Network Working Group Kutscher
Internet-Draft Ott
Expires: September 1, 2003 Bormann
TZI, Universitaet Bremen
March 03, 2003

Session Description and Capability Negotiation
draft-ietf-mmusic-sdpng-06.txt

Status of this Memo

Copyright Notice

Abstract

This document defines a language for describing multimedia sessions with respect to configuration parameters and capabilities of end-systems.

This document is a product of the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at mmusic@ietf.org and/or the authors.

Document Revision

$Revision: 6.9 $

Table of Contents

1. Introduction

Multiparty multimedia conferencing is one of the applications that require dynamic interchange of end-system capabilities and the negotiation of a parameter set that is appropriate for all sending and receiving end-systems in a conference. For some applications, e.g. for loosely coupled conferences or for broadcast scenarios, it may be sufficient to simply have session parameters be fixed by the initiator of a conference. In such a scenario no negotiation is required because only those participants with media tools that support the predefined settings can join a media session and/or a conference.

This approach is applicable for conferences that are announced some time ahead of the actual start date of the conference. Potential participants can check the availability of media tools in advance and tools such as session directories can configure media tools upon startup. This procedure however fails to work for conferences initiated spontaneously including Internet phone calls or ad-hoc multiparty conferences. Fixed settings for parameters such as media types, their encoding etc. can easily inhibit the initiation of conferences, for example in situations where a caller insists on a fixed audio encoding that is not available at the callee's end-system.

To allow for spontaneous conferences, the process of defining a conference's parameter set must therefore be performed either at conference start (for closed conferences) or maybe (potentially) even repeatedly every time a new participant joins an active conference. The latter approach may not be appropriate for every type of conference without applying certain policies: For conferences with TV-broadcast or lecture characteristics (one main active source) it is usually not desired to re-negotiate parameters every time a new participant with an exotic configuration joins because it may inconvenience existing participants or even exclude the main source from media sessions. But conferences with equal "rights" for participants that are open for new participants on the other hand would need a different model of dynamic capability negotiation, for example a telephone call that is extended to a 3-parties conference at some time during the session.

SDP [2] allows to specify multimedia sessions (i.e. conferences, "session" as used here is not to be confused with "RTP session"!) by providing general information about the session as a whole and specifications for all the media streams (RTP sessions and others) to be used to exchange information within the multimedia session.

Currently, media descriptions in SDP are used for two purposes:

- o to describe session parameters for announcements and invitations (the original purpose of SDP) and

- o to describe the capabilities of a system and possibly provide a choice between a number of alternatives (which SDP was not designed for).

A distinction between these two "sets of semantics" is only made implicitly.

This document is based upon a set of requirements specified in a companion document [1]. In the following, we first introduce a model for session description and capability negotiation as well as the basic terms used throughout this specification (Section 2). In Section 3, we provide an overview of options for capability negotiation. Next, we outline the concept for the concepts underlying SDPng and introduce the syntactical components step by step in Section 4.

Appendix B lists the change history.

2. Terminology and System Model

Any (computer) system has, at a time, a number of rather fixed hardware as well as software resources. These resources ultimately define the limitations on what can be captured, displayed, rendered, replayed, etc. with this particular device. We term features enabled and restricted by these resources "system capabilities".

Example: System capabilities may include: a limitation of the screen resolution for true color by the graphics board; available audio hardware or software may offer only certain media encodings (e.g. G.711 and G.723.1 but not GSM); and CPU processing power and quality of implementation may constrain the possible video encoding algorithms.

In multiparty multimedia conferences, participants employ different "components" in conducting the conference.

Example: In lecture multicast conferences one component might be the voice transmission for the lecturer, another the transmission of video pictures showing the lecturer and the third the transmission of presentation material.

Depending on system capabilities, user preferences and other technical and political constraints, different configurations can be chosen to accomplish the use of these components in a conference.

Each component can be characterized at least by (a) its intended use (i.e. the function it shall provide) and (b) one or more possible ways to realize this function. Each way of realizing a particular function is referred to as a "configuration".

Example: A conference component's intended use may be to make transparencies of a presentation visible to the audience on the Mbone. This can be achieved either by a video camera capturing the image and transmitting a video stream via some video tool or by loading a copy of the slides into a distributed electronic white-board. For each of these cases, additional parameters may exist, variations of which lead to additional configurations (see below).

Two configurations are considered different regardless of whether they employ entirely different mechanisms and protocols (as in the previous example) or they choose the same and differ only in a single parameter.

Example: In case of video transmission, a JPEG-based still image protocol may be used, H.261 encoded CIF images could be sent, as

could H.261 encoded QCIF images. All three cases constitute different configurations. Of course there are many more detailed protocol parameters.

Each component's configurations are limited by the participating system's capabilities. In addition, the intended use of a component may constrain the possible configurations further to a subset suitable for the particular component's purpose.

Example: In a system for highly interactive audio communication the component responsible for audio may decide not to use the available G.723.1 audio codec to avoid the additional latency but only use G.711. This would be reflected in this component only showing configurations based upon G.711. Still, multiple configurations are possible, e.g. depending on the use of A-law or u-Law, packetization and redundancy parameters, etc.

In modeling multimedia sessions, we distinguish two types of configurations:

- potential configurations
  (a set of any number of configurations per component) indicating a system's functional capabilities as constrained by the intended use of the various components;

- actual configurations
  (exactly one per instance of a component) reflecting the mode of operation of this component's particular instantiation.

  Example: The potential configuration of the aforementioned video component may indicate support for JPEG, H.261/CIF, and H.261/QCIF. A particular instantiation for a video conference may use the actual configuration of H.261/CIF for exchanging video streams.

In summary, the key terms of this model are:

- A multimedia session (streaming or conference) consists of one or more conference components for multimedia "interaction".

- A component describes a particular type of interaction (e.g. audio conversation, slide presentation) that can be realized by means of different applications (possibly using different protocols).

- A configuration is a set of parameters that are required to implement a certain variation (realization) of a certain component. There are actual and potential configurations.

* Potential configurations describe possible configurations that are supported by an end-system.

* An actual configuration is an "instantiation" of one of the potential configurations, i.e. a decision how to realize a certain component.

In less abstract words, potential configurations describe what a system can do ("capabilities") and actual configurations describe how a system is configured to operate at a certain point in time (media stream spec).

To decide on a certain actual configuration, a negotiation process needs to take place between the involved peers:

1. to determine which potential configuration(s) they have in common, and

2. to select one of this shared set of common potential configurations to be used for information exchange (e.g. based upon preferences, external constraints, etc.).

Note that the meaning of the term "actual configuration" is highly application-specific. For example, for audio transport using RTP, an actual configuration is equivalent to a payload format (potentially plus format parameters), whereas for other applications it may be a MIME type.

In SAP-based [9] session announcements on the Mbone, for which SDP was originally developed, the negotiation procedure is non-existent. Instead, the announcement contains the media stream description sent out (i.e. the actual configurations) which implicitly describe what a receiver must understand to participate.

In point-to-point scenarios, the negotiation procedure is typically carried out implicitly: each party informs the other about what it can receive and the respective sender chooses from this set a configuration that it can transmit.

Capability negotiation must not only work for 2-party conferences but is also required for multi-party conferences. Especially for the latter case it is required that the process to determine the subset of allowable potential configurations is deterministic to reduce the number of required round trips before a session can be established. For instance, in order to be used with SIP, the capability negotiation is required to work with the offer/answer model that is for session initiation with SIP -- limiting the negotiation to exactly one round trip.

