MMUSIC Working Group
Internet-Draft
Intended Status: Proposed Standard
Expires: April 2008

F. Andreasen
Cisco Systems

October 28, 2007

SDP Capability Negotiation

draft-ietf-mmusic-sdp-capability-negotiation-07.txt

Status of this Memo

Copyright Notice

Abstract

The Session Description Protocol (SDP) was intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. SDP was not intended to provide capability indication or capability negotiation, however over the years, SDP has seen widespread adoption and as a result it has been gradually extended to provide limited support for these, notably in the form of the

offer/answer model defined in RFC 3264. SDP and its current extensions do not define how to negotiate one or more alternative transport protocols (e.g. RTP profiles) or attributes. This makes it difficult to deploy new RTP profiles such as secure RTP or RTP with RTCP-based feedback, negotiate use of different security keying mechanisms, etc. It also presents problems for some forms of media negotiation.

The purpose of this document is to address these shortcomings by extending SDP with capability negotiation parameters and associated offer/answer procedures to use those parameters in a backwards compatible manner.

The document defines a general SDP Capability Negotiation framework. It also specifies how to provide attributes and transport protocols as capabilities and negotiate them using the framework. Extensions for other types of capabilities (e.g. media types and media formats) may be provided in other documents.

Table of Contents

1. Introduction

The Session Description Protocol (SDP) was intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. The SDP contains one or more media stream descriptions with information such as IP-address and port, type of media stream (e.g. audio or video), transport protocol (possibly including

profile information, e.g. RTP/AVP or RTP/SAVP), media formats (e.g. codecs), and various other session and media stream parameters that define the session.

Simply providing media stream descriptions is sufficient for session announcements for a broadcast application, where the media stream parameters are fixed for all participants. When a participant wants to join the session, he obtains the session announcement and uses the media descriptions provided, e.g., joins a multicast group and receives media packets in the encoding format specified. If the media stream description is not supported by the participant, he is unable to receive the media.

Such restrictions are not generally acceptable to multimedia session invitations, where two or more entities attempt to establish a media session, that uses a set of media stream parameters acceptable to all participants. First of all, each entity must inform the other of its receive address, and secondly, the entities need to agree on the media stream parameters to use for the session, e.g. transport protocols and codecs. To solve this, RFC 3264 [RFC3264] defined the offer/answer model, whereby an offerer constructs an offer SDP that lists the media streams, codecs, and other SDP parameters that the offerer is willing to use. This offer SDP is sent to the answerer, which chooses from among the media streams, codecs and other SDP parameters provided, and generates an answer SDP with his parameters, based on that choice. The answer SDP is sent back to the offerer thereby completing the session negotiation and enabling the establishment of the negotiated media streams.

Taking a step back, we can make a distinction between the capabilities supported by each participant, the way in which those capabilities can be supported, and the parameters that can actually be used for the session. More generally, we can say that we have the following:

- A set of capabilities for the session and its associated media stream components, supported by each side. The capability indications by themselves do not imply a commitment to use the capabilities in the session.

  Capabilities can for example be that the "RTP/SAVP" profile is supported, that the "PCMU" codec is supported, or that the "crypto" attribute is supported with a particular value.

- A set of potential configurations indicating which combinations of those capabilities can be used for the session and its associated media stream components. Potential configurations are not ready for use. Instead, they provide an alternative that may be used, subject to further negotiation.

  A potential configuration can for example indicate that the "PCMU" codec and the "RTP/SAVP" transport protocol are not only supported (i.e. listed as capabilities), but they are offered for potential use in the session.

- An actual configuration for the session and its associated media stream components, that specifies which combinations of session parameters and media stream components can be used currently and with what parameters. Use of an actual configuration does not require any further negotiation.

  An actual configuration can for example be that the "PCMU" codec and the "RTP/SAVP" transport protocol are offered for use currently.

- A negotiation process that takes the set of actual and potential configurations (combinations of capabilities) as input and provides the negotiated actual configurations as output.

SDP by itself was designed to provide only one of these, namely listing of the actual configurations, however over the years, use of SDP has been extended beyond its original scope. Of particular importance are the session negotiation semantics that were defined by the offer/answer model in RFC 3264. In this model, both the offer and the answer contain actual configurations; separate capabilities and potential configurations are not supported.

Other relevant extensions have been defined as well. RFC 3407 [RFC3407] defined simple capability declarations, which extends SDP with a simple and limited set of capability descriptions. Grouping of media lines, which defines how media lines in SDP can have other semantics than the traditional "simultaneous media streams" semantics, was defined in RFC 3388 [RFC3388], etc.

Each of these extensions was designed to solve a specific limitation of SDP. Since SDP had already been stretched beyond its original intent, a more comprehensive capability declaration and negotiation process was intentionally not defined. Instead, work on a "next generation" of a protocol to provide session description and capability negotiation was initiated [SDPng]. SDPng defined a

comprehensive capability negotiation framework and protocol that was not bound by existing SDP constraints. SDPng was not designed to be backwards compatible with existing SDP and hence required both sides to support it, with a graceful fallback to legacy operation when needed. This, combined with lack of ubiquitous multipart MIME support in the protocols that would carry SDP or SDPng, made it challenging to migrate towards SDPng. In practice, SDPng has not gained traction but rather remained as work in progress for an extended period of time. Existing real-time multimedia communication protocols such as SIP, RTSP, Megaco, and MGCP continue to use SDP. However, SDP and its current extensions do not address an increasingly important problem: the ability to negotiate one or more alternative transport protocols (e.g., RTP profiles) and associated parameters (e.g. SDP attributes). This makes it difficult to deploy new RTP profiles such as secure RTP (SRTP) [RFC3711], RTP with RTCP-Based Feedback [RFC4585], etc. The problem is exacerbated by the fact that RTP profiles are defined independently. When a new profile is defined and N other profiles already exist, there is a potential need for defining N additional profiles, since profiles cannot be combined automatically. For example, in order to support the plain and secure RTP version of RTP with and without RTCP-based feedback, four separate profiles (and hence profile definitions) are needed: RTP/AVP [RFC3551], RTP/SAVP [RFC3711], RTP/AVPF [RFC4585], and RTP/SAVPF [SAVPF]. In addition to the pressing profile negotiation problem, other important real-life limitations have been found as well. Keying material and other parameters for example need to be negotiated with some of the transport protocols, but not others. Similarly, some media formats and types of media streams need to negotiate a variety of different parameters.

The purpose of this document is to define a mechanism that enables SDP to provide limited support for indicating capabilities and their associated potential configurations, and negotiate the use of those potential configurations as actual configurations. It is not the intent to provide a full-fledged capability indication and negotiation mechanism along the lines of SDPng or ITU-T H.245. Instead, the focus is on addressing a set of well-known real-life limitations. More specifically, the solution provided in this document provides a general SDP Capability Negotiation framework that is backwards compatible with existing SDP. It also defines specifically how to provide attributes and transport protocols as capabilities and negotiate them using the framework. Extensions for other types of capabilities (e.g. media types and formats) may be provided in other documents.

As mentioned above, SDP is used by several protocols, and hence the mechanism should be usable by all of these. One particularly important protocol for this problem is the Session Initiation Protocol (SIP) [RFC3261]. SIP uses the offer/answer model [RFC3264] (which is not specific to SIP) to negotiate sessions and hence the mechanism defined here provides the offer/answer procedures to use for the capability negotiation framework.

The rest of the document is structured as follows. In Section 3. we present the SDP Capability Negotiation solution, which consists of new SDP attributes and associated offer/answer procedures. In Section 4. we provide examples illustrating its use and in Section 5. we provide the security considerations.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. SDP Capability Negotiation Solution

In this section we first present the conceptual model behind the SDP capability negotiation framework followed by an overview of the SDP Capability Negotiation solution. We then define new SDP attributes for the solution and provide its associated updated offer/answer procedures.

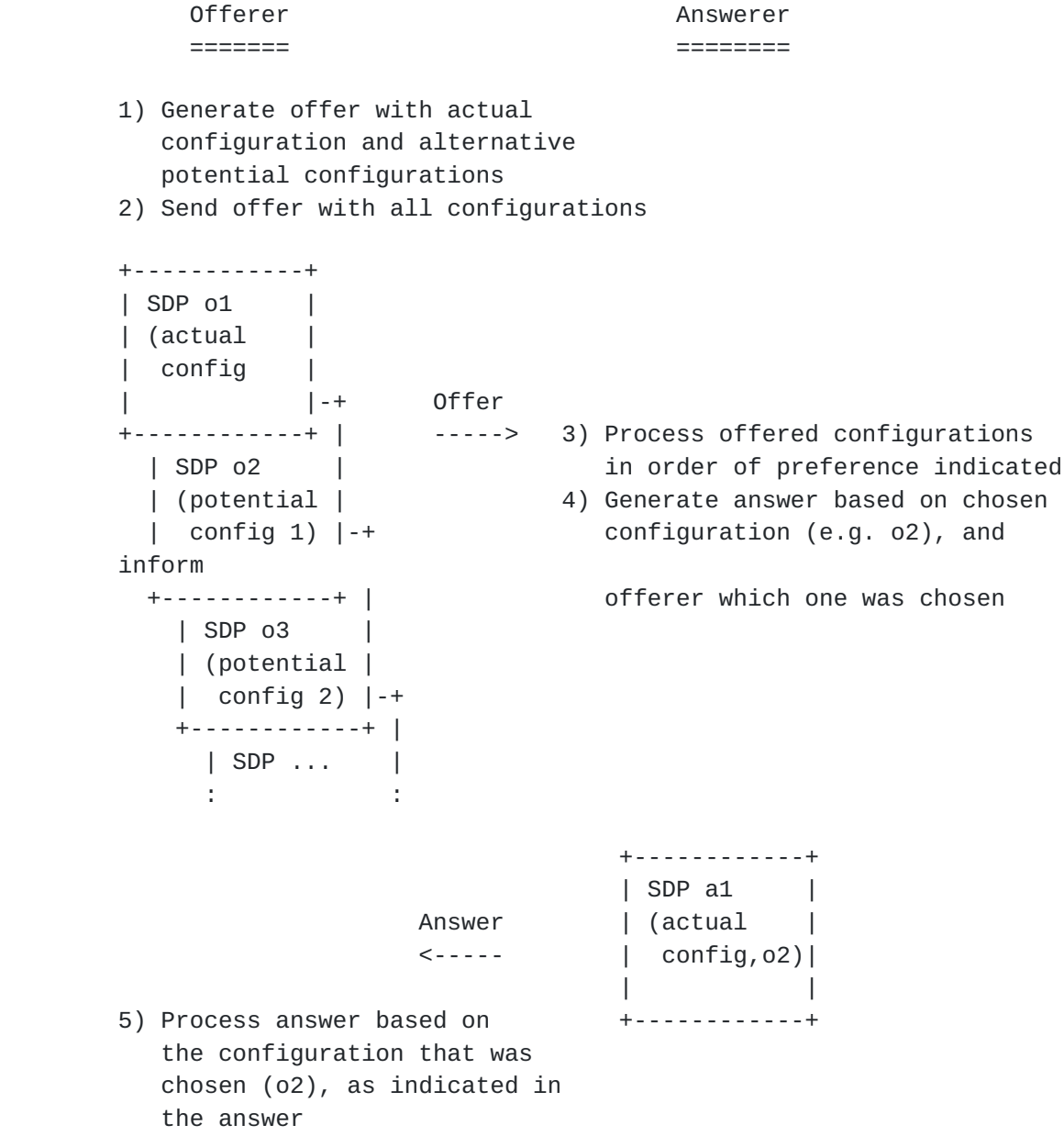
3.1. SDP Capability Negotiation Model

Our model uses the concepts of

- Capabilities
- Potential Configurations
- Actual Configurations
- Negotiation Process

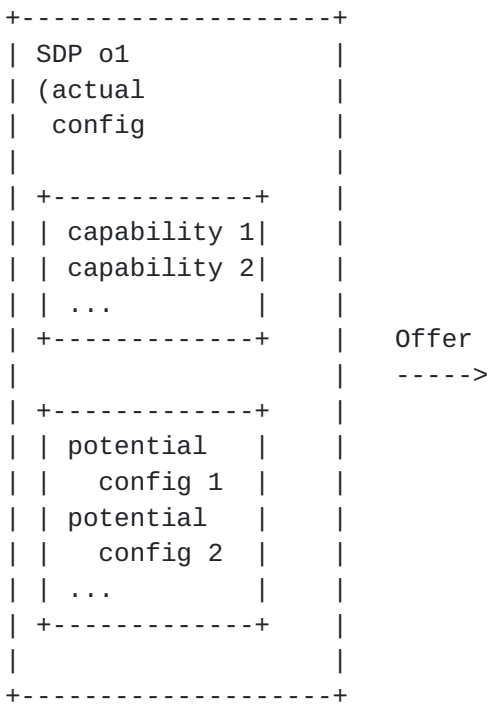
as defined in Section 1. Conceptually, we want to offer not just the actual configuration SDP (which is done with the current offer/answer model), but the actual configuration SDP as well as one or more alternative SDPs, i.e. potential configurations. The answerer must choose either the actual configuration, or one of the potential configurations, and generate an answer SDP based on that.

The offerer may need to perform processing on the answer, which depends on the offer that was chosen (actual or potential configuration). The answerer therefore informs the offerer which configuration the answerer chose. The process can be viewed *conceptually* as follows:



The above illustrates the conceptual model: The actual solution uses a single SDP, which contains the actual configuration (as with

current SDP and the current offer/answer model) and several new attributes and associated procedures, that encode the capabilities and potential configurations. A more accurate depiction of the actual offer SDP is therefore as follows:



The above structure is used for two reasons:

- Backwards compatibility: As noted above, support for multipart MIME is not ubiquitous. By encoding both capabilities and potential configurations in SDP attributes, we can represent everything in a single SDP thereby avoiding any multipart MIME support issues. Furthermore, since unknown SDP attributes are ignored by the SDP recipient, we ensure that entities that do not support the framework simply perform the regular RFC 3264 offer/answer procedures. This provides us with seamless backwards compatibility.

- Message size efficiency: When we have multiple media streams, each of which may potentially use two or more different transport protocols with a variety of different associated parameters, the number of potential configurations can be large. If each possible alternative is represented as a complete SDP in an offer, we can easily end up with large messages. By providing a more compact encoding, we get more efficient message sizes.

In the next section, we describe the exact structure and specific SDP parameters used to represent this.

