allow a caller to specify a bespoke set of objects. When sending reports to a large number of destinations, it is clearly better to be able to multicast a standard report than to have to unicast multiple bespoke reports.

7.3.2 List of sources

The “list of sources” report shall be supported by all units that implement `unitSourceList`. It should consist of `usState` in each record.

NOTE This broadcast simply notifies when new connections appear, and when a connection changes state; a recipient can use a Get request to retrieve other information about the connection. It would not be efficient to include in the broadcast objects that may not have meaningful values. A flow may have many destinations, so reporting the destination would not be viable.

[EdNote] We might consider reporting what the block's input is connected to, but that might not give useful information about the source of the media.

7.3.3 List of destinations

The "list of destinations" report shall be supported by all units that implement `unitDestList`. It should consist of `udState`, `udSourceAddrType`, and `udSourceAddress` in each record.

~~NOTE: Status broadcasts are specified in IEC 62379-1. This standard specifies status pages and groups related to transmission over a network that are not network-specific.~~

~~7.4 Page types~~

~~I imagine we should have a page each for Tables 3-6.~~

~~The AVA spec included a "report type" field coded as: 1 = part of a rolling report, not the last call to be reported on this iteration; 2 = part of a rolling report, last call on this iteration; 3 = call that has just been established, 4 = call that has just been terminated. Hence effectively the record includes a flag saying "change has not yet been reported in status broadcast" which isn't visible in the MIB and prevents the record being deleted.~~

~~For any network that requires the "package rate" to be a multiple of 8kHz, making the "package size" big enough to hold a maximum-length page would be excessive, but if messages are fragmented (as with AAL5 over ATM) it should be OK. The encapsulation of status broadcasts anyway needs to be specified separately for each network technology.~~

~~I do not think the format specified in Part 1 is right; it differs unnecessarily from the way objects are reported in a GetResponse and doesn't cope with variable-length fields.~~

~~I have been using a format in which each object is reported as an (OID, value) pair with periodic report "cycles" that report everything and include sequence numbers and an "end of cycle" flag so the recipient knows whether it has seen it all, and "change" messages which report the new value of objects that have changed as soon as possible after the change happens. I was sending them as large, infrequent "best effort" packets, but it won't require much of a change to adapt the format so the messages are fragmented into smaller, more frequent packets.~~

~~For instance, we can start the packet with one octet containing:~~

~~1 bit: 0 = in-cycle report, 1 = change report~~

~~1 bit: 1 = last packet in cycle or last of the current changes~~

~~1 bit: 0 = packet begins at start of an (OID, value) pair, 1 = continuation~~

~~5 bits sequence number (0 for first in cycle or new set of changes, thereafter cycles round the values 1 to 31)~~

~~The rest of the packet is an (OID, value) pair reporting the current value of an object; if it doesn't fit in one packet then it continues in the next. As we're using small packets, we should start a new packet for each object. Cycle round all the objects that get reported, with any changes interrupting the sequence. The third bit tells anything joining the broadcast when it can start interpreting the data; the ASN.1 length fields tell you when you've got a whole object. Sequence numbers count separately for the two kinds of report.~~

| 7.5 ~~Page Groups~~

|

**Annex A (informative)
Formal definition of MIB**

[To do!]

**Annex B (normative)
Formal Definition of Data Formats**

[To do!]

**~~Annex C~~
Synchronisation**

{Do we want a diatribe on synchronisation issues here?}