The requirements for the SDPng specification, subdivided into general requirements and requirements for session descriptions, potential and actual configurations as well as negotiation rules, are captured in a companion document [1].

The following list explains some terms used in this document:

Actual Configuration
: An actual configuration is an "instantiation" of one of the potential configurations, i.e. a decision how to realize a certain component.

Component
: A component describes a particular type of interaction (e.g. audio conversation, slide presentation) that can be realized by means of different applications (possibly using different protocols).

Library
: A library is a application specific collection of potential configuration definition. For example, the RTP-AVP library would include definitions for the audio and video codecs of the RTP audio/video profile (AVP).

Package
: A package is application specific data schema for expressing potential and actual configurations. For example, an audio package specifies the data schema for audio codecs.

Potential Configuration
: Potential configurations describe possible configurations that are supported by an end-system ("capabilities").

3. Capability Negotiation: Overview and Requirements

SDPng is a description language for both potential configurations (i.e. capabilities) of participants in multimedia conferences and for actual configurations (i.e. final specifications of parameters). Capability negotiation is the process of generating a usable set of potential configurations and finally an actual configuration from a set of potential configurations provided by each potential participant in a multimedia conference.

SDPng supports the specification of endpoint capabilities and defines a negotiation process: In a negotiation process, capability descriptions are exchanged between participants. These descriptions are processed in a "collapsing" step which results in a set of commonly supported potential configurations. In a second step, the final actual configuration is determined that is used for a conference. This section specifies the usage of SDPng for capability negotiation. It defines the collapsing algorithm and the procedures for exchanging SDPng documents in a negotiation phase.

The description language and the rules for the negotiation phase that are defined here are (in general) independent of the means by which descriptions are conveyed during a negotiation phase (a reliable transport service with causal ordering is assumed). There are however properties and requirements of call signaling protocols that have been considered to allow for a seamless integration of the negotiation into the call setup process. For example, in order to be usable with SIP, it must be possible to negotiate the conference configuration within the two-way-handshake of the call setup phase. In order to use SDPng instead of SDP according to the offer/answer model defined in [16] it must be possible to determine an actual configuration in a single request/response cycle.

3.1 Outline of the Negotiation Process

Conceptually, the negotiation process comprises the following individual steps (considering two parties, A and B, where A tries to invite B to a conference). Please note that this describes the steps of the negotiation process conceptually -- it does not specify requirements for implementations. Specific procedures that MUST be followed by implementations are given below.

1. A determines its potential configurations for the components that should be used in the conference (e.g. "interactive audio" and "shared whiteboard") and sends a corresponding SDPng instance to B. This SDPng instances is denoted "CAP(A)".

2. B receives A's SDPng instance and analyzes the set of components

(sdpng:c elements) in the description. For each component that B wishes to support it generates a list of potential configurations corresponding to B's capabilities, denoted "CAP(B)".

3. B applies the collapsing function and obtains a list of potential configurations that both A and B can support, denoted "CAP(A)xCAP(B) = CAP(AB)".

4. B sends CAP(B) to A.

5. A also applies the collapsing function and obtains "CAP(AB)". At this step, both A and B know the capabilities of each other and the potential configurations that both can support.

6. In order to obtain an actual configuration from the potential configuration that has been obtained, both participants have to pick a subset of the potential configurations that should actually be used in the conference and generate the actual configuration. It should be noted that it depends on the specific application whether each component must be assigned exactly one actual configuration (one sdpng:alt element) or whether it is allowed to list multiple actual configurations. In this model we assume that A selects the actual configuration, denoted CFG(AB).

7. A augments CFG(AB) with the transport parameters it intends to use, e.g., on which endpoint addresses A wishes to receive data, obtaining CFG_T(A). A sends CFG_T(A) to A.

8. B receives CFG_T(A) and adds its own transport parameters, resulting in CFG_T(AB). CFG_T(AB) contains the selected actual configurations and the transport parameters of both A and B (plus any other SDPng data, e.g., meta-information on the conference). CFG_T(AB) is the complete conference description. Both A and B now have the following information:

   CAP(A) A's supported potential configurations

   CAP(B) B's supported potential configurations

   CAP(AB) The set of potential configurations supported by both A and B.

   CFG(AB) The set of actual configurations to be used.

   CFG_T(AB) The set of actual configurations to be used augmented with all required parameters.

In this model, the capability negotiation and configuration exchange process leads to a description that represents a global view of the configuration that should be used. This means, it contains the complete configuration for all participants including per-participant information like transport parameters.

Note that the model presented here results in four SDPng messages. As an optimization, this procedure can be abbreviated to two exchanges by including the transport (and other) parameters into the potential configurations. A embeds its desired transport parameters into the list of potential configurations and B also sends all required parameters in the response together with B's potential configurations. Both A and B can then derive CFG_T(AB). Transport parameters are usually not negotiable, therefor they have to be distinguished from other configuration information.

Specific procedures for re-negotiation and multi-party negotiation will be defined in a future version of this document.

3.2 The Negotiation Process

The algorithm for comparing two potential configurations and for obtaining a commonly supported subset is application specific. For some limited application scenarios, a application specific offer/answer process may be employed such as the SDP offer/answer model [16].

More advanced implementations require a generic capability negotiation mechanism that allows for application-independent negotiation of potential configuration with parameters from different application domains. Capability negotiation frameworks such as RFC 2533 [18] can be employed for this purpose. In a future version of this document, we will discuss of employing a RFC 2533 based negotiation process for comparing and matching capability descriptions in SDPng documents.

4. SDPng

This section introduces the underlying concepts of the Session Description Protocol - next generation (SDPng). The focus of this section is on the concepts of the capability description language with a stepwise introduction of the various syntactical elements. Note that this section only provides examples accompanied by explanations. The description elements used in this section are not normative.

4.1 Conceptual Outline

In Section 2 we have distinguished between potential configurations ("capabilities") and actual configurations ("session descriptions"). SDPng provides the possibility to express potential configurations and actual configurations in one document. A potential configuration list is used to declare capabilities and an actual configuration list is used to declare concrete configurations.

Potential configurations are described independently of actual configurations. In a "potential configurations" section, a user agent lists its capabilities as a list of named definitions. For negotiating capabilities from different user agents, the individual definitions are matched in order to determine a commonly supported subset of capabilities. The data schema for potential configurations is defined in "package definitions". An example for an element of a potential configuration would be the definition of a supported audio codec.

Actual configurations can refer to capabilities and specify concrete parameters for application protocol sessions, including transport parameters. Actual configurations cannot be negotiated.

When defining potential configurations, capabilities are never expressed with respect to other potential configuration elements, e.g., the definition of an audio codec capability does not limit the capability of using other audio codec. Constraints like the simultaneous usage of capabilities can be expressed separately from the capabilities themselves.

In addition, information about the communication session itself, such as scheduling, information on the semantics of application protocol sessions and information on the user who has initiated a conference. This information is expressed separately from the definition of potential and actual configurations.

These different elements of session description are discussed in detail in the following sections. There are four different elements:

Potential Configurations; see Section 4.1.1.

Actual Configurations; see Section 4.1.2.

Constraints; see Section 4.1.3.