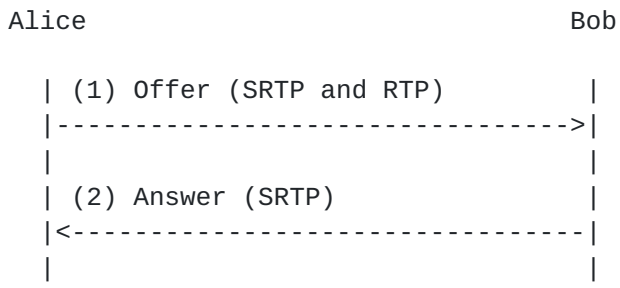
3.2. Solution Overview

The solution consists of the following:

- Two new attributes to support extensions to the framework itself as follows:
  - A new attribute ("a=csup") that lists the supported base (optionally) and any supported extension options to the framework.
  - A new attribute ("a=creq") that lists the extensions to the framework that are required to be supported by the entity receiving the SDP in order to do capability negotiation.
- Two new attributes used to express capabilities as follows (additional attributes can be defined as extensions):
  - A new attribute ("a=acap") that defines how to list an attribute name and its associated value (if any) as a capability.
  - A new attribute ("a=tcap") that defines how to list transport protocols (e.g. "RTP/AVP") as capabilities.
- Two new attributes to negotiate configurations as follows:
  - A new attribute ("a=pcfg") that lists potential configurations supported. This is done by reference to the capabilities from the SDP in question. Extension capabilities can be defined and referenced in the potential configurations. Alternative potential configurations have an explicit ordering associated with them.
  - A new attribute ("a=acfg") to be used in an answer SDP. The attribute identifies a potential configuration from an offer SDP which was used as an actual configuration to form the answer SDP. Extension capabilities can be included as well.

o Extensions to the offer/answer model that allow for capabilities and potential configurations to be included in an offer. Capabilities can be provided at the session level and the media level. Potential configurations can be included at the media level only, where they constitute alternative offers that may be accepted by the answerer instead of the actual configuration(s) included in the "m=" line(s) and associated parameters. The answerer indicates which (if any) of the potential configurations it used to form the answer by including the actual configuration attribute ("a=acfg") in the answer. Capabilities may be included in answers as well, where they can aid in guiding a subsequent new offer.

The mechanism is illustrated by the offer/answer exchange below, where Alice sends an offer to Bob:



Alice's offer includes RTP and SRTP as alternatives. RTP is the default (actual configuration), but SRTP is the preferred one (potential configuration):

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVP 0 18
a=tcap:1 RTP/SAVP
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=pcfg:1 t=1 a=1
```

The "m=" line indicates that Alice is offering to use plain RTP with PCMU or G.729. The capabilities are provided by the "a=tcap" and "a=acap" attributes. The transport capabilities ("a=tcap") indicate that secure RTP under the AVP profile ("RTP/SAVP") is supported with an associated transport capability handle of 1. The "acap" attribute provides an attribute capability with a handle of 1. The attribute

capability is a "crypto" attribute, which provides the keying material for SRTP using SDP security descriptions [RFC4568]. The "a=pcfg" attribute provides the potential configuration included in the offer by reference to the capability parameters. One alternative is provided; it has a configuration number of 1 and it consists of transport protocol capability 1 (i.e. the RTP/SAVP profile - secure RTP), and the attribute capability 1, i.e. the crypto attribute provided. Potential configurations are preferred over the actual configuration included in the offer SDP, and hence Alice is expressing a preference for using secure RTP.

Bob receives the SDP offer from Alice. Bob supports SRTP and the SDP Capability Negotiation framework, and hence he accepts the (preferred) potential configuration for Secure RTP provided by Alice and generates the following answer SDP:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/SAVP 0 18
a=crypto:1 AES_CM_128_HMAC_SHA1_80
      inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
a=acfg:1 t=1 a=1
```

Bob includes the "a=acfg" attribute in the answer to inform Alice that he based his answer on an offer containing the potential configuration with transport protocol capability 1 and attribute capability 1 from the offer SDP (i.e. the RTP/SAVP profile using the keying material provided). Bob also includes his keying material in a "crypto" attribute. If Bob supported one or more extensions to the capability negotiation framework, he would have included option tags for those in the answer as well (in an "a=csup" attribute).

Note that in this particular example, the answerer supported the capability negotiation extensions defined here. Had he not, he would simply have ignored the new attributes and accepted the (actual configuration) offer to use normal RTP. In that case, the following answer would have been generated instead:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
```

```
t=0 0
m=audio 54568 RTP/AVP 0 18
```

3.3. Version and Extension Indication Attributes

In this section, we present the new attributes associated with indicating the SDP Capability Negotiation extensions supported and required.

3.3.1. Supported Capability Negotiation Extensions Attribute

The SDP Capability Negotiation solution allows for capability negotiation extensions to be defined. Associated with each such extension is an option tag that identifies the extension in question. Option-tags MUST be registered with IANA per the procedures defined in Section 6.

The Supported Capability Negotiation Extensions attribute ("a=csup") contains a comma-separated list of option tags identifying the SDP Capability Negotiation extensions supported by the entity that generated the SDP. The attribute is defined as follows:

```
a=csup: <option-tag-list>
```

RFC 4566, Section 9, provides the ABNF [RFC4234] for SDP attributes. The "csup" attribute adheres to the RFC 4566 "attribute" production, with an att-value defined as follows:

```
att-value         = option-tag-list
option-tag-list   = option-tag *("," option-tag)
option-tag        = token    ; defined in [RFC4566]
```

A special base option tag with a value of "cap-v0" is defined for the basic SDP Capability Negotiation framework defined in this document. Entities can use this option tag with the "a=csup" attribute to indicate support for the SDP Capability Negotiation framework specified in this document.

The following examples illustrate use of the "a=csup" attribute with the "cap-v0" option tag and two hypothetical option tags, "foo" and "bar" (note the lack of white space):

```
a=csup:cap-v0

a=csup:foo
```

```
a=csup:bar

a=csup:cap-v0,foo,bar
```

The "a=csup" attribute can be provided at the session and the media-level. When provided at the session-level, it applies to the entire SDP. When provided at the media-level, it applies to the media description in question only (option-tags provided at the session level apply as well). There can be at most one "a=csup" attribute at the session-level and at most one at the media-level (one per media description in the latter case).

Whenever an entity that supports one or more extensions to the SDP Capability Negotiation framework generates an SDP, it SHOULD include the "a=csup" attribute with the option tags for the extensions it supports at the session and/or media-level, unless those option tags are already provided in one or more "a=creq" attribute (see Section 3.3.2.) at the relevant levels. Inclusion of the base option tag is OPTIONAL; support for the base framework can be inferred from presence of the "a=pcfg" attribute defined in Section 3.5.1.

> Use of the base option tag may still be useful in some scenarios, e.g. when using SIP OPTIONS [RFC3261] or generating an answer to an offer that did not use the SDP Capability Negotiation framework.

3.3.2. Required Capability Negotiation Extensions Attribute

The Required Capability Negotiation Extensions attribute ("a=creq") contains a comma-separated list of option tags (see Section 3.3.1.) specifying the SDP Capability Negotiation extensions that MUST be supported by the entity receiving the SDP, in order for that entity to properly process the SDP Capability Negotiation attributes and associated procedures. Support for the basic negotiation framework is implied by the presence of an "a=pcfg" attribute (see Section 3.5.1.) and hence there is no need to include the "a=creq" attribute with the base option-tag ("cap-v0"). Still, it is allowed to do so.

The attribute is defined as follows:

```
a=creq: <option-tag-list>
```

The "creq" attribute adheres to the RFC 4566 "attribute" production, with an att-value defined as follows:

```
att-value   = option-tag-list
```

The following examples illustrate use of the "a=creq" attribute with the "cap-v0" base option tag and two hypothetical option tags, "foo" and "bar" (note the lack of white space):

```
a=creq:cap-v0

a=creq:foo

a=creq:bar

a=creq:cap-v0,foo,bar
```

The "a=creq" attribute can be provided at the session and the media-level. When provided at the session-level, it applies to the entire SDP. When provided at the media-level, it applies to the media description in question only (required option tags provided at the session level apply as well). There can be at most one "a=creq" attribute at the session-level and at most one "a=creq" attribute at the media-level (one per media description in the latter case).

When an entity generates an SDP and it requires the recipient of that SDP to support one or more SDP Capability Negotiation extensions (except for the base) at the session or media level in order to properly process the SDP Capability Negotiation, the "a=creq" attribute MUST be included with option-tags that identify the required extensions at the session and/or media level. If support for an extension is needed only in one or more specific potential configurations, the potential configuration provides a way to indicate that instead (see Section 3.5.1.). Support for the basic negotiation framework is implied by the presence of an "a=pcfg" attribute (see Section 3.5.1.) and hence it is not required to include the "a=creq" attribute with the base option-tag ("cap-v0").

A recipient that receives an SDP and does not support one or more of the required extensions listed in a "creq" attribute, MUST NOT perform the SDP Capability Negotiation defined in this document. For non-supported extensions provided at the session-level, this implies that SDP Capability Negotiation MUST NOT be performed at all. For non-supported extensions at the media-level, this implies that SDP Capability Negotiation MUST NOT be performed for the media stream in question.

An entity that does not support the SDP Capability Negotiation framework at all, will ignore these attributes (as well as the other SDP Capability Negotiation attributes) and not perform any SDP Capability Negotiation in the first place.

When an entity does not support one or more required SDP Capability Negotiation extensions listed in the option tags, the entity MUST proceed as if the SDP Capability Negotiation attributes were not included in the first place, i.e. all the capability negotiation attributes should be ignored. In that case, the entity SHOULD include a "csup" attribute listing the SDP Capability Negotiation extensions it actually supports.

This ensures that introduction of the SDP Capability Negotiation mechanism by itself does not lead to session failures.

3.4. Capability Attributes

In this section, we present the new attributes associated with indicating the capabilities for use by the SDP Capability Negotiation.

3.4.1. Attribute Capability Attribute

Attributes and their associated values can be expressed as capabilities by use of a new attribute capability attribute ("a=acap"), which is defined as follows:

```
a=acap: <att-cap-num> <att-par>
```

where <att-cap-num> is an integer between 1 and 2^31-1 (both included) used to number the attribute capability and <att-par> is an attribute ("a=") in its full '<type>=<value>' form (see [RFC4566]).

The "acap" attribute adheres to the RFC 4566 "attribute" production, with an att-value defined as follows:

```
att-value   = att-cap-num 1*WSP att-par
att-cap-num = 1*DIGIT ;defined in [RFC4234]
att-par     = attribute  ;defined in RFC 4566
```

Note that white-space is not permitted before the att-cap-num.

The "acap" attribute can be provided at the session level only when the attribute capability contains session-level attributes, whereas

media level attributes can be provided in attribute capabilities at either the media level or session-level. The base SDP Capability Negotiation framework however only defines procedures for use of media-level attribute capabilities at the media level (extensions may define use at the session level).

Each occurrence of the "acap" attribute in the entire session description MUST use a different value of <att-cap-num>.

There is a need to be able to reference both session-level and media-level attributes in potential configurations at the media level, and this provides for a simple solution to avoiding overlap between the references (handles) to each attribute capability.

The <att-cap-num> values provided are independent of similar <cap-num> values provided for other types of capabilities, i.e., they form a separate name-space for attribute capabilities.

The following examples illustrate use of the "acap" attribute:

```
a=acap:1 ptime:20

a=acap:2 ptime:30

a=acap:3 key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyONQ6gAA
AAAGEEoo2pee4hp2UaDX8ZE22YwKAAAPZG9uYWxkQGR1Y2suY29tAQAAAAAAQAk0
JKpgaVkDaawi9whVBtBt0KZ14ymNuu62+Nv3ozPLygwK/GbAV9iemnGUIZ19fWQUO
SrzKTAv9zV

a=acap:4 crypto:1 AES_CM_128_HMAC_SHA1_32
      inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
```

The first two attribute capabilities provide attribute values for the ptime attribute. The third provides SRTP parameters by using MIKEY [RFC3830] with the key-mgmt attribute [RFC4567]. The fourth provides SRTP parameters by use of security descriptions with the crypto attribute [RFC4568]. Note that the line-wrapping and new-lines in example three and four are provided for formatting reasons only - they are not permitted in actual SDP.

Readers familiar with RFC 3407 may notice the similarity between the RFC 3407 "cpar" attribute and the above. There are however a couple of important differences, notably that the "acap" attribute contains a handle that enables referencing it and it furthermore supports attributes only (the "cpar" attribute defined in RFC 3407 supports bandwidth information as well). The "acap" attribute also

is not automatically associated with any particular capabilities. See Section 3.14. for the relationship to RFC 3407.

3.4.2. Transport Protocol Capability Attribute

Transport Protocols can be expressed as capabilities by use of a new Transport Protocol Capability attribute ("a=tcap") defined as follows:

a=tcap: <trpr-cap-num> <proto-list>

where <trpr-cap-num> is an integer between 1 and 2^31-1 (both included) used to number the transport address capability for later reference, and <proto-list> is one or more <proto>, separated by white space, as defined in the SDP "m=" line.

The "tcap" attribute adheres to the RFC 4566 "attribute" production, with an att-value defined as follows:

```
att-value      = trpr-cap-num 1*WSP proto-list
trpr-cap-num   = 1*DIGIT ;defined in [RFC4234]
proto-list     = proto *(1*WSP proto) ; defined in RFC 4566
```

Note that white-space is not permitted before the trpr-cap-num.

The "tcap" attribute can be provided at the session-level and the media-level. There can be at most one "a=tcap" attribute at the session-level and at most one at the media-level (one per media description in the latter case). Each occurrence of the "tcap" attribute in the entire session description MUST use a different value of <trpr-cap-num>. When multiple <proto> values are provided, the first one is associated with the value <trpr-cap-num>, the second one with the value one higher, etc. There MUST NOT be any capability number overlap between different "tcap" attributes in the entire SDP. The <trpr-cap-num> values provided are independent of similar <cap-num> values provided for other capability attributes, i.e., they form a separate name-space for transport protocol capabilities.

Below, we provide examples of the "a=tcap" attribute:

a=tcap:1 RTP/AVP

a=tcap:2 RTP/AVPF

a=tcap:3 RTP/SAVP RTP/SAVPF

The first one provides a capability for the "RTP/AVP" profile defined in [RFC3551] and the second one provides a capability for the RTP with RTCP-Based Feedback profile defined in [RFC4585]. The third one provides capabilities for the "RTP/SAVP" (transport capability number 3) and "RTP/SAVPF" profiles (transport protocol capability number 4).

Transport capabilities are inherently included in the "m=" line, however they still need to be specified explicitly in a "tcap" attribute if they are to be used as a capability.

This may seem redundant (and indeed it is from the offerer's point of view), however it is done to protect against intermediaries (e.g. middle-boxes) that may modify "m=" lines while passing unknown attributes through. If an implicit transport capability were used instead (e.g. a reserved transport capability number could be used to refer to the transport protocol in the "m=" line), and an intermediary were to modify the transport protocol in the "m=" line (e.g. to translate between plain RTP and secure RTP), then the potential configuration referencing that implicit transport capability may no longer be correct. With explicit capabilities, we avoid this pitfall; however, the potential configuration preference (see Section 3.5.1.) may not reflect that of the intermediary (which some may view as a feature).

3.4.3. Extension Capability Attributes

The SDP Capability Negotiation framework allows for new types of capabilities to be defined as extensions and used with the general capability negotiation framework. The syntax and semantics of such new capability attributes are not defined here, however in order to be used with potential configurations, they SHOULD allow for a numeric handle to be associated with each capability. This handle can be used as a reference within the potential and actual configuration attributes (see Section 3.5.1. and 3.5.2.). The definition of such extension capability attributes MUST also state whether they can be applied at the session-level, media-level, or both.

3.5. Configuration Attributes

3.5.1. Potential Configuration Attribute

Potential Configurations can be expressed by use of a new Potential Configuration Attribute ("a=pcfg") defined as follows:

