Internet Engineering Task Force
INTERNET DRAFT
May 20, 1999
<draft-huitema-megaco-discuss-sdp-01.txt>

Christian Huitema
Telcordia Technologies

Expires: November 20, 1999

Providing H.245 like capability exchanges with Megaco, using SDP

Status of this document

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Abstract:

This memo is a contribution to the ongoing debate in the Megaco working group on the use of SDP or H.245 to describe Termination Descriptors. One of the stingy point in the debate has been the need, for some gateways, to provide H.323 compatible "capability exchanges." It has sometimes been asserted that such exchanges would be hard to provide with an SDP syntax. This memo, on the contrary, provides a simple solution to the problem, based on the following ideas:

* usage of SDP to describe termination parameters,

* usage of the audit procedure to enquire about capabilities,

* a syntax in the audit procedure allowing to return several

termination descriptors, corresponding to alternative capability sets,

* an algorithm for translating a set of SDP encoded capability sets into an H.245 capability descriptor.

1. Introduction:

There is an ongoing debate in the Megaco working group about the choice of a syntax for "termination parameters" and possibly also the "termination capabilities." The tension results from the need to support two types of signalling between distributed media gateways systems composed of controllers (MGC) and gateways (MG), and also between distributed systems and non distributed multi-media systems:

* Systems abiding to ITU specifications follow the H.323 series of recommendations,

* Systems abiding to IETF specifications follow the Session Invitation Protocol (SIP, RFC 2543),

* Some systems supports variations of the ISDN User Part (ISUP) signalling procedures (ITU recommendation Q.761, 762, 763 and 764.)

The choice is essentially between three alternatives:

* Use the syntax specified in SDP (RFC 2327), as used in SIP,

* Use the syntax specified in H.245, as used in H.323,

* Define a new syntax.

Discussions on the working group showed that there was considerable support for SDP, some support for H.245, and a complete unwillingness to define a new syntax:

* Supporters of SDP generally point out that it is a simpler solution (5 pages of ABNF), and thus a better fit for small systems such as "residential gateways". They also assert that the text syntax, combined with IANA registration procedures, provides for easy extensions with complete backward and forward compatibility.

* Supporters of H.245 point out that this is a more complete solution, with a rich set of features such as capability negotiation. However, there are some very vocal opponents to H.245, which is perceived as too complex (more than 50 pages of ASN.1 specification) and hard to extend. Extensions are typically performed as syntax extensions, which, combined with the particularities of the

packed encoding rules, generate incompatibilities between successive versions.

* While theoretically possible, the definition of a completely new syntax is generally ruled out. A new syntax would require its own registration procedure for future extensions, with the risk of neither being compatible with SIP nor with H.323.

This memo explores the possibility of using the extensions procedures that are already defined in the SDP syntax to provide the capability negotiation procedures required for interworking with H.323 systems.

In order to set up the problem, we will first repeat the definition of the cabality exchanges in H.323 (in fact, H.245), and the extensions procedures defined in SDP.

1.1. Parameters of the Capability Exchange messages

In order to compose a Capability Exchange message, the transmitting terminal assigns a number to each individual mode the terminal is capable of operating in. These numbers are the index of the capability in a capabilityTable. For example, G.723.1 audio, G.728 audio, and CIF H.263 video would each be assigned separate numbers.

These capability numbers are grouped into AlternativeCapabilitySet structures. Each AlternativeCapabilitySet indicates that the terminal is capable of operating in exactly one mode listed in the set. For example, an AlternativeCapabilitySet listing {G.711, G.723.1, G.728} means that the terminal can operate in any one of those audio modes, but not more than one.

These AlternativeCapabilitySet structures are grouped into simultaneousCapabilities structures. Each simultaneousCapabilities structure indicates a set of modes the terminal is capable of using simultaneously. For example, a simultaneousCapabilities structure containing the two AlternativeCapabilitySet structures {H.261, H.263} and {G.711, G.723.1, G.728} means that the terminal can operate either of the video codecs simultaneously with any one of the audio codecs. The simultaneousCapabilities set {{H.261}, {H.261, H.263}, {G.711, G.723.1, G.728} } means the terminal can operate two video channels and one audio channel simultaneously: one video channel per H.261, another video channel per either H.261 or H.263, and one audio channel per either G.711, G.723.1, or G.728.