Session meta information; see Section 4.1.4.

4.1.1 Potential Configurations

A "Potential Configurations" section in an SDPng document lists individual capabilities, e.g., codec capabilities. In a capability negotiation process these potential configurations may be compared to the potential configurations that are defined in an SDPng document from another participant. The outline of such a negotiation process is presented in Section 3.

Please note, that in the following examples, we use a straw-man syntax in order to discuss the concepts. A final syntax will be formally defined in a future version of this document.

These are two examples of elements in a potential configurations section:

```
audio:codec
  name=audio-basic
  encoding="PCMU"
  sampling="8000"
  channels="[1]"


audio:codec
  name="audio-L16-mono"
  encoding="L16"
  sampling="44100"
  channels="[1,2,4]"
```

The following requirements can be stated for expressing potential configurations:

- o It must be possible to name potential configuration elements. In the example above, this is achieved by the property "name". Names MUST NOT be considered in a capability negotiation process.

- o The potential configuration elements are referred to by this name for specifying actual configurations. It MUST be ensured that names that originate from different description documents reside

in separate namespaces in order to avoid collisions.

- The properties of a given potential configuration element MUST have a well-defined type. For example, codec type names are expressed as strings, and for capability negotiation, two codec names can be processed by applying a string comparison. A maximum frame-rate would be expressed as a number that represents an upper limit, and for capability negotiation, the minimum of two numbers would be used as a commonly supported value for the frame-rate.

- User agents MUST be able to infer the type of a given property without referring to an external schema definition, i.e., the type must be specified either implicitly or explicitly. Note, that in the example above, the type is not specified.

- In addition to the data type and its value a property can provide other characteristics: Some properties that a package definition defines for a certain application are mandatory, i.e., they must be specified in configuration descriptions. In the example above, this would apply to the encoding, the sampling-rate and the number of channels. For this single potential configuration elements these properties serve as constraints for a negotiation: The capability description matches only those description from other participants that provide the same encoding, the same sampling-rate and either 1,2 or 4 channels. If a description did not provide one of these properties, the negotiation would fail. There are however properties that can represent optional parameters, such as a codec parameter that can optionally be used. If one participant specified such a property and another participant did not, we would expect the resulting configuration to not include that property, however, the negotiation itself should be successful.

- Some capabilities such as codec capabilities may be associated with additional constraints, e.g., the directionality of media streams ('send-only', 'receive-only'). It will be defined in a future version of this document whether the directionality is specified as a capability (in a potential configuration) or whether it is rather specified as an attribute of an actual configuration.

With these requirements in mind, we add additional characteristics to the properties in potential configuration descriptions (and change the encoding for the second potential configuration element):

```
audio:codec
  name=audio-basic;type=name
  encoding="PCMU";type=string
  sampling="8000";type=maximum-limit
  channels="[1]";type=set
  featureX="200";type=maximum-limit;optional


audio:codec
  name="audio-PCMU-44khz";type=name
  encoding="PCMU";type=string
  sampling="44100";type=maximum-limit
  channels="[1,2,4]";type=set
  featureY="200";type=string;optional
```

Note again, that these descriptions merely present examples in order to present the data model that we use for potential configurations -- this is not the SDPng syntax. In these examples, we have added the optional features 'featureX' and 'featureY'.

If we assume, that the two potential configurations are contributions from different participants for a capability negotiation, a resulting potential configuration, after a negotiation process as outlined in Section 3, could look like this:

```
audio:codec
  encoding="PCMU";type=string
  sampling="8000";type=maximum-limit
  channels="[1]";type=set
```

The name cannot be considered for a capability negotiation, the optional properties 'featureX' and 'featureY' have only been provided by one participant each and the other properties have been processed by the negotiation algorithms (that will be specified in a future version of this document in Section 3).

So far, we have only considered codec capabilities. Other capabilities would include transport mechanisms, e.g., RTP/UDP/IPv4:

```
rtp-udp:transport
  name="rtp-udp-ipv4";type=name
  network="IP4;type=string
```

4.1.2 Actual Configurations

The "Actual Configurations" section lists all the components that constitute the multimedia application (IP telephone call, real-time

streaming application, multi-player gaming session etc.). For each of these components, the actual configurations are given. Potential configurations are used during capability exchange and/or negotiation, actual configurations to configure media streams after negotiation (e.g. with RTSP) or in session announcements (e.g. via SAP).

Each component is labeled with an identifier so that it can be referenced, e.g. to associate semantics with a particular media stream. For such a component, any number of actual configurations may be given with each configuration describing an alternative way to realize the functionality of the respective component.

The semantics of this are application dependent. For example, for SIP applications using the SDP offer/answer model, providing multiple alternatives for a component (a media type in SDP) means that the offerer is prepared to receive at any of the specified addresses any of the specified payload formats (for RTP applications). In this example, the order of alternatives is used to specify a preference, i.e., the first alternative is the most preferred one.

The following example provides two alternative configurations for a component named "interactive-audio". Each alternative refers to the RTP-transport capability named "rtp-udp-ipv4" and to an audio-codec capability.

```
component
  name="interactive-audio"
  alt
    name="AVP-audio-0"
    rtp-udp:transport
      ref="rtp-udp-ipv4"
      pt="96"
      direction="send-receive"
      addr="224.2.0.53"
      rtp-port="7800"
      rtcp-port="7801"
    audio-codec
      ref="audio-basic"

  alt
    name="AVP-audio-11"
      rtp-udp:transport
      ref="rtp-udp-ipv4"
      pt="97"
      direction="send-receive"
      addr="224.2.0.53"
      rtp-port="7800"
      rtcp-port="7801"
    audio-codec
      ref="audio-L16-stereo"
```

For the RTP transport configuration, additional required parameters are provided, such as the payload type number to be used (pt="97"), the IP address and UDP port numbers for RTP and RTCP and the directionality.

Note that in the example above, the actual configuration of the RTP transport is identical for both alternatives -- with the exception of the payload type number. In a final solution, this duplication should be avoided by another level of indirection, i.e., by defining these parameters once and referencing this definition where needed.

In order to determine the usable actual configurations after a capability negotiation, a user agent has to traverse the references in actual configurations to potential configurations and check whether each capability is still supported after a negotiation process. Only those alternatives that reference supported capabilities can be considered for implementing the given component.

The semantics of specifying multiple alternatives for a component are application specific -- for RTP configurations in SDP it means that the endpoint is willing to receive any of the specified formats without further out-of-band signaling and that the first

configuration is preferred.

4.1.3 Constraints

Potential configurations are media, transport, and other capabilities, whereas configurations indicate which combinations of these could be used to provide the desired functionality in a certain setting.

There may, however, be further constraints within a system (such as CPU cycles, DSP resources available, dedicated hardware, etc.) that limit which of these configurations can be instantiated in parallel (and how many instances of these may exist). We deliberately do not couple this aspect of system resource limitations to the various application semantics as the constraints may exist across application boundaries. Also, in many cases, expressing such constraints is simply not necessary (as many uses of the current SDP show), so additional overhead can be avoided where this is not needed.

The usage of constraints will be specified in a future version of this document.

4.1.4 Meta Information

The fourth and final section of an SDPng description is used to specify meta information such as session layer attributes. These attributes largely include those defined by SDP [RFC2327] (which are explicitly indicated in the following specification) to describe originator, purpose, and timing of a multimedia session among other characteristics. Furthermore, SDPng includes attributes indicating the semantics of the various Components in a teleconference or other session.