```
a=pcfg: <config-number> [<pot-cfg-list>]
```

where <config-number> is an integer between 1 and 2^31-1 (both included).

The "pcfg" attribute adheres to the RFC 4566 "attribute" production, with an att-value defined as follows:

```
att-value      = config-number [1*WSP pot-cfg-list]
config-number  = 1*DIGIT ;defined in [RFC4234]
pot-cfg-list   = pot-config *(1*WSP pot-config)
pot-config     = attribute-config-list /
                 transport-protocol-config-list /
                 extension-config-list
```

The missing productions are defined below. Note that white-space is not permitted before the config-number.

The potential configuration attribute can be provided at the media-level only and there can be multiple instances of it within a given media description. The attribute includes a configuration number, which is an integer between 1 and 2^31-1 (both included). The configuration number MUST be unique within the media description (i.e. it has media level scope only). The configuration number also indicates the relative preference of potential configurations; lower numbers are preferred over higher numbers.

A potential configuration list is normally provided after the configuration number. When the potential configuration list is omitted, the potential configuration equals the actual configuration. The potential configuration list contains one or more of attribute, transport and extension configuration lists. The configuration lists generally reference one or more capabilities (extension configuration lists MAY use a different format). Those capabilities are (conceptually) used to construct a new internal version of the SDP by use of purely syntactic add and (possibly) delete operations on the original SDP (actual configuration). This provides an alternative potential configuration SDP that can be used by conventional SDP and offer/answer procedures if selected.

This document defines attribute configuration lists and transport protocol configuration lists. Each of these MUST NOT be present more than once in a particular potential configuration attribute. Extension configuration lists can be included as well. There can be more than one extension configuration list, however each particular extension MUST NOT be present more than once in a given "a=pcfg"

attribute. Together, the various configuration lists define a potential configuration.

There can be multiple potential configurations in a media description. Each of these indicates not only a willingness, but in fact a desire to use the potential configuration.

The example SDP below contains two potential configurations:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVP 0 18
a=tcap:1 RTP/SAVP RTP/SAVPF
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=pcfg:1 t=1 a=1
a=pcfg:2 t=2 a=1
```

Potential configuration 1 contains a transport protocol configuration list that references transport capability 1 ("RTP/SAVP") and an attribute configuration list that references attribute capability 1 ("a=crypto:..."). Potential configuration 2 contains a transport protocol configuration list that references transport capability 2 ("RTP/SAVPF") and an attribute configuration list that references attribute capability 1 ("a=crypto:...").

Attribute capabilities are used in a potential configuration by use of the attribute-config-list parameter, which is defined by the following ABNF:

```
attribute-config-list
                  = "a=" [delete-attributes ":"]
                           mo-att-cap-list *(BAR mo-att-cap-list)

delete-attributes = DELETE ( "m"    ; media attributes
                       / "s"    ; session attributes
                       / "ms" ) ; media and session attributes

mo-att-cap-list      = mandatory-optional-att-cap-list |
                              mandatory-att-cap-list |
                                 optional-att-cap-list
```

```
mandatory-optional-att-cap-list  = mandatory-att-cap-list
                                       "," optional-att-cap-list
mandatory-att-cap-list           = att-cap-list
optional-att-cap-list            = "[" att-cap-list "]"

att-cap-list      = att-cap-num *("," att-cap-num)
att-cap-num       = 1*DIGIT   ;defined in [RFC4234]
BAR               = "|"
DELETE            = "-"
```

Note that white space is not permitted within this production.

Each attribute configuration list can optionally begin with instructions for how to handle attributes that are part of the actual configuration SDP (i.e., the "a=" lines present in the original SDP). By default, such attributes will remain as part of the configuration in question. However, if delete-attributes indicates "-m", then all attribute lines within the media description in question will be deleted (i.e., all "a=" lines under the "m=" line in question). If delete-attributes indicates "-s", then all attribute lines at the session-level will be deleted (i.e., all "a=" lines before the first "m=" line). If delete-attributes indicates "-ms", then all attribute lines within this media description ("m=" line) and all attribute lines at the session-level will be deleted.

The attribute capability list comes next. It contains one or more alternative lists of attribute capabilities. The alternative attribute capability lists are separated by a vertical bar ("|"), and each list contains one or more attribute capabilities separated by commas (","). The attribute capabilities are either mandatory or optional. Mandatory attribute capabilities MUST be supported in order to use the potential configuration, whereas optional attribute capabilities MAY be supported in order to use the potential configuration.

Within each attribute capability list, all the mandatory attribute capabilities (if any) are listed first, and all the optional attribute capabilities (if any) are listed last. The optional attribute capabilities are contained within a pair of angle brackets ("[" and "]"). Each attribute capability is merely an attribute capability number (att-cap-num) that identifies a particular attribute capability by referring to attribute capability numbers defined above and hence MUST be between 1 and 2^31-1 (both included). The following example illustrates the above:

```
a=pcfg:1 a=-m:1,2,[3,4]|1,7,[5]
```

where

- "a=-m:1,2,[3,4]|1,7,[5]" is the attribute configuration list
- "-m" is the delete-attributes
- "1,2,[3,4]" and "1,7,[5]" are both attribute capability lists. The two lists are alternatives, since they are separated by a vertical bar above
- "1", "2" and "7" are mandatory attribute capabilities
- "3", "4" and "5" are optional attribute capabilities

Note that in the example above, we have a single handle ("1") for the potential configuration(s), but there are actually two different potential configurations (separated by a vertical bar). This is done for message size efficiency reasons, which is especially important when we add other types of capabilities to the potential configuration. If there is a need to provide a unique handle for each, then separate "a=pcfg" attributes with different handles MUST be used instead.

Each referenced attribute capability in the potential configuration will result in the corresponding attribute name and its associated value (contained inside the attribute capability) being added to the resulting potential configuration SDP.

Alternative attribute capability lists are separated by a vertical bar ("|"), the scope of which extends to the next alternative (i.e., "," has higher precedence than "|"). The alternatives are ordered by preference with the most preferred listed first. In order for a recipient of the SDP (e.g., an answerer receiving this in an offer) to use this potential configuration, exactly one of the alternative lists MUST be selected in its entirety. This requires that all mandatory attribute capabilities referenced by the potential configuration are supported with the attribute values provided.

Transport protocol configuration lists are included in a potential configuration by use of the transport-protocol-config-list parameter, which is defined by the following ABNF:

```
   transport-protocol-config-list =
                        "t=" trpr-cap-num *(BAR trpr-cap-num)
   trpr-cap-num        = 1*DIGIT   ; defined in [RFC4234]
```

Note that white-space is not permitted within this production.

The trpr-cap-num refers to transport protocol capability numbers defined above and hence MUST be between 1 and 2^31-1 (both included). Alternative transport protocol capabilities are separated by a vertical bar ("|"). The alternatives are ordered by preference with the most preferred listed first. If there are no transport protocol capabilities included in a potential configuration at the media level, the transport protocol information from the associated "m=" line MUST be used. In order for a recipient of the SDP (e.g., an answerer receiving this in an offer) to use this potential configuration, exactly one of the alternatives MUST be selected. This requires that the transport protocol in question is supported.

In the presence of intermediaries (the existence of which may not be known), care should be taken with assuming that the transport protocol in the "m=" line will not be modified by an intermediary. Use of an explicit transport protocol capability will guard against capability negotiation implications of that.

Extension capabilities can be included in a potential configuration as well by use of extension configuration lists. Such extension configuration lists MUST adhere to the following ABNF:

```
   extension-config-list= ["+"] ext-cap-name "="
                                ext-cap-list
   ext-cap-name              = 1*(ALPHA / DIGIT)
   ext-cap-list              = 1*VCHAR      ; defined in [RFC4234]
```

Note that white-space is not permitted within this production.

The ext-cap-name refers to the name of the extension capability and the ext-cap-list is here merely defined as a sequence of visible characters. The actual extension supported MUST refine both of these further. For extension capabilities that merely need to be referenced by a capability number, it is RECOMMENDED to follow a structure similar to what has been specified above. Unsupported or unknown potential extension configuration lists in a potential configuration attribute MUST be ignored, unless they are prefixed with the plus ("+") sign, which indicates that the extension is mandatory and MUST be supported in order to use that potential configuration.

The "creq" attribute and its associated rules can be used to ensure that required extensions are supported in the first place.

Potential configuration attributes can be provided at the media level only, however it is possible to reference capabilities provided at either the session or media level. There are certain semantic rules and restrictions associated with this:

A (media level) potential configuration attribute in a given media description MUST NOT reference a media-level capability provided in a different media description; doing so invalidates that potential configuration (note that a potential configuration attribute can contain more than one potential configuration by use of alternatives). A potential configuration attribute can however reference a session-level capability. The semantics of doing so depends on the type of capability. In the case of transport protocol capabilities it has no particular implication. In the case of attribute capabilities however, it does. More specifically, the attribute name and value (provided within that attribute capability) will be considered part of the resulting SDP for that particular configuration at the *session* level. In other words, it will be as-if that attribute was provided with that value at the session-level in the first place. As a result, the base SDP Capability Negotiation framework REQUIRES that potential configurations do not reference any session-level attribute capabilities that contain media-level attributes (since that would place a media-level attribute at the session level). Extensions may modify this behavior, as long as it is fully backwards compatible with the base specification.

Individual media streams perform capability negotiation individually, and hence it is possible that one media stream (where the attribute was part of a potential configuration) chose a configuration without a session level attribute that was chosen by another media stream. The session-level attribute however remains "active" and applies to the entire resulting potential configuration SDP. In theory, this is problematic if one or more session-level attributes either conflicts with or potentially interacts with another session-level or media-level attribute in an undefined manner. In practice, such examples seem to be rare (at least with the currently defined SDP attributes).

A related set of problems can occur if we need coordination between session-level attributes from multiple media streams in order for a particular functionality to work. The grouping framework [RFC3388] is an example of this. If we use the SDP Capability Negotiation framework to select a session-level group

attribute (provided as an attribute capability), and we require two media descriptions to do this consistently, we could have a problem. The FEC grouping semantics [RFC4756] is one example where this in theory could cause problems, however in practice, it is unclear that there is a significant problem with the currently defined grouping semantics.

Resolving the above issues in general requires inter-media stream constraints and synchronized potential configuration processing; this would add considerable complexity to the overall solution. In practice, with the currently defined SDP attributes, it does not seem to be a significant problem, and hence the core SDP Capability Negotiation solution does not provide a solution to this issue. Instead, it is RECOMMENDED that use of session-level attributes in a potential configuration is avoided when possible, and when not, that such use is examined closely for any potential interaction issues. If interaction is possible, the entity generating the SDP SHOULD NOT assume that well-defined operation will occur at the receiving entity.

The session-level operation of extension capabilities is undefined: Consequently, each new session-level extension capability defined MUST specify the implication of making it part of a configuration at the media level.

Below, we provide an example of the "a=pcfg" attribute in a complete media description in order to properly indicate the supporting attributes:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVPF 0 18
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=tcap:1 RTP/AVPF RTP/AVP
a=tcap:3 RTP/SAVP RTP/SAVPF
a=pcfg:1 t=4|3 a=1
a=pcfg:8 t=1|2
```

We have two potential configuration attributes listed here. The first one (and most preferred, since its configuration number is "1") indicates that either of the profiles RTP/SAVPF or RTP/SAVP (specified by the transport protocol capability numbers 4 and 3) can

be supported with attribute capability 1 (the "crypto" attribute); RTP/SAVPF is preferred over RTP/SAVP since its capability number (4) is listed first in the preferred potential configuration. Note that although we have a single potential configuration attribute and associated handle, we have two potential configurations.

The second potential configuration attribute indicates that the RTP/AVPF or RTP/AVP profiles can be used, with RTP/AVPF being the preferred one. This non secure RTP alternative is the less preferred one since its configuration number is "8". Again, note that we have two potential configurations here and hence a total of four potential configurations in the SDP above.

3.5.2. Actual Configuration Attribute

The actual configuration attribute identifies which of the potential configurations from an offer SDP was selected and used as the actual configuration to generate an answer SDP. This is done by including the configuration number and the configuration lists (if any) from the offer that were selected and used by the answerer in his offer/answer procedure as follows:

- A selected attribute configuration MUST include the delete-attributes and the selected alternative mo-att-cap-list (i.e., containing all mandatory and optional capability numbers from the potential configuration, irrespective of whether the optional ones were supported or not). If delete-attributes were not included in the potential configuration, they will of course not be present here either.

- A selected transport protocol configuration MUST include the selected transport protocol capability number.

- A selected potential extension configuration MUST include the selected extension configuration parameters as specified for that particular extension.

- When a configuration list contains alternatives (separated by "|"), the selected configuration only MUST be provided.

Note that the selected configuration number and all selected capability numbers used in the actual configuration attribute refer to those from the offer; not the answer.

The answer may for example include capabilities as well to inform the offerer of the answerers capabilities above and beyond the

negotiated configuration. The actual configuration attribute does not refer to any of those answer capabilities though.

The Actual Configuration Attribute ("a=acfg") is defined as follows:

```
a=acfg: <config-number> [<sel-cfg-list>]
```

where <config-number> is an integer between 1 and 2^31-1 (both included).

The "acfg" attribute adheres to the RFC 4566 "attribute" production, with an att-value defined as follows:

```
att-value      = config-number [1*WSP sel-cfg-list]
                   ;config-number defined in Section 3.5.1.
sel-cfg-list   = sel-cfg *(1*WSP sel-cfg)
sel-cfg        = sel-attribute-config /
                      sel-transport-protocol-config /
                      sel-extension-config

sel-attribute-config =
         "a=" [delete-attributes ":"] mo-att-cap-list
                                ; defined in Section 3.5.1.

sel-transport-protocol-config =
         "t=" trpr-cap-num    ; defined in Section 3.5.1.

sel-extension-config =
         ext-cap-name "=" 1*VCHAR   ; defined in Section 3.5.1.
```

Note that white-space is not permitted before the config-number.

The actual configuration ("a=acfg") attribute can be provided at the media-level only. There MUST NOT be more than one occurrence of an actual configuration attribute within a given media description.

Below, we provide an example of the "a=acfg" attribute (building on the previous example with the potential configuration attribute):

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/SAVPF 0
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:WSJ+PSdFcGdUJShpX1ZjNzB4d1BINUAvLEw6UzF3|2^20|1:32
a=acfg:1 t=4 a=1
```

It indicates that the answerer used an offer consisting of potential configuration number 1 with transport protocol capability 4 from the offer (RTP/SAVPF) and attribute capability 1 (the "crypto" attribute). The answerer includes his own "crypto" attribute as well.

3.6. Offer/Answer Model Extensions

In this section, we define extensions to the offer/answer model defined in [RFC3264] to allow for potential configurations to be included in an offer, where they constitute alternative offers that may be accepted by the answerer instead of the actual configuration(s) included in the "m=" line(s).

The procedures defined in the following subsections apply to both unicast and multicast streams.

3.6.1. Generating the Initial Offer

An offerer that wants to use the SDP Capability Negotiation defined in this document MUST include the following in the offer:

- An attribute capability attribute ("a=acap") as defined in Section 3.4.1. for each attribute name and associated value (if any) that needs to be indicated as a capability in the offer.