The capabilities are defined by selecting first a capability type, such as VideoCapability, AudioCapability, DataApplicationCapability or ConferenceCapability. H.245 does not provide identifiers for capability types, but simply define them as alternatives in the ASN.1 "Capability"

type of the syntax. Each specific capability, in turn, is defined by its own ASN.1 type, such as for example:

```
VideoCapability ::=CHOICE
{
    nonStandard NonStandardParameter ,
    h261VideoCapability H261VideoCapability,
    h262VideoCapability H262VideoCapability,
    h263VideoCapability H263VideoCapability,
    is11172VideoCapability IS11172VideoCapability,
    ...
}
```

The most used types, such as for example "H261VideoCapability" are explicitly defined by their own ASN.1 data type, such as:

```
H261VideoCapability ::=SEQUENCE
{
    qcifMPI INTEGER (1..4) OPTIONAL, -- units 1/29.97 Hz
    cifMPI  INTEGER (1..4) OPTIONAL, -- units 1/29.97 Hz
    temporalSpatialTradeOffCapability BOOLEAN,
    maxBitRate INTEGER (1..19200),   -- units of 100 bit/s
    stillImageTransmission BOOLEAN,  -- annex D of H.261
    ...
}
```

Less used types and extensions can used the NonStandardParameter structure, which is composed of a parameter identifier (NonStandardIdentifier, generally an ASN.1 Object Identifier) and of an unstructured object string.

1.2. SDP extension procedures

SDP has been designed to be extensible in three ways:

* New media types can be registered to the IANA, using the procedure defined for MIME media types.

* New encoding types can be registered to the IANA,

* New session and media attributes can be defined.

The extensions procedures have two effects. They enable the systems to evolve with the advent of new technology, but they also enable interoperability between old and new systems. For example, one could define a new media type for, for example, a smell carrying channel, and use this

definition to invite a partner to a truly ecstatic multimedia session, as in the following description:

```
v=0
o=chuitema 2890844526 2890842807 IN IP4 192.96.41.1
c=IN IP4 224.2.17.12/127
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=smell 32416 udp wperfume
```

If the invited partner is equipped to handle the new media, it will provide its own address and port in the response. If it is not equipped to do so, it will be able to parse the message and to understand that the "m=smell" line refers to an unsupported media. It can still set up the session, using only the audio and video media.

The proposed encoding types, when using SDP, are defined by the RTP payload type. The values "0" and "31" in the previous example correspond to payload type reserved in the standard Audio Video profile (RFC 1890, whose revision is in progress). The value 0 is assigned to "PCMU" (G.711 with mu law encodings) and the value 31 is assigned to H.261. New algorithms can be defined using the "negotiated payload type" facility. For example, if Telcordia Technologies defined a new audio coding scheme for HiFi audio and reserved the corresponding name "TThifi", one could issue an invite message that carried a list of proposed codecs such as:

```
v=0
o=chuitema 2890844526 2890842807 IN IP4 192.96.41.1
c=IN IP4 224.2.17.12/127
m=audio 49170 RTP/AVP 98 0
a=rtpmap:98 TThifi/44100/16
```

The "rtpmap" attribute, in this example, provides a complete definition of the non standard payload type "98." The general form of an rtpmap attribute is:

```
a=rtpmap:<payload type> <encoding name>/<clock rate>
                                        [/<encoding parameters>]
```

The encoding parameters are specific to each media.

If the invited partner is equipped to handle the new encoding, it can use it in RTP packets. If it is not equipped to do so, it will be able to parse the message and to choose the next alternative, in our case PCMU.

According to the SDP specification, "attributes are the primary means for extending SDP." "Attributes that will be commonly used can be registered with IANA(see Appendix B). Unregistered attributes should begin with "X-" to prevent inadvertent collision with registered attributes. In either case, if an attribute is received that is not understood, it should simply be ignored by the receiver."