A session-level specification for connection information (SDP "c=" line), bandwidth information (SDP "b=" line), and encryption keys (SDP "k=" lines) is deliberately not provided for in SDPng. The relevant information can be specified directly in the Configuration section for individual alternatives.

The section for meta information will provide for integrating and re-using existing meta-information frameworks such as MPEG-7. Details will be specified in a future version of this document.

4.2 Syntax Definition Mechanisms

In this section, we specify the syntax definition mechanisms for SDPng.

In order to allow for the possibility to validate session descriptions and in order to allow for structured extensibility, SDPng relies on a syntax framework that provides concepts as well as concrete procedures for document validation and extending the set of allowed syntax elements.

SGML/XML technologies allow for the creation of Document Type Definitions (DTDs) that can define the allowed content models for the elements of conforming documents. Documents can be formally validated against a given DTD to check their conformance and correctness. XML DTDs however, cannot easily be extended. It is not possible to alter to content models of element types or to add new element types by third-party definition packages without creating the possibility of name collisions.

For SDPng, a mechanism is needed that allows the specification of a base syntax -- for example basic elements for the high level structure of description documents -- while allowing extensions, for example elements and attributes for new transport mechanisms, new media types etc. to be added on demand. Still, it has to be ensured that extensions do not result in name collisions. Furthermore, it must be possible for applications that process descriptions documents to distinguish extensions from base definitions.

For XML, mechanisms have been defined that allow for structured extensibility of document schemata: XML Namespace and XML Schema.

XML Schema mechanisms allow to constrain the allowed document content, e.g. for documents that contain structured data and also provide the possibility that document instances can conform to several XML Schema definitions at the same time, while allowing Schema validators to check the conformance of these documents.

Extensions of the session description language, say for allowing to express the parameters of a new media type, would require the creation of a corresponding XML schema definition that contains the specification of element types that can be used to describe configurations of components for the new media type. Session description documents have to reference the extension Schema module, thus enabling parsers and validators to identify the elements of the new extension module and to either ignore them (if they are not supported) or to consider them for processing the session/capability description.

It is important to note that the functionality of validating capability and session description documents is not necessarily required to generate or process them. For example, endpoints would be configured to understand only those parts of description documents

that are conforming to the baseline specification and simply ignore extensions they cannot support. The usage of XML and XML Schema is thus rather motivated by the need to allow for extensions being defined and added to the language in a structured way that does not preclude the possibility to have applications to identify and process the extensions elements they might support. The baseline specification of XML Schema definitions and packages must be well-defined and targeted to the set of parameters that are relevant for the protocols and algorithms of the Internet Multimedia Conferencing Architecture, i.e. transport over RTP/UDP/IP, the audio video profile of RFC1890 etc.

Section 4.4 describes package definitions and library definition.

4.3 Referencing Definitions

SDPng provides a referencing concept for definitions. For example, in the specification of an actual configuration, we reference the capabilities of the potential configurations section.

The concrete reference mechanism depends on the syntax in use and will be specified in a future version of this document.

4.4 External Definition Packages

There are two types of external definitions:

Package Definitions (Section 4.4.1) define rules, i.e., a data schema, for specifying parameters that are not covered by the base SDPng specification.

Library Definitions (Section 4.4.2) contain definitions that can be referenced in SDPng documents.

4.4.1 Package Definitions

In order to allow for extensibility it must be possible to define extensions to the basic SDPng configuration options.

For example, if some application requires the use of a new transport protocol, endpoints must be able to describe their configuration with respect to the parameters of that transport protocol. The mandatory and optional parameters that can be configured and negotiated when using the transport protocol will be specified in a definition document. Such a definition document is called a "package".

A package contains rules that specify how SDPng is used to describe

conferences or end-system capabilities with respect to the parameters of the package. The specific properties of the package definitions mechanism are still to be defined.

An example of such a package would be the RTP package that defines how to specify RTP parameters. Another example would be the audio codec package that defines how specify audio codec parameters.

4.4.2 Library Definitions

While package definitions specify the allowed parameters for a given profile, SDPng "Definitions" sections refer to package definitions and define concrete configurations based on a specific package.

In order for such definitions to be imported into SDPng documents, "SDPng libraries" may be defined and referenced in SDPng documents. A library is a set of definitions that is conforming to one or more package definitions.

The purpose of the library concept is to allow certain common definitions to be factored-out so that not every SDPng document has to include the basic definitions, for example the PCMU codec definition. SDP [2] uses a similar concept by relying on the well known static payload types (defined in RFC1890 [4]) that are also just referenced but never defined in SDP documents.

An SPDng document that references definitions from an external library has to declare the use of the external library. The external library, being a set of configuration definitions for a given package, again needs to declare the use of the package that it is conforming to. A library itself can make reference to other external libraries.

There are different possibilities of how package definitions and libraries can be used in SDPng documents:

o In an SPDng document, a package definition can be referenced and all the configuration definitions are provided within the document itself. The SDPng document is self-contained with respect to the definitions it uses.

o In an SPDng document, the use of an external library can be declared. The library references a package definition and the SDPng document references the library. There are two alternatives how external libraries can be referenced:

by name: Referencing libraries by names implies the use of a registration authority where definitions and reference names

can be registered with. It is conceivable that the most common SDPng definitions be registered that way and that there will be a baseline set of definitions that minimal implementations must understand. Secondly, a registration procedure will be defined, that allows vendors to register frequently used definitions with a registration authority (e.g., IANA) and to declare the use of registered definition packages in conforming SDPng documents. Of course, care should be taken not to make the external references too complex and thus require too much a priori knowledge in a protocol engine implementing SDPng. Relying on this mechanism in general is also problematic because it impedes the extensibility, as it requires implementors to provide support for new extensions in their products before they can inter-operate. Registration is not useful for spontaneous or experimental extensions that are defined in an SDPng library.

by address: An alternative to referencing libraries by name is to declare the use of an external library by providing an address, i.e., an URL, that specifies where the library can be obtained. While this allows the use of arbitrary third-party libraries that can extend the basic SDPng set of configuration options in many ways, in introduces additional complexity that could result in in higher latency for the processing of a description document with references to external libraries. In addition, there are problems if the referenced libraries cannot be accessed by all communication partners.

- o Because of these problematic properties of external libraries, the final SDPng specification will have to provide a set of recommendations under which circumstances the different mechanisms of referring to external definitions should be used.

4.5 Mappings

A mapping needs to be defined in particular to SDP that allows to translate final session descriptions (i.e. the result of capability negotiation processes) to SDP documents. In principle, this can be done in a rather schematic fashion for the base specification and a set of basic packages.

In addition, mappings to H.245 will be defined in order to support applications like SIP-H.323 gateways.

5. Syntax Definition

An SDPng description is an XML document with different element types for the different sections. The SDPng base syntax specification defines this overall document structure.

5.1 Potential Configurations

A section for potential configurations is an XML element that can provide a list of child elements. Each child element represents an individual capability as described in Section 4.1.1. Each property is represented by an XML attribute. The element types are defined in package definitions. XML Namespaces are used to disambiguate element types and to allow for extensibility.