  Session-level attributes and associated values MUST be provided in attribute capabilities at the session-level only, whereas media-level attributes and associated values can be provided in attribute capabilities at either the media-level or session-level. Attributes that are allowed at either the session- or media-level can be provided in attribute capabilities at either level. If there is no need to indicate any attributes as attribute capabilities, then there will not be any "a=acap" attributes either.

- One or more transport protocol capability attributes ("a=tcap") as defined in Section 3.4.2. with values for each transport protocol that needs to be indicated as a capability in the offer. Transport protocol capabilities that apply to multiple media descriptions SHOULD be provided at the session-level whereas transport protocol capabilities that apply to a specific media description ("m=" line) only, SHOULD be provided within that particular media description. In either case, there MUST NOT be more than a single "a=tcap" attribute at the session-level and a single "a=tcap" attribute in each media description. If there is no need to indicate any transport protocols as transport protocol capabilities, then there will not be any "a=tcap" attributes either.

- One or more extension capability attributes (as outlined in Section 3.4.3.) for each extension capability that is referenced by a potential configuration. Extension capability attributes that are not referenced by a potential configuration MAY be provided as well.

- One or more potential configuration attributes ("a=pcfg"), as defined in Section 3.5.1. , in each media description where alternative potential configurations are to be negotiated. Each potential configuration attribute MUST adhere to the rules provided in Section 3.5.1. and the additional rules provided below.

If the offerer requires support for more or extensions (besides the base protocol defined here), then the offerer MUST include one or more "a=creq" attribute as follows:

- If support for one or more capability negotiation extensions is required for the entire session description, then option tags for those extensions MUST be included in a single session-level "creq" attribute.

- For each media description that requires support for one or more capability negotiation extensions not listed at the session-level, a single "creq" attribute containing all the required extensions for that media description MUST be included within the media description (in accordance with Section 3.3.2.).

Note that extensions that only need to be supported by a particular potential configuration can use the "mandatory" extension prefix ("+") within the potential configuration (see Section 3.5.1.).

The offerer SHOULD furthermore include the following:

- A supported capability negotiation extension attribute ("a=csup") at the session-level and/or media-level as defined in Section 3.3.2. for each capability negotiation extension supported by the offerer and not included in a corresponding "a=creq" attribute (i.e., at the session-level or in the same media description). Option tags provided in a "a=csup" attribute at the session-level indicate extensions supported for the entire session description, whereas option tags provided in a "a=csup" attribute in a media description indicate extensions supported for that particular media description only.

Capabilities provided in an offer merely indicate what the offerer is capable of doing. They do not constitute a commitment or even an indication to use them. In contrast, each potential configuration constitutes an alternative offer that the offerer would like to use. The potential configurations MUST be used by the answerer to negotiate and establish the session.

The offerer MUST include one or more potential configuration attributes ("a=pcfg") in each media description where the offerer wants to provide alternative offers (in the form of potential configurations). Each potential configuration attribute in a given media description MUST contain a unique configuration number and one or more potential configuration lists, as described in Section 3.5.1. Each potential configuration list MUST refer to capabilities that are provided at the session-level or within that particular media description; otherwise, the potential configuration is considered invalid. The base SDP Capability Negotiation framework REQUIRES that potential configurations do not reference any session-level attribute capabilities that contain media-level only attributes, however extensions may modify this behavior, as long as it is fully backwards compatible with the base specification. Furthermore, it is RECOMMENDED that potential configurations avoid use of session-level capabilities whenever possible; refer to Section 3.5.1.

The current actual configuration is included in the "m=" line (as defined by [RFC3264]) and any associated parameters for the media description (e.g., attribute ("a=") and bandwidth ("b=") lines). Note that the actual configuration is by default the least-preferred configuration, and hence the answerer will seek to negotiate use of one of the potential configurations instead. If the offerer wishes a different preference for the actual configuration, the offerer MUST include a corresponding potential configuration with the relevant

configuration number (which indicates the relative preference between potential configurations); this corresponding potential configuration should simply duplicate the actual configuration.

This can either be done implicitly (by not referencing any capabilities), or explicitly (by providing and using capabilities for the transport protocol and all the attributes that are part of the actual configuration). The latter may help detect intermediaries that modify the actual configuration but are not SDP Capability Negotiation aware.

Per [RFC3264], once the offerer generates the offer, he must be prepared to receive incoming media in accordance with that offer. That rule applies here as well, but for the actual configurations provided in the offer only: Media received by the offerer according to one of the potential configurations MAY be discarded, until the offerer receives an answer indicating what the actual selected configuration is. Once that answer is received, incoming media MUST be processed in accordance with the actual selected configuration indicated and the answer received (provided the offer/answer exchange completed successfully).

The above rule assumes that the offerer can determine whether incoming media adheres to the actual configuration offered or one of the potential configurations instead; this may not always be the case. If the offerer wants to ensure he does not play out any garbage, the offerer SHOULD discard all media received before the answer SDP is received. Conversely, if the offerer wants to avoid clipping, he should attempt to play any incoming media as soon as it is received (at the risk of playing out garbage). For further details, please refer to Section 3.9.

3.6.2. Generating the Answer

When receiving an offer, the answerer MUST check for the presence of a required capability negotiation extension attribute ("a=creq") provided at the session level. If one is found, then capability negotiation MUST be performed. If none is found, then the answerer MUST check each offered media description for the presence of a required capability negotiation extension attribute ("a=creq") and one or more potential configuration attributes ("a=pcfg"). Capability negotiation MUST be performed for each media description where either of those is present in accordance with the procedures described below.

The answerer MUST first ensure that it supports any required capability negotiation extensions:

- If a session-level "creq" attribute is provided, and it contains an option-tag that the answerer does not support, then the answerer MUST NOT use any of the potential configuration attributes provided for any of the media descriptions. Instead, the normal offer/answer procedures MUST continue as per [RFC3264]. Furthermore, the answerer MUST include a session-level supported capability negotiation extensions attribute ("a=csup") with option tags for the capability negotiation extensions supported by the answerer.

- If a media-level "creq" attribute is provided, and it contains an option tag that the answerer does not support, then the answerer MUST NOT use any of the potential configuration attributes provided for that particular media description. Instead, the offer/answer procedures for that media description MUST continue as per [RFC3264] (SDP Capability Negotiation is still performed for other media descriptions in the SDP). Furthermore, the answerer MUST include a supported capability negotiation extensions attribute ("a=csup") in that media description with option tags for the capability negotiation extensions supported by the answerer for that media description.

Assuming all required capability negotiation extensions are supported, the answerer now proceeds as follows.

For each media description where capability negotiation is to be performed (i.e. all required capability negotiation extensions are supported and at least one valid potential configuration attribute is present), the answerer MUST attempt to perform capability negotiation by using the most preferred potential configuration that is valid to the answerer. A potential configuration is valid to the answerer if:

1. It is in accordance with the syntax and semantics provided in Section 3.5.1.

2. It contains a configuration number that is unique within that media description.

3. All attribute capabilities referenced by the potential configuration are valid themselves (as defined in Section 3.4.1.) and each of them is provided either at the session-level or within this particular media description. For session-level attribute capabilities referenced, the attributes contained inside them MUST NOT be media-level only attributes.

4. All transport protocol capabilities referenced by the potential configuration are valid themselves (as defined in Section 3.4.2.) and each of them is furthermore provided either at the session-level or within this particular media description.

5. All extension capabilities referenced by the potential configuration and supported by the answerer are valid themselves (as defined by that particular extension) and each of them are furthermore provided either at the session-level or within this particular media description. Unknown or unsupported extension capabilities MUST be ignored, unless they are prefixed with the plus ("+") sign, which indicates that the extension MUST be supported in order to use that potential configuration. If the extension is not supported, that potential configuration is not valid to the answerer.

The most preferred valid potential configuration in a media description is the valid potential configuration with the lowest configuration number. The answerer MUST now process the offer for that media stream based on the most preferred valid potential configuration. Conceptually, this entails the answerer constructing an (internal) offer that consists of the actual configuration offer SDP, with the following changes for each media stream offered:

- If a transport protocol capability is included in the potential configuration, then it replaces the transport protocol provided in the "m=" line for that media description.

- If attribute capabilities are present with a delete-attributes session indication ("-s"), then all session-level attributes from the actual configuration SDP MUST be deleted in accordance with the procedures in Section 3.5.1. If attribute capabilities are present with a delete-attributes media indication ("-m"), then all attributes from the actual configuration SDP inside this media description MUST be deleted.

- If a session-level attribute capability is included, the attribute (and its associated value, if any) contained in it MUST be added to the resulting SDP. All such added session-level attributes MUST be listed before the session-level attributes that were initially present in the SDP. Furthermore, the added session-level attributes MUST be added in the order they were provided in the potential configuration (see also Section 3.5.1.).

  This allows for attributes with implicit preference ordering to be added in the desired order; the "crypto" attribute [RFC4568] is one such example.

- If a media-level attribute capability is included, then the attribute (and its associated value, if any) MUST be added to the resulting SDP within the media description in question. All such added media-level attributes MUST be listed before the media-level attributes that were initially present in the SDP in the media description in question. Furthermore, the added media-level attributes MUST be added in the order they were provided in the potential configuration (see also Section 3.5.1.).

- If a supported extension capability is included, then it MUST be processed in accordance with the rules provided for that particular extension capability.

Note that a transport protocol from the potential configuration replaces the transport protocol in the actual configuration, but an attribute capability from the potential configuration is simply added to the actual configuration. In some cases, this can result in having one or more meaningless attributes in the resulting potential configuration SDP, or worse, ambiguous or potentially even illegal attributes. Use of delete-attributes for the session and/or media level attributes MUST be done to avoid such scenarios. Nevertheless, it is RECOMMENDED that implementations ignore meaningless attributes that may result from potential configurations.

For example, if the actual configuration was using Secure RTP and included an "a=crypto" attribute for the SRTP keying material, then use of a potential configuration that uses plain RTP would make the "crypto" attribute meaningless. The answerer may or may not ignore such a meaningless attribute. The offerer can here ensure correct operation by using delete-attributes to remove the crypto attribute (but will then need to provide attribute capabilities to reconstruct the SDP with the necessary attributes deleted, e.g. rtpmaps).

Please refer to Section 3.6.2.1. for examples of how the answerer may conceptually "see" the resulting offered alternative potential configurations.

The answerer MUST check that he supports all mandatory attribute capabilities from the potential configuration (if any), the transport protocol capability (if any) from the potential configuration, and all mandatory extension capabilities from the potential configuration (if any) in accordance with the rules provided for these. If he does not, the answerer MUST proceed to the second-most preferred valid potential configuration for the media description, etc. In the case of attribute capabilities, support implies that the attribute name contained in the capability is supported and it can (and will) be used successfully in the negotiation process with the value provided. This does not necessarily imply that the value provided is supported in its entirety. For example, the "a=fmtp" parameter is often provided with one or more values in a list, where the offerer and answerer negotiate use of some subset of the values provided. Other attributes may include mandatory and optional parts to their values; support for the mandatory part is all that is required here.

A side-effect of the above rule is that whenever an "fmtp" or "rtpmap" parameter is provided as a mandatory attribute capability, the corresponding media format (codec) must be supported and use of it negotiated successfully. If this is not the offerer's intent, the corresponding attribute capabilities must be listed as optional instead.

If the answerer has exhausted all potential configurations for the media description, without finding a valid one that is also supported, then the answerer MUST process the offered media stream based on the actual configuration plus any session-level attributes added by a valid and supported potential configuration from another media description in the offered SDP.

The above process describes potential configuration selection as a per media stream process. Inter-media stream coordination of selected potential configurations however is required in some cases. First of all, session-level attributes added by a potential configuration for one media description MUST NOT cause any problems for potential configurations selected by other media descriptions in the offer SDP. If the session-level attributes are mandatory, then those session-level attributes MUST furthermore be supported by the session as a whole (i.e., all the media descriptions if relevant). As mentioned earlier, this adds additional complexity to the overall

processing and hence it is RECOMMENDED not to use session-level attribute capabilities in potential configurations, unless absolutely necessary.

Once the answerer has selected a valid and supported offered potential configuration for all of the media streams (or has fallen back to the actual configuration plus any added session attributes), the answerer MUST generate a valid answer SDP based on the selected potential configuration SDP, as "seen" by the answerer (see Section 3.6.2.1. for examples). Furthermore, if the answerer selected one of the potential configurations in a media description, the answerer MUST include an actual configuration attribute ("a=acfg") within that media description. The "a=acfg" attribute MUST identify the configuration number for the selected potential configuration as well as the actual parameters that were used from that potential configuration; if the potential configuration included alternatives, the selected alternatives only MUST be included. Only the known and supported parameters will be included. Unknown or unsupported parameters MUST NOT be included in the actual configuration attribute. In the case of attribute capabilities, only the known and supported capabilities are included; unknown or unsupported attribute capabilities MUST NOT be included.

If the answerer supports one or more capability negotiation extensions that were not included in a required capability negotiation extensions attribute in the offer, then the answerer SHOULD furthermore include a supported capability negotiation attribute ("a=csup") at the session-level with option tags for the extensions supported across media streams. Also, if the answerer supports one or more capability negotiation extensions for particular media descriptions only, then a supported capability negotiation attribute with those option-tags SHOULD be included within each relevant media description.

The offerer's originally provided actual configuration is contained in the offer media description's "m=" line (and associated parameters). The answerer MAY send media to the offerer in accordance with that actual configuration as soon as it receives the offer, however it MUST NOT send media based on that actual configuration if it selects an alternative potential configuration. If the answerer selects one of the potential configurations, then the answerer MAY immediately start to send media to the offerer in accordance with the selected potential configuration, however the offerer MAY discard such media or play out garbage until the offerer receives the answer. Please refer to section 3.9. for additional

considerations and possible alternative solutions outside the base SDP Capability Negotiation framework.

If the offerer selected a potential configuration instead of the actual configuration, then it is RECOMMENDED that the answerer sends back an answer SDP as soon as possible. This minimizes the risk of having media discarded or played out as garbage by the offerer. In the case of SIP [RFC3261] without any extensions, this implies that if the offer was received in an INVITE message, then the answer SDP should be provided in the first non-100 provisional response sent back (per RFC3261, the answer would need to be repeated in the 200 response as well, unless a relevant extension such as [RFC3262] is being used).