2. Providing capability negotiation in Megaco

The first difference between H.245 and SDP is the need to support a "capability exchange" procedure. The exchange procedure itself cannot be supported in SDP, which only defines format. We propose to support it through a combination of SDP and the Megaco audit procedure. We may expect that capacities will be mostly static, and that the MGC will in practice only need to query the MG at infrequent intervals. However, we must define a general procedure that can be used, if needed, on a call by call basis.

2.1. Mapping of the Capability Exchange parameters using SDP

In order to map the Capability Exchange parameters using SDP, we need to provide mappings for:

* Individual capacities,

* Simultaneous Capabilities,

* Alternative Capability Set .

We will detail the proposed mappings in the following subsections.

2.1.1. Mapping of individual capacities into SDP

The definition of audio, video or data capacities in H.245 varies from the very simple to the fairly complex. Most "audio capacities" are simply defined as an algorithm type and the number of audio frames that can be supported per packet. Some require more sophistication, such as the specification of the mode values for G.723 annex C. Video capacities, on the other hand, are defined by complex set of parameters corresponding to the various encoding options. The capability to support a given algorithm is expressed in SDP by:

* Placing a reference to the standard payload type associated with the algorithm in the corresponding "media" line,

* Or by placing a reference to a dynamic payload type in a media line, and by defining the corresponding algorithm type and

algorithm parameters in the "rtpmap" attribute.

In addition to the type, the RTPMAP attribute systematically defines the clock rate used for the media, and may also include algorithm specific attribute. It does not generally include a count of frame per packet, because stations are in most cases expected to use the default value defined for the media. Non default value could be provided by using the "ptime" attribute.

The following table provide a list of audio and video formats supported in H.323, with the RTP/AVP equivalence when it is defined:

H.245 capa.	AVP #	Description
g711Alaw64k, g711Alaw56k	PCMA 8	No difference in RTP between 56 and 64 kbps.
g711Ulaw64k, g711Ulaw56k	PCMU 0	No difference in RTP between 56 and 64 kbps.
g722-64k, g722-56k, g722-48k	G722 9	The RTP/AVP code point only refers to the 64 kbps encoding.
g7231, g7231AnnexC-Capability	G723 4	The RTP/AVP code point is common to all variants of G.723. Support of low bit rate, high bit rate and comfort noise is mandatory in receivers. The RTP/AVP profiles don't include a specific code point.
g728	G728 15	
g729, g729AnnexA, g729wAnnexB, g729Annex-AwAnnexB	G729 18	According to RTP/AVP, a terminal that supports G729 shall be capable of receiving packets composed of AnnexA and AnnexB frames. The payload name G729B is reserved for flows that would only include comfort noise.
is11172Audio-Capability, is13818Audio-Capability	MPA 14	MPEG audio encoding in RTP is defined in RFC 2038, "RTP payload format for MPEG1/MPEG2 video," which also defines video encodings (MPV).
gsmFullRate, gsmHalfRate, gsmEnhanced-FullRate	GSM 3	Current RTP/AVP specification only refers to the GSM "full rate" encoding (13 kbps).

H.245 capability	AVP	#	Description
h261VideoCapability	H261	31	
h262VideoCapability			
h263VideoCapability	H263	34	
is11172VideoCapability	MPV	32	MPEG Video, as defined in RFC 2038.

We should note that the RTP/AVP specification defines types that have no equivalent in H.245, such as:

AVP	#	Description
1016	1	
G726-32	2	ADPCM, G.726
DVI4	5	Intel's DVI format, 4 bits adaptative, 8KHz
VDVI		Variable length DVI
DVI4	6	16 Khz
DVI4	16	11.025 KHz
DVI4	17	22.050 KHz
LPC	7	Linear Prediction Coding.
L16	10	Linear 16 bits 44.1 KHz, stereo
L16	11	Linear 16 bits 44.1 KHz, mono
CN	19	Comfort noise, 8 KHz.
CelB	25	Cell-B format, defined by SUN
JPEG	26	JPEG video.
nv	28	Network Video format, defined by Xerox/PARC.
MP2T	33	MPEG stream, mixing audio and video, as defined in RFC 2038.
RED	77	Redundancy audio encodings

These capacities will have to be translated into "non standard" audio or video H.245 capabilities, using the NonStandardParameter data type, which is generally composed of an identifier and an octet string. A generic support could be achieved by:

* Providing an algorithmic procedure to map an IANA registered name, such as "nv", into a NonStandardParameter identifier. Currently, the identifiers can only be defined as "Object Identifiers." Interworking would be greatly eased if a alternate "textual identifier" was allowed.