Each element MUST provide an attribute "name". The value of this attribute SHOULD be composed of a prefix (representing a namespace-name) and a unique name for the corresponding capability within that namespace. The namespace-name designates a namespace for the source of the capability definition. If a prefix is specified, it MUST be separated by a colon (':') from the name.

Each element represents a "feature set" (using the terminology of [18]). Therefore, each attribute can provide a "range" of values -- not only a single value. For example, an attribute can specify a set of supported alternative values for a given property, e.g., for the sampling rate of an audio codec. SDPng provides two different ways for representing "value ranges": An attribute can specify a set of tokens or a numerical range.

Each property that is represented by an XML attribute has a well-defined type that is specified in the package definition. The type is encoded implicitly in the attribute value (similar to the syntax in RFC 2533 [18]). The following types are distinguished:

Text strings, tokens

An attribute may provide a token (a symbolic name), e.g., for a codec name.
An example for a corresponding attribute:

```
encoding="PCMU"
```

Token MUST be directly embedded into the attribute content, i.e., the token is the attribute value.
The complete formal syntax definition of tokens will be provided in a later version of this document.

Token sets

An attribute can specify a set of tokens (representing a list of alternative values for a certain property).
An example for a corresponding attribute:

```
sampling="[8000,16000,44100]"
```

A token set MUST be specified as a list of tokens that are separated by commas (',') and are enclosed by square brackets ('[',']').
The complete formal syntax definition of token sets will be provided in a later version of this document.

Numbers and Numerical ranges

An attribute can specify a number or a range (minimum and maximum) for numbers.
An example for an attribute specifying a number:

```
bitrate="(64)"
```

A number MUST be specified as a literal value in brackets ('(', ')').
An example for an attribute specifying a numerical range:

```
bitrate="(64,128)"
```

A numerical range MUST be specified as a pair of values. The first value is the minimum value (included in the range) and the seconds value is the maximum value (included in the range). The values are separated by a comma (',') and are enclosed in ('(', ')'). One of the range limits MAY be omitted, i.e., either the minimum or the maximum value, e.g., if the range has no upper or lower limit.
An example for an attribute specifying a numerical range without an upper limit:

```
bitrate="(64,)"
```

The complete formal syntax definition of numbers and numerical ranges (and a definition of the exact number type) will be provided in a later version of this document.

An example for an XML element describing an individual capability:

```
<audio:codec name="avp:pcmu" encoding="PCMU" channels="[1,2]"
sampling="[8000,16000]"/>
```

Capability elements MAY also provide attribute from different XML namespaces. For example, a video-codec capability MAY be described with attributes declaring general video capabilities, and this

element MAY provide a list of additional codec specific attributes, as depicted in the following example:

```
<video:codec name="h263+-enhanced" resolution="QCIF" frame-rate="(,24)"
                h263plus:A="foo" h263plus:B="bar" />
```

The definition of "optional properties" (properties that to do not constitute a constraint but not optional enhancement -- see Section 4.1.1) will be provided in a future version of this document.

6. Specification of the Capability Negotiation

The SDPng specification defines the syntax and the semantics of capability declarations (potential configurations). The algorithms that are used for processing descriptions and for comparing capability descriptions from different participants are application specific.

In this section we discuss two alternative algorithms for implementations: A model that is base on the SDP offer/answer scheme (Section 6.1 and a model that is based on the feature matching algorithm that is specified in RFC 2533 [18] (Section 6.2).

6.1 Offer/Answer

The offer/answer model allows to communicating peers to determine a (common) mode of operation to exchange media streams in a single round-trip. Basically, the offerer proposes a set of components, providing one or more alternatives ("potential configurations") for each of these. From this offer, the answerer learns which components may be used and which configurations are conceivable to realize these components. The answerer indicates which components it supports (e.g. disallow a video session and go with a audio-only conversation) and also provides possible configurations to implement those components. Along with the media types and codec parameters, offerer and answerer specify which transport addresses to use and, in case of RTP, which payload types they want to use for sending. Offerer and answerer agree on a common set of media streams ("components") and on a possible set of codecs ("configurations") as well as the transport addresses and other parameters to be used. However, they do not fix a certain configuration (unless only a single one is exchanged in each direction). Instead, for each selected media stream, either peer may choose and dynamically switch to any of the configurations indicated by the other side in the respective offer or answer.

For using SDPng with the offer/answer model (RFC 3264), the following considerations apply to the SDPng documents:

- For each component to be used, all necessary parameters for at least one actual configuration MUST be given, i.e. transport addresses and payload formats MUST be specified along with the capabilities.

- Matching of components is done based upon their identification in the session part of the SDPng document using predefined identifiers.

- Components that shall not be instantiated (i.e. that are refused by the answerer) shall either be present but have all parameters of the actual configuration removed (i.e. no transport addresses, etc.) if they may be (re-)instantiated at a later stage. Or they shall be removed entirely from the answer if the respective component is not supported at all. In the latter case, the corresponding configurations MUST be removed from both the configuration section and the session section of the SDPng document.

- For each component, the alternative potential configurations MUST be listed in the order of preference. A certain preference indicator ("q=" value) may be included in a future revision of this document. The considerations of RFC 3264 to simply arriving at symmetric codec use apply.

The rules for matching properties and determining answers based upon the offers are similar to those specified in RFC 3264.

A future revision of this document will define the details for all the attributes discussed in RFC 3264 that require special considerations (e.g. the directionality attribute for media streams).

6.2 RFC2533 Negotiation

SDPng potential configurations can be processed using the RFC 2533 algorithm as defined in [18]. This involves the following steps:

Translating SDPng potential configurations to RFC 2533 feature set expressions;

Applying the RFC 2533 feature match algorithm; and

Integrating the resulting feature set expressions into the SDPng selection of actual configurations.

6.2.1 Translating SDPng to RFC 2533 Expressions

SDPng potential configurations can be translated to RFC 2533 feature sets in a straightforward way, because SDPng uses a subset of the mechanisms provided by RFC 2533 with a different syntax.

Each capability in an SDPng section for potential configurations is represented as an XML element with a set of attributes. We first describe how to translate a single capability element into a RFC 2533 feature set, and then consider the combination of multiple capability

elements.

Basically, all attributes of an SDPng capability element and its child elements MUST be transformed to a RFC 2533 expression, whereas each attribute MUST be translated to a feature predicate. The resulting feature predicate are combined using the '&' (AND) operator. The name attributes MUST NOT be considered.

Each predicate MUST be encapsulated by brackets ('(', ')'). Each XML attribute value is taken as a feature predicate value, i.e., the quote are not considered. Each attribute name is directly adopted as a feature tag, including the namespace name.

The SDPng data types map to RFC 2533 feature types as follows:

Token
A token MUST be directly adopted as a RFC 2533 token.

Token set
A token set MUST be directly adopted as a RFC 2533 set.

Number
A single number in round brackets MUST be adopted as a RFC 2533 number. The brackets MUST be removed.