3.6.2.1. Example Views of Potential Configurations

The following examples illustrate how the answerer may conceptually "see" a potential configuration. Consider the following offered SDP:

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.example.com
s=
t=0 0
c=IN IP4 lost.example.com
a=tool:foo
a=acap:1 key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
a=tcap:1 RTP/SAVP RTP/AVP
m=audio 59000 RTP/AVP 98
a=rtpmap:98 AMR/8000
a=acap:2 crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=pcfg:1 t=1 a=1|2
m=video 52000 RTP/AVP 31
a=rtpmap:31 H261/90000
a=acap:3 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=pcfg:1 t=1 a=1|3
```

This particular SDP offers an audio stream and a video stream, each of which can either use plain RTP (actual configuration) or secure RTP (potential configuration). Furthermore, two different keying mechanisms are offered, namely session-level Key Management Extensions using MIKEY (attribute capability 1) and media-level SDP Security Descriptions (attribute capabilities 2 and 3). There are several potential configurations here, however, below we show the

one the answerer "sees" when using potential configuration 1 for both audio and video, and furthermore using attribute capability 1 (MIKEY) for both (we have removed all the capability negotiation attributes for clarity):

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.example.com
s=
t=0 0
c=IN IP4 lost.example.com
a=tool:foo
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
m=audio 59000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 52000 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

Note that the transport protocol in the media descriptions indicate use of secure RTP.

Below, we show the offer the answerer "sees" when using potential configuration 1 for both audio and video and furthermore using attribute capability 2 and 3 respectively (SDP security descriptions) for the audio and video stream - note the order in which the resulting attributes are provided:

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.example.com
s=
t=0 0
c=IN IP4 lost.example.com
a=tool:foo
m=audio 59000 RTP/SAVP 98
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=rtpmap:98 AMR/8000
m=video 52000 RTP/SAVP 31
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
   a=rtpmap:31 H261/90000
```

Again, note that the transport protocol in the media descriptions indicate use of secure RTP.

And finally, we show the offer the answerer "sees" when using potential configuration 1 with attribute capability 1 (MIKEY) for the audio stream, and potential configuration 1 with attribute capability 3 (SDP security descriptions) for the video stream:

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.example.com
s=
t=0 0
c=IN IP4 lost.example.com
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
a=tool:foo
m=audio 59000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 52000 RTP/SAVP 31
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=rtpmap:31 H261/90000
```

3.6.3. Offerer Processing of the Answer

When the offerer attempted to use SDP Capability Negotiation in the offer, the offerer MUST examine the answer for actual use of SDP Capability Negotiation.

For each media description where the offerer included a potential configuration attribute ("a=pcfg"), the offerer MUST first examine that media description for the presence of an actual configuration attribute ("a=acfg"). If an actual configuration attribute is not present in a media description, then the offerer MUST process the answer SDP for that media stream per the normal offer/answer rules defined in [RFC3264]. However, if one is found, the offerer MUST instead process the answer as follows:

- The actual configuration attribute specifies which of the potential configurations was used by the answerer to generate the answer for this media stream. This includes all the supported attribute capabilities and the transport capabilities referenced by the potential configuration selected, where the attribute capabilities have any associated delete-attributes included. Extension capabilities supported by the answerer are included as well.

- The offerer MUST now process the answer in accordance with the rules in [RFC3264], except that it must be done as if the offer consisted of the selected potential configuration instead of the original actual configuration, including any transport protocol changes in the media ("m=") line(s), attributes added and deleted by the potential configuration at the media and session level, and any extensions used.

If the offer/answer exchange was successful, and if the answerer selected one of the potential configurations from the offer as the actual configuration, then the offerer MAY perform another offer/answer exchange: This new offer SHOULD contain the selected potential configuration as the actual configuration, i.e., with the actual configuration used in the "m=" line and any other relevant attributes and extensions. This second offer/answer exchange will not modify the session in any way, however it will help intermediaries (e.g. middleboxes) that look at the SDP, but do not understand or support the capability negotiation extensions, to understand the details of the media stream(s) that were actually negotiated. If it is known or suspected that one or more such intermediaries exist, then this second offer/answer SHOULD be performed (this is already done when using Interactive Connectivity Establishment [ICE], and in those cases, there will not be a need for a third offer/answer exchange). Note that, per normal offer/answer rules, the second offer/answer exchange still needs to update the version number in the "o=" line ((<sess-version> in [RFC4566]). Attribute lines carrying keying material SHOULD repeat the keys from the previous offer, unless re-keying is necessary, e.g. due to a previously forked SIP INVITE request. Please refer to Section 3.12. for additional considerations related to intermediaries.

3.6.4. Modifying the Session

Capabilities and potential configurations may be included in subsequent offers as defined in [RFC3264], Section 8. The procedure for doing so is similar to that described above with the answer including an indication of the actual selected configuration used by the answerer.

If the answer indicates use of a potential configuration from the offer, then the guidelines provided in Section 3.6.3. for doing a second offer/answer exchange using that potential configuration as the actual configuration apply.

3.7. Interactions with ICE

Interactive Connectivity Establishment (ICE) [ICE] provides a mechanism for verifying connectivity between two endpoints by sending STUN messages directly between the media endpoints. The basic ICE specification [ICE] is defined to support UDP-based connectivity only, however it allows for extensions to support other transport protocols, such as TCP, which is being specified in [ICETCP]. ICE defines a new "a=candidate" attribute, which, among other things, indicates the possible transport protocol(s) to use and then associates a priority with each of them. The most preferred transport protocol that *successfully* verifies connectivity will end up being used.

When using ICE, it is thus possible that the transport protocol that will be used differs from what is specified in the "m=" line. Since both ICE and SDP Capability Negotiation may specify alternative transport protocols, there is a potentially unintended interaction when using these together.

We provide the following guidelines for addressing that.

There are two basic scenarios to consider:

1) A particular media stream can run over different transport protocols (e.g. UDP, TCP, or TCP/TLS), and the intent is simply to use the one that works (in the preference order specified).

2) A particular media stream can run over different transport protocols (e.g. UDP, TCP, or TCP/TLS) and the intent is to have the negotiation process decide which one to use (e.g. T.38 over TCP or UDP).

In scenario 1, there should be ICE "a=candidate" attributes for UDP, TCP, etc. but otherwise nothing special in the potential configuration attributes to indicate the desire to use different transport protocols (e.g. UDP, or TCP). The ICE procedures essentially cover the capability negotiation required (by having the answerer select something it supports and then use of trial and error connectivity checks).

Scenario 2 does not require a need to support or use ICE. Instead, we simply use transport protocol capabilities and potential configuration attributes to indicate the desired outcome.

The scenarios may be combined, e.g. by offering potential configuration alternatives where some of them can support one transport protocol only (e.g. UDP), whereas others can support multiple transport protocols (e.g. UDP or TCP). In that case, there is a need for tight control over the ICE candidates that will be used for a particular configuration, yet the actual configuration may want to use all of the ICE candidates. In that case, the ICE candidate attributes can be defined as attribute capabilities and the relevant ones should then be included in the proper potential configurations (for example candidate attributes for UDP only for potential configurations that are restricted to UDP, whereas there could be candidate attributes for UDP, TCP, and TCP/TLS for potential configurations that can use all three). Furthermore, use of the delete-attributes in a potential configuration can be used to ensure that ICE will not end up using a transport protocol that is not desired for a particular configuration.

3.8. Interactions with SIP Option Tags

SIP [RFC3261] allows for SIP extensions to define a SIP option tag that identifies the SIP extension. Support for one or more such extensions can be indicated by use of the SIP Supported header, and required support for one or more such extensions can be indicated by use of the SIP Require header. The "a=csup" and "a=creq" attributes defined by the SDP Capability Negotiation framework are similar, except that support for these two attributes by themselves cannot be guaranteed (since they are specified as extensions to the SDP specification [RFC4566] itself).

SIP extensions with associated option tags can introduce enhancements to not only SIP, but also SDP. This is for example the case for SIP preconditions defined in [RFC3312]. When using SDP Capability Negotiation, some potential configurations may include certain SDP extensions, whereas others may not. Since the purpose of the SDP Capability Negotiation is to negotiate a session based on the features supported by both sides, use of the SIP Require header for such extensions may not produce the desired result. For example, if one potential configuration requires SIP preconditions support, another does not, and the answerer does not support preconditions, then use of the SIP Require header for preconditions would result in a session failure, in spite of the fact that a valid and supported potential configuration was included in the offer.

In general, this can be alleviated by use of mandatory and optional attribute capabilities in a potential configuration. There are however cases where permissible SDP values are tied to the use of

the SIP Require header. SIP preconditions [RFC3312] is one such example, where preconditions with a "mandatory" strength-tag can only be used when a SIP Require header with the SIP option tag "precondition" is included. Future SIP extensions that may want to use the SDP Capability Negotiation framework should avoid such coupling.

3.9. Processing Media before Answer

The offer/answer model requires an offerer to be able to receive media in accordance with the offer prior to receiving the answer. This property is retained with the SDP Capability Negotiation extensions defined here, but only when the actual configuration is selected by the answerer. If a potential configuration is chosen, it is permissible for the offerer to not process any media received before the answer is received. This may lead to clipping. Consequently, the SDP Capability Negotiation framework recommends sending back an answer SDP as soon as possible.

The issue can be resolved by introducing a three-way handshake. In the case of SIP, this can for example be done by defining a precondition [RFC3312] for capability negotiation (or use an existing precondition that is known to generate a second offer/answer exchange before proceeding with the session). However, preconditions are often viewed as complicated to implement and they may add to overall session establishment delay by requiring an extra offer/answer exchange.

An alternative three-way handshake can be performed by use of ICE [ICE]. When ICE is being used, and the answerer receives a STUN Binding Request for any one of the accepted media streams from the offerer, the answerer knows the offer has received his answer. At that point, the answerer knows that the offerer will be able to process incoming media according to the negotiated configuration and hence he can start sending media without the risk of the offerer either discarding it or playing garbage.

In some use cases a three-way handshake is not needed. An example is when the offerer does not need information from the answer, such as keying material in the SDP, in order to process incoming media. The SDP Capability Negotiation framework does not define any such solutions, however extensions may do so. For example, one technique proposed for best-effort SRTP in [BESRTP] is to provide different RTP payload type mappings for different transport protocols used, outside of the actual configuration, while still allowing them to be used by the answerer (exchange of keying material is still needed,

e.g. inband). The basic SDP Capability Negotiation framework defined here does not include the ability to do so, however extensions that enable that may be defined.

3.10. Indicating Bandwidth Usage

The amount of bandwidth to use for a particular media stream depends on the codecs, transport protocol and other parameters being used. For example use of Secure RTP [RFC3711] with integrity protection requires more bandwidth than plain RTP [RFC3551]. SDP defines the bandwidth ("b=") parameter to indicate the proposed bandwidth for the session or media stream,.

In current SDP, each media description contains one transport protocol and one or more codecs. When specifying the proposed bandwidth, the worst case scenario must be taken into account, i.e., use of the highest bandwidth codec provided, the transport protocol indicated, and the worst case (bandwidth-wise) parameters that can be negotiated (e.g., a 32-bit HMAC or an 80-bit HMAC).

The core SDP capability negotiation framework does not provide a way to negotiate bandwidth parameters. The issue thus remains, however it is potentially worse than with current SDP, since it is easier to negotiate additional codecs, and furthermore possible to negotiate different transport protocols. The recommended approach for addressing this is the same as for plain SDP; the worst case (now including potential configurations) needs to be taken into account when specifying the bandwidth parameters in the actual configuration. This can make the bandwidth value less accurate than in current SDP (due to potential greater variability in the potential configuration bandwidth use). Extensions can be defined to address this shortcoming. Also, the Transport Independent Application Specific Maximum (TIAS) bandwidth type defined in [RFC3890] can be used to alleviate bandwidth variability concerns due to different transport protocols.

Note, that when using RTP retransmission [RFC4588] with the RTCP-based feedback profile [RFC4585] (RTP/AVPF), the retransmitted packets are part of the media stream bandwidth when using SSRC-multiplexing. If a non-feedback based protocol is offered as an alternative transport protocol, it is possible that the bandwidth indication should have been lower.

3.11. Dealing with Large Number of Potential Configurations

When using the SDP Capability Negotiation, it is easy to generate offers that contain a large number of potential configurations. For example, in the offer:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVP 0 18
a=tcap:1 RTP/SAVPF RTP/SAVP RTP/AVPF
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
   FEC_ORDER=FEC_SRTP
a=acap:2 key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
a=acap:3 rtcp-fb:0 nack
a=pcfg:1 t=1 a=1,3|2,3
a=pcfg:2 t=2 a=1|2
a=pcfg:3 t=3 a=3
```

we have 5 potential configurations on top of the actual configuration for a single media stream. Adding an extension capability with just two alternatives for each would double that number (to 10), and doing the equivalent with two media streams would again double that number (to 20). While it is easy (and inexpensive) for the offerer to generate such offers, processing them at the answering side may not be. Consequently, it is RECOMMENDED that offerers do not create offers with unnecessarily large number of potential configurations in them.

On the answering side, implementers MUST take care to avoid excessive memory and CPU consumption. For example, a na implementation that first generates all the valid potential configuration SDPs internally, could find itself being memory exhausted, especially if it supports a large number of endpoints. Similarly, a na implementation that simply performs iterative trial-and-error processing on each possible potential configuration SDP (in the preference order specified) could find itself being CPU constrained. An alternative strategy is to prune the search space first by discarding the set of offered potential configurations where the transport protocol indicated (if any) is not supported, and/or one or more mandatory attribute capabilities (if any) are either not supported or not valid. Potential configurations with

unsupported mandatory extension configurations in them can be discarded as well.

3.12. SDP Capability Negotiation and Intermediaries

An intermediary is here defined as an entity between a SIP user agent A and a SIP user agent B, that need to perform some kind of processing on the SDP exchanged between A and B, in order for the session establishment to operate as intended. Examples of such intermediaries include Session Border Controllers (SBCs) that may perform media relaying, Proxy Call Session Control Functions (P-CSCF) that may authorize use of a certain amount of network resources (bandwidth), etc. The presence and design of such intermediaries may not follow the "Internet" model or the SIP requirements for proxies (which are not supposed to look in message bodies such as SDP), however they are a fact of life in some deployment scenarios currently and hence deserve consideration.

If the intermediary needs to understand the characteristics of the media sessions being negotiated, e.g. the amount of bandwidth used or the transport protocol negotiated, then use of the SDP Capability Negotiation framework may impact them. For example, some intermediaries are known to disallow answers where the transport protocol differs from the one in the offer. Use of the SDP Capability Negotiation framework in the presence of such intermediaries could lead to session failures. Intermediaries that need to authorize use of network resources based on the negotiated media stream parameters are affected as well. If they inspect only the offer, then they may authorize parameters assuming a different transport protocol, codecs, etc. than what is actually being negotiated. For these, and other, reasons it is RECOMMENDED that implementers of intermediaries add support for the SDP Capability Negotiation framework.

The SDP Capability Negotiation framework itself attempts to help out these intermediaries as well, by optionally performing a second offer/answer exchange when use of a potential configuration has been negotiated (see Section 3.6.3.). However, there are several limitations with this approach. First of all, the second offer/answer exchange is not required and hence may not be performed. Secondly, the intermediary may refuse the initial answer, e.g. due to perceived transport protocol mismatch. Thirdly, the strategy is not foolproof, since the offer/answer procedures [RFC3264] leave the original offer/answer exchange in effect when a subsequent one fails; consider the following example:

1. Offerer generates an SDP offer with the actual configuration specifying a low bandwidth configuration (e.g. plain RTP) and a potential configuration specifying a high(er) bandwidth configuration (e.g. secure RTP with integrity).

2. An intermediary (e.g. an SBC or P-CSCF), that does not support SDP Capability Negotiation, authorizes the session based on the actual configuration it sees in the SDP.