* Providing an algorithmic procedure to map the SDP description into the NonStandardParameter octet string, possibly by mapping the text of the RTPMAP attribute into the octet string.

Data capacities are generally not expressed using the RTP payload format. In SDP, a data stream will be defined by its own media line, as in for example:

```
m=application 32416 udp wb
a=orient:portrait
```

This means that for each data medium that is desired, the SDP description will incorporate a line describing the application, plus a set of attributes. H.245 defines a set of applications, such as:

```
t120
dsm-cc
userData
t84
t434
h224
nlpid
dsvdControl
h222DataPartitioning
t30fax
vChat
```

The data capability information includes the name of the application that can be run and parameters that describe the application profile. These parameters are generally composed of a "DataProtocolCapability" element and of a bandwidth notation. In complement, a few applications are further qualified by application specific data types, such as for example a T84 Profile. To map such capabilities into SDP, one will have to design a generic DataProtocolCapability attribute and set of adequate attributes, for example " a "T84Profile" attribute. With these definition, the mapping of a T84 data capability, for example, could be expressed as:

```
m=application 32416 tcp t84
b=AS:128
a=DataProtocolCapability:...
a=T84Profile:...
```

2.1.2. Using SDP to describe capability descriptors and simultaneous capacities

The SDP structures, by nature, provides a list of supported media and a list of algorithms or applications that can be supported for each media. We have seen in the previous subsection how each media specification could be translated into a set of capabilities. This suggest an algorithm for deriving capability sets:

* Initialize an H.245 Capability Descriptor structure and an H.245 Capability Table.

* Examine each media line in the SDP description. Initialize an Alternative Capability Set element in the simultaneous capability list corresponding to that element.

* From each media line, examine each supported encoding, and note the encoding specific parameters (a=rtpmap) as well as the media level parameters (b=, a=ptime, etc.)

* Using these parameters, construct an individual capability for each supported algorithm, and add that capability to the "capability table."

* Add the capability number to the Alternative Capability Set corresponding to the element.

This algorithm transforms the list of media descriptions provided in SDP into exactly one H.245 Capability Descriptor.

2.1.3. Using SDP to describe the H.245 Terminal Capability Set

The preceding algorithm allows us to initialize an H.245 Terminal Capability Set element from the content of an SDP specification. Within the set:

* The sequenceNumber is initialized by the MGC, as part of the H.245 protocol support,

* The protocolIdentifier is set to the recommended value,

* The multiplexCapability is not set,

* The capabilityTable reflects the list of algorithms specified in the various media descriptions,

* The capabilityDescriptors contains exactly one Capability Descriptor, which in turn contains one Alternative Capability Set per

media description.

In order to support multiple sets, we need to:

* either increment the SDP syntax to allow for the notation of alternatives,

* or make sure that the Megaco protocol can carry several alternatives.

The first option does not present any particular difficulty, but would make the SDP version used in Megaco incompatible with the existing implementations of SDP. The second option is very easy to implement if we are only concerned with the support of the capability negotiation facility. Basically, we only need to slightly upgrade the algorithm presented in the previous subsection, to:

* Initialize a Terminal Capability Set with an empty Capability Table and an empty Capability Descriptor set.

* For each SDP specification, initialize a Capability Descriptor structure

* Examine each media line in the SDP description. Initialize an Alternative Capability Set element in the simultaneous capability list corresponding to that element.

* From each media line, examine each supported encoding, and note the encoding specific parameters (a=rtpmap) as well as the media level parameters (b=, a=ptime, etc.)