Numerical Ranges
A numerical range MUST be transformed to a feature set expression with two feature predicates that are combined using the "&" (AND) operator. The first predicate specifies the lower limit and the second predicate specified the upper limit.
For example, the attribute bitrate="(64,128)" would be transformed to the following feature set:

```
(& (bitrate>=64) (bitrate<=128))
```

A numerical range without a lower limit MUST be transformed to a corresponding predicate with a '<=' operator and a numerical range without a upper limit MUST be transformed to a corresponding predicate with a '>=' operator.
For example, the attribute bitrate="(,128)" would be transformed to the following feature set:

```
(bitrate<=128)
```

The following sample SDPng potential configuration would be transformed as follows:

Original SDPng expression:

```
<video:codec name="h263+-enhanced" resolution="QCIF" frame-rate="(,24)"
                h263plus:A="foo" h263plus:B="bar" />
```

Transforming attributes to feature predicates:

```
(& (resolution=QCIF) (frame-rate<=24) (h263plus:A=foo) (h263plus:B=bar))
```

Note that in example above, the namespace name is not used for feature tags, instead we use the namespace prefix (for abbreviation). RFC 2533 uses the syntax rules of RFC 2506 for feature tags. An additional requirement for transforming fully-qualified attribute names (including namespace names) to feature tags will be specified in a future version of this document.

Multiple independent capability elements MUST each be transformed using the specification above and then combined into a single RFC 2533 feature set by connecting the individual feature sets using the '|' (OR) operator. For example, the following sample SDPng potential configuration would be transformed as follows:

```
   <audio:codec name="avp:pcmu" encoding="PCMU" channels="[1,2]"
sampling="[8000,16000]"/>
   <video:codec name="h263+-enhanced" resolution="QCIF" frame-rate="(,24)"
                h263plus:A="foo" h263plus:B="bar" />
```

Transforming attributes to feature predicates:

```
(|
  (& (encoding=PCMU) (channels=[1,2]) (sampling=[8000,16000]))
  (& (resolution=QCIF) (frame-rate<=24) (h263plus:A=foo) (h263plus:B=bar))
)
```

Additional requirements for processing "optional" parameters (see Section 5) will be specified in a future version of this document.

6.2.2 Applying RFC 2533 Canonicalization

After transforming different SDPng capability descriptions from different participants into their equivalent RFC 2533 form, the following steps MUST be performed to calculate the common subset of capabilities:

1. The individual feature sets MUST be combined into a single expression by creating conjunction of the feature sets, i.e., the feature sets MUST be connected by the '&' (AND) operator.

2. The resulting expression MUST be reduced to disjunctive normal form, i.e., the canonical from as specified by RFC 2533 [18].

6.2.3 Integrating Feature Sets into SDPng

A feature set that has been created by combining multiple independent feature sets and by reducing the result for canonical form does not indicate directly which of the capability elements belong the common subset of capabilities. The following steps MUST be performed to determine whether an individual capability element (e.g., from one of the contributing SDPng capability descriptions) belongs to the result feature set.

Let R be the result feature set obtained from the canonicalization as specified in Section 6.2.2.

1. For each capability element, generate the equivalent RFC 2533 feature set by applying the steps specified in Section 6.2.1. Let C be the resulting feature set.

2. Combine R and C into a single feature set by building a conjunction of the two feature sets (& R C). Let the result be the feature set T.

3. Reduce T to disjunctive normal form by applying the canonicalization as defined in RFC 2533 [18].

4. If the remaining disjunction is non-empty, the constraints specified by capability element (the origin of C) can be satisfied by R, i.e., C represents a commonly supported capability.

A future version of this document will specify requirements for exchanging calculated capabilities and for selecting appropriate actual configurations.

7. Open Issues

Definition of baseline libraries

Libraries provide partially specified definitions, i.e. without transport parameters. How can SDPng documents reference the definitions and augment them with specific transport parameters?

Referencing extension packages: XML-Schema does not support the declaration of multiple schemas via the schemaLocation attribute. Conceivable solution: When extension packages are used, the SDPng description is a "multi-part" object, that consists of an integrating schema definition (that references all necessary packages and the base definition) and the actual description document that is a schema instance of the integrating schema.

Uniqueness of attribute values: When libraries are used they will contain definition elements with "name" attributes for later referencing. How to avoid name clashes for those identifiers? When an SDPng document uses libraries from different sources they could be incompatible because of name collisions. Possible solution: Prefix such IDs with a namespace name (either explicitly or implicitly by interpreting applications). The explicit prefixes have the advantage that no special knowledge would be required to resolve links at the cost of very long ID values.

A registry (reuse of SDP mechanisms and names etc.) needs to be set up.

Implicit declaration of SDPng schema and default package

Should overwriting of child elements be allowed for referencing existing definitions with the "ref" attribute?

We need a package definition language. XML-DTDs or XML-Schema is not sufficient as we need ways to specify the type (string/symbol, set, numerical range) and additional attributes (optional).

8. Acknowledgements

The authors would like to thank Teodora Guenkova, Goran Petrovic and Markus Nosse for their feedback and detailed comments.

References

[1] Kutscher, D., Ott, J., Bormann, C. and I. Curcio, "Requirements for Session Description and Capability Negotiation", Internet Draft draft-ietf-mmusic-sdpng-req-01.txt, April 2001.

[2] Handley, M. and V. Jacobsen, "SDP: Session Description Protocol", RFC 2327, April 1998.

[3] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobsen, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[4] Schulzrinne, H., "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890, January 1996.

[5] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", draft-ietf-avt-profile-new-10.txt (work in progress), March 2001.

[6] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A. and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[7] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

[8] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.

[9] Handley, M., Perkins, C. and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[10] World Wide Web Consortium (W3C), "Extensible Markup Language (XML) 1.0 (Second Edition)", Status W3C Recommendation, Version http://www.w3.org/TR/2000/REC-xml-20001006, October 2000.

[11] World Wide Web Consortium (W3C), "Namespaces in XML", Status W3C Recommendation, Version http://www.w3.org/TR/1999/REC-xml-names-19990114, January 1999.

[12] World Wide Web Consortium (W3C), "XML Inclusions (XInclude) Version 1.0", Status W3C Candidate Recommendation, Version http://www.w3.org/TR/2002/CR-xinclude-20020221, February 2002.

[13] World Wide Web Consortium (W3C), "XML Schema Part 1: Structures", Version http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/, Status W3C Recommendation, May 2001.

[14] World Wide Web Consortium (W3C), "XML Schema Part 2: Datatypes", Version http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/, Status W3C Recommendation, May 2001.

[15] World Wide Web Consortium (W3C), "XML Linking Language (XLink) Version 1.0", Version http://www.w3.org/TR/2001/REC-xlink-20010627/, Status W3C Recommendation, June 2001.

[16] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with SDP", RFC 3264, June 2002.

[17] Hollenbeck, S., Rose, M. and L. Masinter, "Guidelines for The Use of XML within IETF Protocols", draft-hollenbeck-ietf-xml-guidelines-05.txt (work in progress), June 2002.

[18] Klyne, G., "A Syntax for Describing Media Feature Sets", RFC 2533, March 1999.

Authors' Addresses

Dirk Kutscher
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.218-7595, sip:dku@tzi.org
Fax: +49.421.218-7000
EMail: dku@tzi.uni-bremen.de

Joerg Ott
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.201-7028, sip:jo@tzi.org
Fax: +49.421.218-7000
EMail: jo@tzi.uni-bremen.de

Carsten Bormann
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.218-7024, sip:cabo@tzi.org
Fax: +49.421.218-7000
EMail: cabo@tzi.org

Appendix A. Use of SDPng in conjunction with other IETF Signaling Protocols

The SDPng model provides the notion of Components to indicate the intended types of collaboration between the users in e.g. a teleconferencing scenario.