3. The answerer chooses the high(er) bandwidth potential configuration and generates an answer SDP based on that.

4. The intermediary passes through the answer SDP.

5. The offerer sees the accepted answer, and generates an updated offer that contains the selected potential configuration as the actual configuration. In other words, the high(er) bandwidth configuration (which has already been negotiated successfully) is now the actual configuration in the offer SDP.

6. The intermediary sees the new offer, however it does not authorize the use of the high(er) bandwidth configuration, and consequently generates a rejection message to the offerer.

7. The offerer receives the rejected offer.

After step 7, per RFC 3264, the offer/answer exchange that completed in step 5 remains in effect, however the intermediary may not have authorized the necessary network resources and hence the media stream may experience quality issues. The solution to this problem is to upgrade the intermediary to support the SDP Capability Negotiation framework.

3.13. Considerations for Specific Attribute Capabilities

3.13.1. The rtpmap and fmtp Attributes

The core SDP Capability Negotiation framework defines transport capabilities and attribute capabilities. Media capabilities, which can be used to describe media formats and their associated parameters, are not defined in this document, however the "rtpmap" and "fmtp" attributes can nevertheless be used as attribute capabilities. Using such attribute capabilities in a potential configuration requires a bit of care though.

The rtpmap parameter binds an RTP payload type to a media format (e.g. codec). While it is possible to provide rtpmaps for payload types not found in the corresponding "m=" line, such rtpmaps provide no value in normal offer/answer exchanges, since only the payload types found in the "m=" line are part of the offer (or answer). This applies to the core SDP Capability Negotiation framework as well: Only the media formats (e.g. RTP payload types) provided in the "m=" line are actually offered; inclusion of rtpmap attributes with other RTP payload types in a potential configuration does not change this fact and hence they do not provide any useful information there. They may still be useful as pure capabilities though (outside a potential configuration) in order to inform a peer of additional codecs supported.

It is possible to provide an rtpmap attribute capability with a payload type mapping to a different codec than a corresponding actual configuration "rtpmap" attribute for the media description has. Such practice is permissible as a way of indicating a capability. If that capability is included in a potential configuration, then delete-attributes (see Section 3.5.1.) MUST be used to ensure that there is not multiple rtpmap attributes for the same payload type in a given media description (which would not be allowed by SDP [RFC4566]).

Similar considerations and rules apply to the "fmtp" attribute. An fmtp attribute capability for a media format not included in the "m=" line is useless in a potential configuration (but may be useful as a capability by itself). An fmtp attribute capability in a potential configuration for a media format that already has an fmtp attribute in the actual configuration may lead to multiple fmtp format parameters for that media format and that is not allowed by SDP [RFC4566]. The delete-attributes MUST be used to ensure that there is not multiple fmtp attributes for a given media format in a media description.

Extensions to the core SDP Capability Negotiation framework may change the above behavior.

3.13.2. Direction Attributes

SDP defines the "inactive", "sendonly", "recvonly", and "sendrecv" direction attributes. The direction attributes can be applied at either the session-level or the media-level. In either case, it is possible to define attribute capabilities for these direction capabilities; if used by a potential configuration, the normal offer/answer procedures still apply. For example, if an offered

potential configuration includes the "sendonly" direction attribute, and it is selected as the actual configuration, then the answer MUST include a corresponding "recvonly" (or "inactive") attribute.

3.14. Relationship to RFC 3407

RFC 3407 defines capability descriptions with limited abilities to describe attributes, bandwidth parameters, transport protocols and media formats. RFC 3407 does not define any negotiation procedures for actually using those capability descriptions.

This document defines new attributes for describing attribute capabilities and transport capabilities. It also defines procedures for using those capabilities as part of an offer/answer exchange. In contrast to RFC 3407, this document does not define bandwidth parameters, and it also does not define how to express ranges of values. Extensions to this document may be defined in order to fully cover all the capabilities provided by RFC 3407 (for example more general media capabilities).

It is RECOMMENDED that implementations use the attributes and procedures defined in this document instead of those defined in [RFC3407]. If capability description interoperability with legacy RFC 3407 implementations is desired, implementations MAY include both RFC 3407 capability descriptions and capabilities defined by this document. The offer/answer negotiation procedures defined in this document will not use the RFC 3407 capability descriptions.

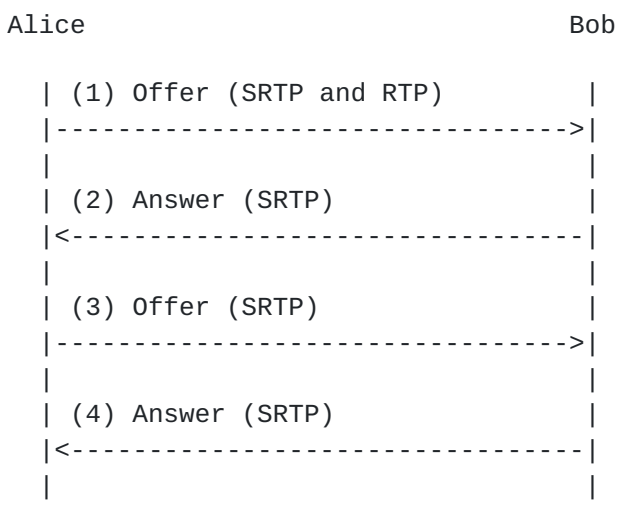
4. Examples

In this section, we provide examples showing how to use the SDP Capability Negotiation.

4.1. Best-Effort Secure RTP

The following example illustrates how to use the SDP Capability Negotiation extensions to support so-called Best-Effort Secure RTP. In that scenario, the offerer supports both RTP and Secure RTP. If the answerer does not support secure RTP (or the SDP Capability Negotiation extensions), an RTP session will be established. However, if the answerer supports Secure RTP and the SDP Capability Negotiation extensions, a Secure RTP session will be established.

The best-effort Secure RTP negotiation is illustrated by the offer/answer exchange below, where Alice sends an offer to Bob:



Alice's offer includes RTP and SRTP as alternatives. RTP is the default, but SRTP is the preferred one:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVP 0 18
a=tcap:1 RTP/SAVP RTP/AVP
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
   FEC_ORDER=FEC_SRTP
a=pcfg:1 t=1 a=1
```

The "m=" line indicates that Alice is offering to use plain RTP with PCMU or G.729. The capabilities are provided by the "a=tcap" and "a=acap" attributes. The "tcap" capability indicates that both Secure RTP and normal RTP are supported. The "acap" attribute provides an attribute capability with a handle of 1. The capability is a "crypto" attribute, which provides the keying material for SRTP using SDP security descriptions [RFC4568]. The "a=pcfg" attribute provides the potential configurations included in the offer by reference to the capabilities. A single potential configuration with a configuration number of "1" is provided. It includes the transport protocol capability 1 (RTP/SAVP, i.e. secure RTP) together with the attribute capability 1, i.e. the crypto attribute provided. Note that attribute capability 1 is mandatory, and hence it must be supported in order for the potential configuration to be used.

Bob receives the SDP offer from Alice. Bob supports SRTP and the SDP Capability Negotiation framework, and hence he accepts the potential configuration for Secure RTP provided by Alice:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/SAVP 0 18
a=crypto:1 AES_CM_128_HMAC_SHA1_80
      inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
a=acfg:1 t=1 a=1
```

Bob includes the "a=acfg" attribute in the answer to inform Alice that he based his answer on an offer containing the potential configuration with transport protocol capability 1 and attribute capability 1 from the offer SDP (i.e. the RTP/SAVP profile using the keying material provided). Bob also includes his keying material in a crypto attribute.

When Alice receives Bob's answer, session negotiation has completed, however Alice nevertheless chooses to generate a new offer using the actual configuration. This is done purely to assist any intermediaries that may reside between Alice and Bob but do not support the SDP Capability Negotiation framework (and hence may not understand the negotiation that just took place):

Alice's updated offer includes only SRTP, and it is not using the SDP Capability Negotiation framework (Alice could have included the capabilities as well is she wanted to):

```
v=0
o=- 25678 753850 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/SAVP 0 18
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
   FEC_ORDER=FEC_SRTP
```

The "m=" line now indicates that Alice is offering to use secure RTP with PCMU or G.729. The "crypto" attribute, which provides the SRTP keying material, is included with the same value again.

Bob receives the SDP offer from Alice, which he accepts, and then generates an answer to Alice:

```
v=0
o=- 24351 621815 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/SAVP 0 18
a=crypto:1 AES_CM_128_HMAC_SHA1_80
      inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

Bob includes the same crypto attribute as before, and the session proceeds without change. Although Bob did not include any capabilities in his answer, he could have done so if he wanted to.

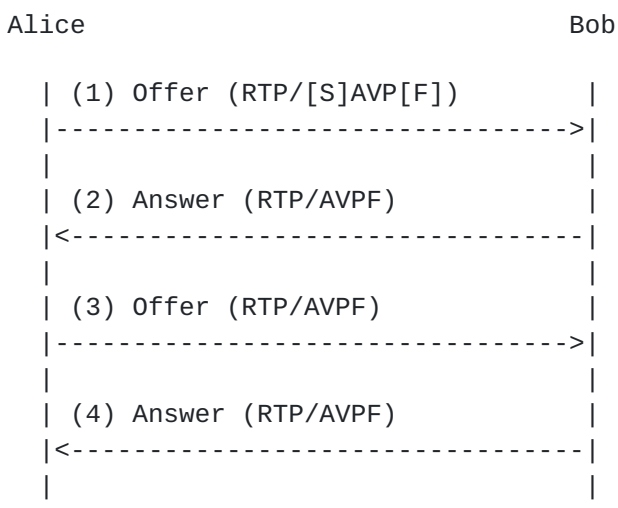
Note that in this particular example, the answerer supported the SDP Capability Negotiation framework, and hence the attributes and procedures defined here, however had he not, the answerer would simply have ignored the new attributes received in step 1 and accepted the offer to use normal RTP. In that case, the following answer would have been generated in step 2 instead:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/AVP 0 18
```

4.2. Multiple Transport Protocols

The following example illustrates how to use the SDP Capability Negotiation extensions to negotiate use of one out of several possible transport protocols. As in the previous example, the offerer uses the expected least-common-denominator (plain RTP) as the actual configuration, and the alternative transport protocols as the potential configurations.

The example is illustrated by the offer/answer exchange below, where Alice sends an offer to Bob:



Alice's offer includes plain RTP (RTP/AVP), RTP with RTCP-based feedback (RTP/AVPF), Secure RTP (RTP/SAVP), and Secure RTP with RTCP-based feedback (RTP/SAVPF) and SRTP as alternatives. RTP is the default, with RTP/SAVPF, RTP/SAVP, and RTP/AVPF as the alternatives and preferred in the order listed:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVP 0 18
a=tcap:1 RTP/SAVPF RTP/SAVP RTP/AVPF
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
   FEC_ORDER=FEC_SRTP
a=acap:2 rtcp-fb:0 nack
a=pcfg:1 t=1 a=1,[2]
a=pcfg:2 t=2 a=1
a=pcfg:3 t=3 a=[2]
```

The "m=" line indicates that Alice is offering to use plain RTP with PCMU or G.729. The capabilities are provided by the "a=tcap" and "a=acap" attributes. The "tcap" capability indicates that Secure RTP with RTCP-Based feedback (RTP/SAVPF), Secure RTP (RTP/SAVP), and RTP with RTCP-Based feedback are supported. The first "acap" attribute provides an attribute capability with a handle of 1. The capability is a "crypto" attribute, which provides the keying material for SRTP using SDP security descriptions [RFC4568]. The second "acap" attribute provides an attribute capability with a

handle of 2. The capability is an "rtcp-fb" attribute, which is used by the RTCP-based feedback profiles to indicate that payload type 0 (PCMU) supports feedback type "nack". The "a=pcfg" attributes provide the potential configurations included in the offer by reference to the capabilities. There are three potential configurations:

- Potential configuration 1, which is the most preferred potential configuration specifies use of transport protocol capability 1 (RTP/SAVPF) and attribute capabilities 1 (the "crypto" attribute) and 2 (the "rtcp-fb" attribute). Support for the first one is mandatory whereas support for the second one is optional.

- Potential configuration 2, which is the second most preferred potential configuration specifies use of transport protocol capability 2 (RTP/SAVP) and mandatory attribute capability 1 (the "crypto" attribute).

- Potential configuration 3, which is the least preferred potential configuration (but the second least preferred configuration overall, since the actual configuration provided by the "m=" line is always the least preferred configuration), specifies use of transport protocol capability 3 (RTP/AVPF) and optional attribute capability 2 (the "rtcp-fb" attribute).

Bob receives the SDP offer from Alice. Bob does not support any secure RTP profiles, however he supports plain RTP and RTP with RTCP-based feedback, as well as the SDP Capability Negotiation extensions, and hence he accepts the potential configuration for RTP with RTCP-based feedback provided by Alice:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/AVPF 0 18
a=rtcp-fb:0 nack
a=acfg:1 t=3 a=[2]
```

Bob includes the "a=acfg" attribute in the answer to inform Alice that he based his answer on an offer containing the potential configuration with transport protocol capability 3 and optional attribute capability 2 from the offer SDP (i.e. the RTP/AVPF profile using the "rtcp-fb" value provided). Bob also includes an "rtcp-fb" attribute with the value "nack" value for RTP payload type 0.

When Alice receives Bob's answer, session negotiation has completed, however Alice nevertheless chooses to generate a new offer using the actual configuration. This is done purely to assist any intermediaries that may reside between Alice and Bob but do not support the SDP Capability Negotiation framework (and hence may not understand the negotiation that just took place):

Alice's updated offer includes only RTP/AVPF, and it is not using the SDP Capability Negotiation framework (Alice could have included the capabilities as well if she wanted to):

```
v=0
o=- 25678 753850 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=audio 53456 RTP/AVPF 0 18
a=rtcp-fb:0 nack
```

The "m=" line now indicates that Alice is offering to use RTP with RTCP-based feedback and using PCMU or G.729. The "rtcp-fb" attribute provides the feedback type "nack" for payload type 0 again (but as part of the actual configuration).

Bob receives the SDP offer from Alice, which he accepts, and then generates an answer to Alice:

```
v=0
o=- 24351 621815 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/AVPF 0 18
a=rtcp-fb:0 nack
```

Bob includes the same "rtcp-fb" attribute as before, and the session proceeds without change. Although Bob did not include any capabilities in his answer, he could have done so if he wanted to.