* Using these parameters, construct an individual capability for each supported algorithm and

* Add that capability to the "capability table."

* Add the capability number to the Alternative Capability Set corresponding to the element.

* Add the Alternative Capability Set to the Capability Descriptor.

* add the Capability Descriptor to the Capability Descriptor Set.

The algorithm has two advantages:

* it provides for automated mappings,

* it fully supports the semantics of the H.245 Terminal Capability Set notation.

2.2. Usage of the audit procedure for capability negotiation

The audit procedure is designed to let the MGC obtain information on the capability of an MG, or on the capability of a termination. The general structure of the protocol should allow the following exchange:

* The MGC selects a termination and AUDIT that termination,

* The MG respond with a list of capabilities,

* The MGC translates the list of capabilities into a H.245 Capability Exchange message.

The parameters of an MG, or a termination, are not entirely covered by definition of algorithms and supported media types. The protocol definition separates six groups of parameters:

* The TerminationState, which describes local parameters such as the termination mode,

* The LocalTerminationDescriptor, which describes what type of media and what algorithm can be received on the termination,

* The RemoteTerminationDescriptor, which describes what type of media and what algorithm can be sent from the termination,

* The EventsDescriptor, which describes the events that can be observed,

* The SignalDescriptor, which describes the signals that can be applied,

* The DigitMapDescriptor, which describes the digit maps.

The AUDIT command can be used to list the current values of an endpoint parameter. In order to be useful for negotiations, we must also be able to list the "possible" values of these parameters. Specifically, in order to be compatible with the capability exchange, we need to be able to list the possible values of the "LocalTerminationDescriptor." In order to enable negotiation by the MGC on behalf of the MG, we may also need to audit the acceptable values of the RemoteTerminationDescriptor.

We propose here to introduce these functions in the Megaco protocol by redefining the syntax of the RequestedInfo and Capacities parameters.

The current definition of the RequestedInfo parameter is:

```
RequestedInfo = [infoCode 0*(infoCode)]
infoCode = TerminationStateToken / LocalTermDescToken /
           RemoteTermDescToken / EventDeccToken /
           SignalDescToken / DigMapToken / StatsToken /
           ObsrvdEvntsToken / CapsToken
```

We propose to extend this to the following value:

```
RequestedInfo = [infoCode 0*(infoCode)]
infoCode = actualInfoCode / CapsToken capabilityRequest
actualInfoCode =
           TerminationStateToken / LocalTermDescToken /
           RemoteTermDescToken / EventDeccToken /
           SignalDescToken / DigMapToken / StatsToken /
           ObsrvdEvntsToken
capabilityRequest = 1*actualInfoCode
```

Using this definition, it becomes possible to audit the possible values of the LocalTerminationDescriptor by issuing an audit command whose requested info is set to "capabilityRequest (LocalTermDescToken)."

The Capability parameter is not defined in the current syntax. A logical description would be to consider it as a set of valid encoding of the requested parameters, as in:

```
Capabilities =
      (*(TSToken TerminationState)
       *(LTToken LocalTerminationDescriptor )
       *(RTToken RemoteTerminationDescriptor )
       *(EDToken EventsDescriptor )
       *(SDToken SignalDescriptor )
       *(DMToken DigitMapDescriptor )
       *(RIToken RequestedInfo )
       *(RMToken ServiceChangeMethod )
       *(RDToken ServiceChangeDelay )
       *(OEToken ObservedEvents )
       *(STToken Statistics )
       *(MiToken MGCIdentity )
       *(extensionParameterToken parameterValue))
```

This proposed syntax allows for multiple instantiations of each parameter. For example, a request for the "Local Termination Descriptor" token would result in several values of the Local Termination Descriptor, each encoded as in SDP. Each value could describe a set of

alternative capabilities.

3. Extensions to SDP

The work that we did here assumes that SDP can encode the requested algorithms. This will require a coordinated set of extensions, to define parameters such as echo control, gain control, or algorithm specific parameters. This work should be organized by coordination between MMUSIC and MEGACO.