Three different abstractions are defined that are used for describing the properties of a specific Component:

- o a Capability refers to the fact that one of the involved parties supports one particular way of exchanging media -- defined in terms of transport, codec, and other parameters -- as part of the media session.

- o a Potential Configuration denotes a set of matching Capabilities from all those involved parties required to successfully realize one particular Component.

- o an Actual Configuration indicates the Potential Configuration which was chosen by the involved parties to realize a certain Component at one particular point in time.

As mentioned before, this abstract notion of the interactions between a number of communicating systems needs to be mapped to the application scenarios of SDPng in conjunction with the various IETF signaling protocols: SAP, SIP, RTSP, and MEGACO.

In general, this section provides recommendations and possible scenarios for the use of SDPng within specific protocols and applications. Is does not specify normative requirements.

A.1 The Session Announcement Protocol (SAP)

SAP is used to disseminate a previously created (and typically fixed) session description to a potentially large audience. An interested member of the audience will use the SDPng description contained in SAP to join the announced media sessions.

This means that a SAP announcement contains the Actual Configurations of all Components that are part of the overall teleconference or broadcast.

A SAP announcement may contain multiple Actual Configurations for the same Component. In this case, the "same" (i.e. semantically equivalent) media data from one configuration must be available from each of the Actual Configurations. In practice, this limits the use of multiple Actual Configurations to single-source multicast or

broadcast scenarios.

Each receiver of a SAP announcement with SDPng compares its locally stored Capabilities to realize a certain Component against the Actual Configurations contained in the announcement. If the intersection yields one or more Potential Configurations for the receiver, it chooses the one it sees fit best. If the intersection is empty, the receiver cannot participate in the announced session.

SAP may be substituted by HTTP (in the general case, at least), SMTP, NNTP, or other IETF protocols suitable for conveying a media description from one entity to one or more other without the intend for further negotiation of the session parameters.

Example from the SAP spec. to be provided.

A.2 Session Initiation Protocol (SIP)

SIP is used to establish and modify multimedia sessions, and SDPng may be carried at least in SIP INVITE and ACK messages as well as in a number of responses. From dealing with legacy SDP (and its essential non-suitability for capability negotiation), a particular use and interpretation of SDP has been defined for SIP.

One of the important flexibilities introduced by SIP's usage of SDP is that a sender can change dynamically between all codecs that a receiver has indicated support (and has provided an address) for. Codec changes are not signaled out-of-band but only indicated by the payload type within the media stream. From this arises one important consequence to the conceptual view of a Component within SDPng.

There is no clear distinction between Potential and Actual Configurations. There need not be a single Actual Configuration be chosen at setup time within the SIP signaling. Instead, a number of Potential Configurations is signaled in SIP (with all transport parameters required for carrying media streams) and the Actual Configuration is only identified by the payload type which is actually being transmitted at any point in time.

Note that since SDPng does not explicitly distinguish between Potential and Actual Configurations, this has no implications on the SDPng signaling itself.

SIP relies on an "offer/answer" model for the exchange of capability and configuration information. Either the caller or the callee sends an initial session description that is processed by the other side and returned. For capability negotiation, this means that the negotiation follows a two-stage-process: The "offerer" sends its

capability description to the receiver. The receiver processes the offerers capabilities and his own capabilities and generates a result capability description that is sent back to the offerer. Both sides now know the commonly supported configurations and can initiate the media sessions.

Because of this strict "offer/answer" model, the offerer must already send complete configurations (i.e. include transport addresses) along with the capability descriptions. The answer must also contain complete configuration parameters. The following figure shows, how SDPng content can be used in an INVITE request with a correspong 200 OK message.

Simple description document with only one alternative:

```
   F1 INVITE A -> B

   INVITE sip:B@example.com SIP/2.0
   Via: SIP/2.0/UDP hostA.example.com:5060
   From: A <sip:A@example.com>
   To: B <sip:B@example.com>
   Call-ID: 1234@hostA.example.com
   CSeq: 1 INVITE
   Contact: <sip:UserA@192.168.1.1>
   Content-Type: application/sdpng
   Content-Length: 685
```


```
<def>
 <audio:codec name="audio-basic" encoding="PCMU"
              sampling="8000" channels="1"/>

 <rtp:pt name="rtp-avp-0" pt="0">
  <audio:codec ref="audio-basic"/>
 </rtp:pt>
</def>

<cfg>
  <component name="interactive-audio" media="audio">
    <alt name="AVP-audio-0">
      <rtp:session>
       <rtp:pt ref="rtp-avp-0"/>
       <rtp:udp role="receive" endpoint="A" addr="192.168.1.1"
             rtp-port="7800"/>
      </rtp:session>
    </alt>
   </component>
</cfg>
```

```
<conf>
 <owner user="A@example.com" id="98765432" version="1" nettype="IN"
                     addrtype="IP4" addr="192.168.1.1"/>
 <session name="SDPng questions">
 </session>

 <info name="interactive-audio" function="voice">
  Telephony media stream
 <info>
</conf>
```

==

```
   F2 (100 Trying) B -> A

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP hostA.example.com:5060
   From: A <sip:A@example.com>
   To: B <sip:B@example.com>
   Call-ID: 1234@hostA.example.com
   CSeq: 1 INVITE
   Content-Length: 0
```

==

```
   F3 180 Ringing B -> A

   SIP/2.0 180 Ringing
   Via: SIP/2.0/UDP hostA.example.com:5060
   From: A <sip:A@example.com>
   To: B <sip:B@example.com>;tag=987654
   Call-ID: 1234@hostA.example.com
   CSeq: 1 INVITE
   Content-Length: 0
```

==

```
   F4 200 OK B -> A

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP hostA.example.com:5060
   From: A <sip:A@example.com>
   To: B <sip:B@example.com>;tag=987654
   Call-ID: 1234@hostA.example.com
   CSeq: 1 INVITE
   Contact: <sip:B@192.168.1.2>
   Content-Type: application/sdpng
   Content-Length: 479
```

```
<def>
 <audio:codec name="audio-basic" encoding="PCMU"
              sampling="8000" channels="1"/>

 <rtp:pt name="rtp-avp-0" pt="0">
  <audio:codec ref="audio-basic"/>
 </rtp:pt>
</def>

<cfg>
  <component name="interactive-audio" media="audio">
    <alt name="AVP-audio-0">
      <rtp:session>
       <rtp:pt ref="rtp-avp-0"/>
       <rtp:udp role="receive" endpoint="A" addr="192.168.1.1"
             rtp-port="7800"/>
       <rtp:udp role="receive" endpoint="B" addr="192.168.1.2"
             rtp-port="9410"/>
      </rtp:session>
    </alt>
   </component>
</cfg>
==================================================

ACK from A to B omitted
```

In the INVITE message, A sends B a description document, that specifies exactly one component with one alternative (the PCMU audio stream). All required transport parameters all already contained in the description. The rtp:udp element provides an attribute "role" with a value of "receive", indicating that the specified endpoint address is used by the endpoint to receive media data. The element also provides the attribute "endpoint" with a value of "A", denominating the endpoint that can receive data on the specified address. This means, the semantics of specified transport addresses in configuration descriptions are the same as for SDP (when used with SIP): An endpoint specifies where it wants to receive data.