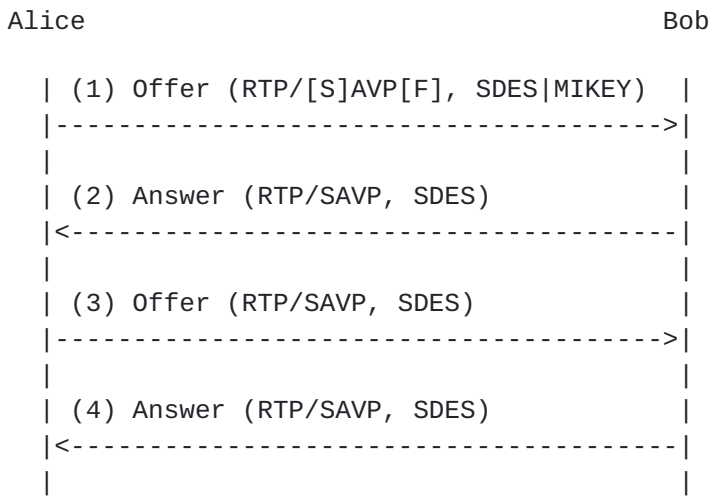
Note that in this particular example, the answerer supported the SDP Capability Negotiation framework and hence the attributes and procedures defined here, however had he not, the answerer would simply have ignored the new attributes received in step 1 and accepted the offer to use normal RTP. In that case, the following answer would have been generated in step 2 instead:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=audio 54568 RTP/AVP 0 18
```

4.3. Best-Effort SRTP with Session-Level MIKEY and Media Level Security Descriptions

The following example illustrates how to use the SDP Capability Negotiation extensions to support so-called Best-Effort Secure RTP as well as alternative keying mechanisms, more specifically MIKEY [RFC3830] and SDP Security Descriptions. The offerer (Alice) wants to establish an audio and video session. Alice prefers to use session-level MIKEY as the key management protocol, but supports SDP security descriptions as well.

The example is illustrated by the offer/answer exchange below, where Alice sends an offer to Bob:



Alice's offer includes an audio and a video stream. The audio stream offers use of plain RTP and secure RTP as alternatives, whereas the video stream offers use of plain RTP, RTP with RTCP-based feedback, Secure RTP, and Secure RTP with RTCP-based feedback as alternatives:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
t=0 0
```

```
c=IN IP4 192.0.2.1
a=acap:1 key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
a=tcap:1 RTP/SAVPF RTP/SAVP RTP/AVPF
m=audio 59000 RTP/AVP 98
a=rtpmap:98 AMR/8000
a=acap:2 crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=pcfg:1 t=2 a=1|2
m=video 52000 RTP/AVP 31
a=rtpmap:31 H261/90000
a=acap:3 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=acap:4 rtcp-fb:* nack
a=pcfg:1 t=1 a=1,4|3,4
a=pcfg:2 t=2 a=1|3
a=pcfg:3 t=3 a=4
```

The potential configuration for the audio stream specifies use of transport capability 2 (RTP/SAVP) and either attribute capability 1 (session-level MIKEY as the keying mechanism) or 2 (SDP Security Descriptions as the keying mechanism). Support for either of these attribute capabilities is mandatory. There are three potential configurations for the video stream.

- The first configuration with configuration number 1 uses transport capability 1 (RTP/SAVPF) with either attribute capabilities 1 and 4 (session-level MIKEY and the "rtcp-fb" attribute) or attribute capabilities 3 and 4 (SDP security descriptions and the "rtcp-fb" attribute). In this example, the offerer insists on not only the keying mechanism being supported, but also that the "rtcp-fb" attribute is supported with the value indicated. Consequently, all the attribute capabilities are marked as mandatory in this potential configuration.

- The second configuration with configuration number 2 uses transport capability 2 (RTP/SAVP) and either attribute capability 1 (session-level MIKEY) or attribute capability 3 (SDP security descriptions). Both attribute capabilities are mandatory in this configuration.

- The third configuration with configuration number 3 uses transport capability 3 (RTP/AVPF) and mandatory attribute capability 4 (the "rtcp-fb" attribute).

Bob receives the SDP offer from Alice. Bob supports Secure RTP, Secure RTP with RTCP-based feedback and the SDP Capability

Negotiation extensions. Bob also supports SDP Security Descriptions, but not MIKEY, and hence he generates the following answer:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
t=0 0
c=IN IP4 192.0.2.2
m=audio 54568 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:WSJ+PSdFcGdUJShpX1ZjNzB4d1BINUAvLEw6UzF3|2^20|1:32
a=acfg:1 t=2 a=2
m=video 55468 RTP/SAVPF 31
a=rtpmap:31 H261/90000
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:AwWpVLFJhQX1cfHJSojd0RmdmcmVCspeEc3QGZiN|2^20|1:32
a=rtcp-fb:* nack
a=acfg:1 t=1 a=3,4
```

For the audio stream, Bob accepted the use of secure RTP, and hence the profile in the "m=" line is "RTP/SAVP". Bob also includes a "crypto" attribute with his own keying material, and an "acfg" attribute identifying actual configuration 1 for the audio media stream from the offer, using transport capability 2 (RTP/SAVP) and attribute capability 2 (the crypto attribute from the offer). For the video stream, Bob accepted the use of secure RTP with RTCP-based feedback, and hence the profile in the "m=" line is "RTP/SAVPF". Bob also includes a "crypto" attribute with his own keying material, and an "acfg" attribute identifying actual configuration 1 for the video stream from the offer, using transport capability 1 (RTP/SAVPF) and attribute capabilities 3 (the crypto attribute from the offer) and 4 (the "rtcp-fb" attribute from the offer).

When Alice receives Bob's answer, session negotiation has completed, however Alice nevertheless chooses to generate a new offer using the actual configuration. This is done purely to assist any intermediaries that may reside between Alice and Bob but do not support the capability negotiation extensions (and hence may not understand the negotiation that just took place):

Alice's updated offer includes only SRTP for the audio stream SRTP with RTCP-based feedback for the video stream, and it is not using the SDP Capability Negotiation framework (Alice could have included the capabilities as well is she wanted to):

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
t=0 0
c=IN IP4 192.0.2.1
m=audio 59000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
m=video 52000 RTP/SAVPF 31
a=rtpmap:31 H261/90000
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=rtcp-fb:* nack
```

The "m=" line for the audio stream now indicates that Alice is offering to use secure RTP with PCMU or G.729, whereas the "m=" line for the video stream indicates that Alice is offering to use secure RTP with RTCP-based feedback and H.261. Each media stream includes a "crypto" attribute, which provides the SRTP keying material, with the same value again.

Bob receives the SDP offer from Alice, which he accepts, and then generates an answer to Alice:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
t=0 0
c=IN IP4 192.0.2.2
m=audio 54568 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:WSJ+PSdFcGdUJShpX1ZjNzB4d1BINUAvLEw6UzF3|2^20|1:32
m=video 55468 RTP/SAVPF 31
a=rtpmap:31 H261/90000
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:AwWpVLFJhQX1cfHJSojd0RmdmcmVCspeEc3QGZiN|2^20|1:32
a=rtcp-fb:* nack
```

Bob includes the same crypto attribute as before, and the session proceeds without change. Although Bob did not include any capabilities in his answer, he could have done so if he wanted to.

Note that in this particular example, the answerer supported the capability extensions defined here, however had he not, the answerer

would simply have ignored the new attributes received in step 1 and accepted the offer to use normal RTP. In that case, the following answer would have been generated in step 2 instead:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
t=0 0
c=IN IP4 192.0.2.2
m=audio 54568 RTP/AVP 98
a=rtpmap:98 AMR/8000
m=video 55468 RTP/AVP 31
a=rtpmap:31 H261/90000
a=rtcp-fb:* nack
```

Finally, if Bob had chosen to use session-level MIKEY instead of SDP security descriptions instead, the following answer would have been generated:

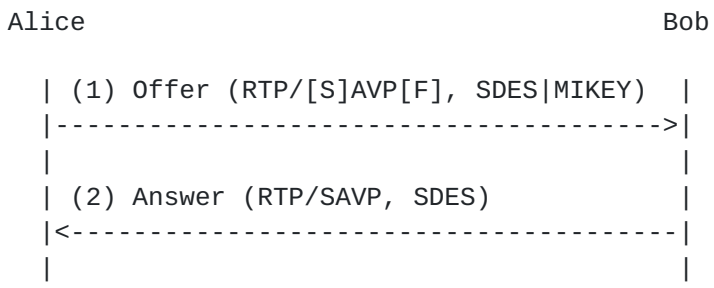
```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
t=0 0
c=IN IP4 192.0.2.1
a=key-mgmt:mikey AQEFgM0XflABAAAAAAAAAAAAAAYAyO...
m=audio 59000 RTP/AVP 98
a=rtpmap:98 AMR/8000
a=acfg:1 t=2 a=1
m=video 52000 RTP/SAVPF 31
a=rtpmap:31 H261/90000
a=rtcp-fb:* nack
a=acfg:1 t=1 a=1,4
```

It should be noted, that although Bob could have chosen session-level MIKEY for one media stream, and SDP Security Descriptions for another media stream, there are no well-defined offerer processing rules of the resulting answer for this, and hence the offerer may incorrectly assume use of MIKEY for both streams. To avoid this, if the answerer chooses session-level MIKEY, then all secure RTP based media streams SHOULD use MIKEY (this applies irrespective of whether SDP Capability Negotiation is being used or not). Use of media-level MIKEY does not have a similar constraint.

4.4. SRTP with Session-Level MIKEY and Media Level Security Descriptions as Alternatives

The following example illustrates how to use the SDP Capability Negotiation framework to negotiate use of either MIKEY or SDP Security Descriptions, when one of them is included as part of the actual configuration, and the other one is being selected. The offerer (Alice) wants to establish an audio and video session. Alice prefers to use session-level MIKEY as the key management protocol, but supports SDP security descriptions as well.

The example is illustrated by the offer/answer exchange below, where Alice sends an offer to Bob:



Alice's offer includes an audio and a video stream. Both the audio and the video stream offer use of secure RTP:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
t=0 0
c=IN IP4 192.0.2.1
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
m=audio 59000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=pcfg:1 a=-s:1
m=video 52000 RTP/SAVP 31
a=rtpmap:31 H261/90000
a=acap:2 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=pcfg:1 a=-s:2
```

Alice does not know whether Bob supports MIKEY or SDP Security Descriptions. She could include attributes for both, however the

resulting procedures and potential interactions are not well-defined. Instead, she places a session-level key-mgmt attribute for MIKEY in the actual configuration with SDP security descriptions as an alternative in the potential configuration. The potential configuration for the audio stream specifies that all session level attributes are to be deleted (i.e. the session-level "a=key-mgmt" attribute) and that mandatory attribute capability 2 is to be used (i.e. the crypto attribute). The potential configuration for the video stream is similar, except it uses it's own mandatory crypto attribute capability (2). Note how deletion of the session-level attributes does not affect the media-level attributes.

Bob receives the SDP offer from Alice. Bob supports Secure RTP and the SDP Capability Negotiation framework. Bob also supports both SDP Security Descriptions and MIKEY. Since the potential configuration is more preferred than the actual configuration, Bob (conceptually) generates an internal potential configuration SDP that contains the crypto attributes for the audio and video stream, but not the key-mgmt attribute for MIKEY, thereby avoiding any ambiguity between the two keying mechanisms. As a result, he generates the following answer:

```
v=0
o=- 24351 621814 IN IP4 192.0.2.2
s=
t=0 0
c=IN IP4 192.0.2.2
m=audio 54568 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:WSJ+PSdFcGdUJShpX1ZjNzB4d1BINUAvLEw6UzF3|2^20|1:32
a=acfg:1 a=-s:1
m=video 55468 RTP/SAVP 31
a=rtpmap:31 H261/90000
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:AwWpVLFJhQX1cfHJSojd0RmdmcmVCspeEc3QGZiN|2^20|1:32
a=acfg:1 a=-s:2
```

For the audio stream, Bob accepted the use of secure RTP using SDP security descriptions. Bob therefore includes a "crypto" attribute with his own keying material, and an "acfg" attribute identifying actual configuration 1 for the audio media stream from the offer, with the delete-attributes ("-s") and attribute capability 1 (the crypto attribute from the offer). For the video stream, Bob also accepted the use of secure RTP using SDP security descriptions. Bob therefore includes a "crypto" attribute with his own keying

material, and an "acfg" attribute identifying actual configuration 1 for the video stream from the offer, with the delete-attributes ("-s") and attribute capability 2.

Below, we illustrate the offer SDP, when Bob instead offers the "crypto" attribute as the actual configuration keying mechanism and "key-mgmt" as the potential configuration:

```
v=0
o=- 25678 753849 IN IP4 192.0.2.1
s=
t=0 0
c=IN IP4 192.0.2.1
a=acap:1 key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
m=audio 59000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=crypto:1 AES_CM_128_HMAC_SHA1_32
   inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=acap:2 rtpmap:98 AMR/8000
a=pcfg:1 a=-m:1,2
m=video 52000 RTP/SAVP 31
a=rtpmap:31 H261/90000
a=acap:3 crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=acap:4 rtpmap:31 H261/90000
a=pcfg:1 a=-m:1,4
```

Note how we this time need to perform delete-attributes at the media-level instead of the session-level. When doing that, all attributes from the actual configuration SDP, including the rtpmaps provided, are removed. Consequently, we had to include these rtpmaps as capabilities as well, and then include them in the potential configuration, thereby effectively recreating the original rtpmap attributes in the resulting potential configuration SDP.

5. Security Considerations

The SDP Capability Negotiation Framework is defined to be used within the context of the offer/answer model, and hence all the offer/answer security considerations apply here as well. Similarly, the Session Initiation Protocol (SIP) uses SDP and the offer/answer model, and hence, when used in that context, the SIP security considerations apply as well.

However, SDP Capability Negotiation introduces additional security issues. Its use as a mechanism to enable alternative transport

protocol negotiation (secure and non-secure) as well as its ability to negotiate use of more or less secure keying methods and material warrant further security considerations. Also, the (continued) support for receiving media before answer combined with negotiation of alternative transport protocols (secure and non-secure) warrant further security considerations. We discuss these issues below.

The SDP Capability Negotiation framework allows for an offered media stream to both indicate and support various levels of security for that media stream. Different levels of security can for example be negotiated by use of alternative attribute capabilities each indicating more or less secure keying methods as well as more or less strong ciphers. Since the offerer indicates support for each of these alternatives, he will presumably accept the answerer seemingly selecting any of the offered alternatives. If an attacker can modify the SDP offer, he can thereby force the negotiation of the weakest security mechanism that the offerer is willing to accept. This may enable the attacker to compromise the security of the negotiated media stream. Similarly, if the offerer wishes to negotiate use of a secure media stream (e.g. secure RTP), but includes a non-secure media stream (e.g. plain RTP) as a valid (but less preferred) alternative, then an attacker that can modify the offered SDP will be able to force the establishment of an insecure media stream. The solution to both of these problems involves the use of integrity protection over the SDP. Ideally, this integrity protection provides end-to-end integrity protection in order to protect from any man-in-the-middle attack; secure multiparts such as S/MIME [RFC3851] provide one such solution, however S/MIME requires use and availability of a Public Key Infrastructure (PKI). A slightly less secure alternative when using SIP, but generally much easier to deploy in practice (since it does not require a PKI), is to use SIP Identity [RFC4474]; this requires the existence of an authentication service (see [RFC4474]). Yet another, and considerably less secure, alternative is to use hop-by-hop security only, e.g. TLS or IPSec thereby ensuring the integrity of the offered SDP on a hop-by-hop basis. Note however that SIP proxies or other intermediaries processing the SIP request at each hop are able to perform a man-in-the-middle attack by modifying the offered SDP.

Per the normal offer/answer procedures, as soon as the offerer has generated an offer, the offerer must be prepared to receive media in accordance with that offer. The SDP Capability Negotiation preserves that behavior for the actual configuration in the offer, however the offerer has no way of knowing which configuration (actual or potential) configuration was selected by the offerer, until an answer indication is received. This opens up a new security issue

where an attacker may be able to interject media towards the offerer until the answer is received. For example, the offerer may use plain RTP as the actual configuration and secure RTP as an alternative potential configuration. Even though the answerer selects secure RTP, the offerer will not know that until he receives the answer, and hence an attacker will be able to send media to the offerer meanwhile. The easiest protection against such an attack is to not offer use of the non-secure media stream in the actual configuration, however that may in itself have undesirable side-effects: If the answerer does not support the secure media stream and also does not support the capability negotiation framework, then negotiation of the media stream will fail. Alternatively, SDP security preconditions [RFC5027] can be used. This will ensure that media is not flowing until session negotiation has completed and hence the selected configuration is known. Use of preconditions however requires both sides to support them. If they don't, and use of them is required, the session will fail. As a (limited) work around to this, it is RECOMMENDED that SIP entities generate an answer SDP and send it to the offerer as soon as possible, for example in a 183 Session Progress message. This will limit the time during which an attacker can send media to the offerer. Section 3.9. presents other alternatives as well.

Additional security considerations apply to the answer SDP as well. The actual configuration attribute tells the offerer which potential configuration the answer was based on, and hence an attacker that can either modify or remove the actual configuration attribute in the answer can cause session failure as well as extend the time window during which the offerer will accept incoming media that does not conform to the actual answer. The solutions to this SDP answer integrity problem are the same as for the offer, i.e. use of end-to-end integrity protection, SIP identity, or hop-by-hop protection. The mechanism to use depends on the mechanisms supported by the offerer as well as the acceptable security trade-offs.

As described in Section 3.1. , SDP Capability Negotiation conceptually allows an offerer to include many different offers in a single SDP. This can cause the answerer to process a large number of alternative potential offers, which can consume significant memory and CPU resources. An attacker can use this amplification feature to launch a denial of service attack against the answerer. The answerer MUST protect itself from such attacks. As explained in Section 3.10. , the answerer can help reduce the effects of such an attack by first discarding all potential configurations that contain unsupported transport protocols, unsupported or invalid mandatory attribute capabilities, or unsupported mandatory extension

configurations. The answerer SHOULD also look out for potential configurations that are designed to pass the above test, but nevertheless produce a large number of potential configuration SDPs that cannot be supported.

A possible way of achieving that is for an attacker to find a valid session-level attribute that causes conflicts or otherwise interferes with individual media description configurations. Currently, we do not know of such an SDP attribute, however this does not mean it does not exist, or that it will not exist in the future. If such attributes are found to exist, implementers should explicitly protect against them.

A significant number of valid and supported potential configurations may remain. However, since all of those contain only valid and supported transport protocols and attributes, it is expected that only a few of them will need to be processed on average. Still, the answerer MUST ensure that it does not needlessly consume large amounts of memory or CPU resources when processing those as well as be prepared to handle the case where a large number of potential configurations still need to be processed.

6. IANA Considerations

6.1. New SDP Attributes

The IANA is hereby requested to register the following new SDP attributes as follows:

Attribute name:	csup
Long form name:	Supported capability negotiation extensions
Type of attribute:	Session-level and media-level
Subject to charset:	No
Purpose:	Option tags for supported SDP capability negotiation extensions
Appropriate values:	See Section 3.3.1.

Attribute name:	creq
Long form name:	Required capability negotiation extensions
Type of attribute:	Session-level and media-level
Subject to charset:	No
Purpose:	Option tags for required SDP capability negotiation extensions
Appropriate values:	See Section 3.3.2.

Attribute name:	acap
Long form name:	Attribute capability
Type of attribute:	Session-level and media-level
Subject to charset:	No
Purpose:	Attribute capability containing an attribute name and associated value
Appropriate values:	See Section 3.4.1.

Attribute name:	tcap
Long form name:	Transport Protocol Capability
Type of attribute:	Session-level and media-level
Subject to charset:	No
Purpose:	Transport protocol capability listing one or more transport protocols
Appropriate values:	See Section 3.4.2.

Attribute name:	pcfg
Long form name:	Potential Configuration
Type of attribute:	Media-level
Subject to charset:	No
Purpose:	Potential configuration for SDP capability negotiation
Appropriate values:	See Section 3.5.1.

Attribute name:	acfg
Long form name:	Actual configuration
Type of attribute:	Media-level
Subject to charset:	No
Purpose:	Actual configuration for SDP capability negotiation
Appropriate values:	See Section 3.5.2.

6.2. New SDP Capability Negotiation Option Tag Registry

The IANA is hereby requested to create a new SDP Capability Negotiation Option Tag registry. An IANA SDP Capability Negotiation option tag registration MUST be documented in an RFC in accordance with the [RFC2434] Specification Required policy. The RFC MUST provide the name of the option tag, a syntax and a semantic specification of any new SDP attributes and any extensions to the potential and actual configuration attributes provided in this document. New SDP attributes that are intended to be capabilities for use by the capability negotiation framework MUST adhere to the guidelines provided in Section 3.4.3. Extensions to the potential and actual configuration attributes MUST adhere to the syntax provided in Section 3.5.1. and 3.5.2.

The option tag "cap-v0" is defined in this document and the IANA is hereby requested to register this option tag.

6.3. New SDP Capability Negotiation Potential Configuration Parameter Registry

The IANA is hereby requested to create a new SDP Capability Negotiation Potential Configuration Parameter registry. An IANA SDP Capability Negotiation potential configuration registration MUST be documented in an RFC in accordance with the [RFC2434] Specification Required policy. The RFC MUST define the syntax and semantics of each new potential configuration parameter. The syntax MUST adhere to the syntax provided for extensions in Section 3.5.1. and the semantics MUST adhere to the semantics provided for extensions in Section 3.5.1. and 3.5.2. Associated with each registration MUST be the encoding name for the parameter as well as a short descriptive name for it.

The potential configuration parameters "a" for "attribute" and "t" for "transport protocol" are defined in this document and the IANA is hereby requested to register these.

7. Acknowledgments

This document is heavily influenced by the discussions and work done by the SDP Capability Negotiation Design team. The following people in particular provided useful comments and suggestions to either the document itself or the overall direction of the solution defined in here: Francois Audet, John Elwell, Roni Even, Robert Gilman, Cullen Jennings, Jonathan Lennox, Matt Lepinski, Joerg Ott, Colin Perkins, Jonathan Rosenberg, Thomas Stach, and Dan Wing.

8. Change Log

8.1. draft-ietf-mmusic-sdp-capability-negotiation-07

- Removed the ability to have attribute capabilities provide attribute names without values, when those attributes otherwise require an associated value.

- Document no longer obsoletes RFC 3407 but instead recommends that it is being used instead of RFC 3407.

- Added ability to specific that specific extensions in a potential configuration are mandatory.

- Changed ABNF for extension-config-list in potential configurations.
- Removed the redundant "a=" part of attribute capabilities.
- Clarified what it means to support an attribute capability in the offer/answer procedures.
- Changed "a=acap" attribute and offer/answer procedures to include only the known and supported attribute capabilities.
- Added new section on indicating bandwidth usage.

8.2. draft-ietf-mmusic-sdp-capability-negotiation-06

- Added additional background text on terminology used, and a new section on the negotiation model.
- Allowed for session-level attribute capabilities to contain media-level only attributes, albeit the base framework does not define (or allow) them to be used in a potential configuration (extensions may change that)
- Disallowing multiple "a=tcap" attributes at the session-level and/or on a per media description basis; at most one at the session-level and per media description now.
- Changed the "a=pcfg" attribute to make a potential configuration list optional in order to allow for the actual configuration to be referenced.
- Removed the ability to delete and replace individual attributes from the actual configuration SDP.
- Introduced the notion of mandatory and optional attribute capabilities in a potential configuration and updated the "a=pcfg" attribute and associated procedures accordingly.
- Specified that mandatory attribute capabilities and the transport protocol (if any) from a potential configuration need to be supported in order to select that potential configuration. Offer/answer procedures updated accordingly as well.

- Noted potential interaction and synchronization issues with use of session-level attributes and attribute capabilities and added recommendation to avoid use of session-level attributes when possible.

- Fixed error in "a=acfg" grammar (missing config-number) and updated attribute definition in accordance with the "a=pcfg" attribute changes.

- Updated text associated with processing media before answer to allow for playing out garbage or discard until answer received. Additional detail on alternative solutions provided as well.

- Added recommendation to send back answer SDP as soon as possible, when a potential configuration different from the actual configuration has been chosen.

- Added new section on interactions with SIP option tags.

- Added new section on dealing with large number of potential configurations.

- Added new section on SDP capability negotiation and intermediaries.

- Updated examples in accordance with other changes and to illustrate use of mandatory and optional attribute capabilities in a potential configuration.

- Updated security considerations to address potential denial of service attack caused by large number of potential configurations.

- Various editorial updates throughout.

8.3. draft-ietf-mmusic-sdp-capability-negotiation-05

- Allowed for '<type>=<value>' attributes to be listed as attribute capabilities the attribute name only.

- Changed IP-address to conform to RFC 3330 guidelines.

- Added section on relationship to RFC 3407 and "Obsoletes: 3407" in the front.

- Disallowed use of white space in a number of places for more consistency with existing SDP practice

- Changed "csup" and "creq" attributes to not allow multiple instances at the session-level and multiple instances per media description (only one for each now)

- Changed to not require use of "creq" with base option tag ("cap-v0").

- Relaxed restrictions on extension capabilities

- Updated potential configuration attribute syntax and semantics. In particular, potential configuration attributes can now replace and delete various existing attributes in original SDP to better control potential attribute interactions with the actual configuration while preserving message size efficiency.

- Updated actual configuration attribute to align with the updates to the potential configuration attributes.

- Updated offer/answer procedures to align with other changes.

- Changed recommendation for second offer/answer exchange to "MAY" strength, unless for the cases where it is known or suspected that it is needed.

- Updated ICE interactions to explain how the new attribute delete/replace features can solve certain potential interactions.

- Updated rtpmap and fmtp section to allow potential configurations to use remapped payload types in attribute capabilities for rtpmaps and fmtp parameters.

- Added section on direction attributes.

- Added another example showing SRTP with session-level MIKEY and SDP Security Descriptions using the attribute capability DELETE operator.

8.4. draft-ietf-mmusic-sdp-capability-negotiation-04

The following are the major changes compared to version -03:

- Added explicit ordering rules for attributes added by potential configurations.

- Noted that ICE interaction issues (ice-tcp specifically) may not be as clear as originally thought.
- Added considerations on using rtpmap and fmtp attributes as attribute capabilities.
- Added multiple transport protocol example.
- Added session-level MIKEY and media level security descriptions example.

8.5. draft-ietf-mmusic-sdp-capability-negotiation-03

The following are the major changes compared to version -02:

- Base option tag name changed from "v0" to "cap-v0".
- Added new section on extension capability attributes
- Firmed up offer/answer procedures.
- Added security considerations
- Added IANA considerations

8.6. draft-ietf-mmusic-sdp-capability-negotiation-02

The following are the major changes compared to version -01:

- Potential configurations are no longer allowed at the session level
- Renamed capability attributes ("capar" to "acap" and "ctrpr" to "tcap")
- Changed name and semantics of the initial number (now called configuration number) in potential configuration attributes; must now be unique and can be used as a handle
- Actual configuration attribute now includes configuration number from the selected potential configuration attribute
- Added ABNF throughout
- Specified that answerer should include "a=csup" in case of unsupported required extensions in offer.

- Specified use of second offer/answer exchange when answerer selected a potential configuration
- Updated rules (and added restrictions) for referencing media- and session-level capabilities in potential configurations (at the media level)
- Added initial section on ICE interactions
- Added initial section on receiving media before answer

8.7. draft-ietf-mmusic-sdp-capability-negotiation-01

The following are the major changes compared to version -00:

- Media capabilities are no longer considered a core capability and hence have been removed. This leaves transport protocols and attributes as the only capabilities defined by the core.
- Version attribute has been removed and an option tag to indicate the actual version has been defined instead.
- Clarified rules for session-level and media level attributes provided at either level as well how they can be used in potential configurations.
- Potential configuration parameters no longer have implicit ordering; an explicit preference indicator is now included.
- The parameter name for transport protocols in the potential and actual configuration attributes have been changed "p" to "t".
- Clarified operator precedence within potential and actual configuration attributes.
- Potential configurations at the session level now limited to indicate latent capability configurations. Consequently, an actual configuration attribute can no longer be provided at the session level.
- Cleaned up capability and potential configuration terminology - they are now two clearly different things.

8.8. draft-ietf-mmusic-sdp-capability-negotiation-00

Version 00 is the initial version. The solution provided in this initial version is based on an earlier (individual submission) version of [SDPCapNeg]. The following are the major changes compared to that document:

- Solution no longer based on RFC 3407, but defines a set of similar attributes (with some differences).
- Various minor changes to the previously defined attributes.
- Multiple transport capabilities can be included in a single "tcap" attribute
- A version attribute is now included.
- Extensions to the framework are formally supported.
- Option tags and the ability to list supported and required extensions are supported.
- A best-effort SRTP example use case has been added.
- Some terminology change throughout to more clearly indicate what constitutes capabilities and what constitutes configurations.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC3264] Rosenberg, J., and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3407] F. Andreasen, "Session Description Protocol (SDP) Simple Capability Declaration", RFC 3407, October 2002.

[RFC4234] Crocker, D., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

9.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3312] G. Camarillo, W. Marshall, and J. Rosenberg, "Integration of Resource Management and Session Initiatio Protocol (SIP)", RFC 3312, October 2002.

[RFC3262] J. Rosenberg, and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

[RFC3388] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.

[RFC3551] Schulzrinne, H., and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 3551, July 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP).", RFC 3711, March 2004.

[RFC3830] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[RFC3851] B. Ramsdell, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[RFC3890] M. Westerlund, "A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP).", RFC 3890, September 2004.

[RFC4474] J. Peterson, and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

[RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol Security Descriptions for Media Streams", RFC 4568, July 2006.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-Time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC4756] A. Li, "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, November 2006.

[RFC5027] Andreasen, F. and D. Wing, "Security Preconditions for Session Description Protocol Media Streams", RFC 5027, October 2007.

[BESRTP] Kaplan, H., and F. Audet, "Session Description Protocol (SDP) Offer/Answer Negotiation for Best-Effort Secure Real-Time Transport Protocol, Work in progress, August 2006.

[ICE] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", work in progress, September 2007.

[ICETCP] J. Rosenberg, "TCP Candidates with Interactive Connectivity Establishment (ICE)", work in progress, July 2007.

[SAVPF] Ott, J., and E Carrara, "Extended Secure RTP Profile for RTCP-based Feedback (RTP/SAVPF)", Work in Progress, May 2007.

[SDPCapNeg] Andreasen, F. "SDP Capability Negotiation", work in progress, December 2006.

[SDPng] Kutscher, D., Ott, J., and C. Bormann, "Session Description and Capability Negotiation", Work in Progress, February 2005.

Author's Addresses

Flemming Andreasen
Cisco Systems
Edison, NJ

Email: fandreas@cisco.com

Intellectual Property Statement

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.