In the 200 OK message, B sends an updated description document to A. For the sake of conciseness, the conf element (containing meta information about the conference) has been omitted. B supports the payload format that A has offered and adds his own transport

parameters to the configuration information, specifying the endpoint address where B wants to receive media data. In order to disambiguate its transport configurations from A's, B sets the attribute "endpoint" to the value "B". The specific value of the "endpoint" attribute is not important, the only requirements are that a party that contributes to the session description, must use a unique name for the endpoint attribute and that a contributing party must use the same value for the endpoint attributes of all elements it adds to the session description.

The following example shows a capability description that provides two alternatives for the audio component.

Description document with two alternatives:

```
   F1 INVITE A -> B

   INVITE sip:B@example.com SIP/2.0
   Via: SIP/2.0/UDP hostA.example.com:5060
   From: A <sip:A@example.com>
   To: B <sip:B@example.com>
   Call-ID: 1234@hostA.example.com
   CSeq: 1 INVITE
   Contact: <sip:UserA@192.168.1.1>
   Content-Type: application/sdpng
   Content-Length: 935

<def>
 <audio:codec name="audio-basic" encoding="PCMU"
              sampling="8000" channels="1"/>

 <audio:codec name="g729" encoding="G729" channels="1" sampling="8000"/>

 <rtp:pt name="rtp-avp-0" pt="0">
   <audio:codec ref="audio-basic"/>
 </rtp:pt>

 <rtp:pt name="rtp-avp-18" pt="18">
   <audio:codec ref="g729"/>
 </rtp:pt>

 <rtp:udp name="A-rcv" role="receive" endpoint="A" addr="192.168.1.1"
       rtp-port="7800"/>
</def>

<cfg>
  <component name="interactive-audio" media="audio">
    <alt name="AVP-audio-0">
```

```
        <rtp:session format="rtp-avp-0">
         <rtp:udp ref="A-rcv"/>
        </rtp:session>
      </alt>
      <alt name="AVP-audio-18">
        <rtp:session format="rtp-avp-18"/>
         <rtp:udp ref="A-rcv"/>
        </rtp:session>
      </alt>
     </component>
  </cfg>

  <conf>
   <owner user="A@example.com" id="98765432" version="1" nettype="IN"
                        addrtype="IP4" addr="192.168.1.1"/>
   <session name="SDPng questions">
   </session>

   <info name="interactive-audio" function="voice">
    Telephony media stream
   <info>
  </conf>

  ==================================================

     F2 (100 Trying) B -> A

     SIP/2.0 100 Trying
     Via: SIP/2.0/UDP hostA.example.com:5060
     From: A <sip:A@example.com>
     To: B <sip:B@example.com>
     Call-ID: 1234@hostA.example.com
     CSeq: 1 INVITE
     Content-Length: 0

  ==================================================

     F3 180 Ringing B -> A

     SIP/2.0 180 Ringing
     Via: SIP/2.0/UDP hostA.example.com:5060
     From: A <sip:A@example.com>
     To: B <sip:B@example.com>;tag=987654
     Call-ID: 1234@hostA.example.com
     CSeq: 1 INVITE
     Content-Length: 0

  ==================================================
```

F4 200 OK B -> A

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP hostA.example.com:5060
From: A <sip:A@example.com>
To: B <sip:B@example.com>;tag=987654
Call-ID: 1234@hostA.example.com
CSeq: 1 INVITE
Contact: <sip:B@192.168.1.2>
Content-Type: application/sdpng
Content-Length: 479
```

```
<def>
 <audio:codec name="audio-basic" encoding="PCMU"
              sampling="8000" channels="1"/>

 <audio:codec name="g729" encoding="G729" channels="1" sampling="8000"/>

 <rtp:pt name="rtp-avp-0" pt="0">
   <audio:codec ref="audio-basic"/>
 </rtp:pt>

 <rtp:pt name="rtp-avp-18" pt="18">
   <audio:codec ref="g729"/>
 </rtp:pt>

 <rtp:udp name="A-rcv" role="receive" endpoint="A" addr="192.168.1.1"
       rtp-port="7800"/>

 <rtp:udp name="B-rcv" role="receive" endpoint="B" addr="192.168.1.2"
       rtp-port="9410"/>
</def>

<cfg>
  <component name="interactive-audio" media="audio">
    <alt name="AVP-audio-0">
      <rtp:session format="rtp-avp-0">
        <rtp:udp ref="A-rcv"/>
        <rtp:udp ref="B-rcv"/>
     </rtp:session>
    </alt>
   </component>
</cfg>
==================================================
```

ACK from A to B omitted

In the INVITE message, A sends B a description document, that specifies one component with two alternatives for the audio stream (PCMU and G.729). Since A wants to use the same transport address for receiving media data regardless of the payload format, A provides the transport specification in the def element and references this definition in the rtp:session elements for both alternatives by using the attribute "transport".

In the 200 OK message, B sends an updated description document to A. B does only support PCMU, so it removes the alternative for G.729 from the description. B also defines its transport address in the def element and references this definition by referencing the rtp:udp element named "B-rcv".

A.3 Real-Time Streaming Protocol (RTSP)

In contrast to SIP, RTSP has, from its intended usage, a clear distinction between offering Potential Configurations (typically by the server) and choosing one out of these (by the client), and, in some cases; some parameters (such as multicast addresses) may be dictated by the server. Hence with RTSP, there is a clear distinguish between Potential Configurations during the negotiation phase and a finally chosen Actual Configuration according to which streaming will take place.

Example from the RTSP spec to be provided.

A.4 Media Gateway Control Protocol (MEGACOP)

The MEGACO architecture also follows the SDPng model of a clear separation between Potential and Actual Configurations. Upon startup, a Media Gateway (MG) will "register" with its Media Gateway Controller (MGC) and the latter will audit the MG for its Capabilities. Those will be provided as Potential Configurations, possibly with extensive Constraints specifications. Whenever a media path needs to be set up by the MGC between two MGs or an MG needs to be reconfigured internally, the MGC will use (updated) Actual Configurations.

Details and examples to be defined.

Appendix B. Change History

draft-ietf-mmusic-sdpng-06.txt

* Removed section on capability negotiation algorithm and section on formal specification. Added Section 3.

* Removed specification of concrete XML syntax from Section 4. Added requirements and theoretic considerations.

* Added clarification of term "actual configuration" in Section 2.

* Changed "profile" to "package".

* Added introducing text to Section 4.1.

* Added a list of terms with explanation at the end of Section 2.

* Removed audio and RTP packages from appendix.

* Added Section 5 ("Syntax Definition").

* Added Section 6 ("Specification of the Capability Negotiation").

draft-ietf-mmusic-sdpng-05.txt

* Moved audio and RTP packages to appendix.

* Moved section "Use of SDPng in conjunction with other IETF Signaling Protocols" to appendix.

* Changed mechanism for references to definitions: Definition elements provide an attribute "ref" that can be used to referenced existing definitions (Section 4.3). Removed other mechanisms for referencing (attributes "format" and "transport", element type "use").

* Corrections to schema definitions and examples

draft-ietf-mmusic-sdpng-04.txt

* New section on capability negotiation.

* New section on referencing definitions (Section 4.3).

* New section on properties.

* New section on definition groups.

draft-ietf-mmusic-sdpng-03.txt

* Extension of the SDPng schema (use of Xlinks etc.)

* Clarification in the text

* Fixed examples

* Added example libraries as appendices

* More details on usage with SIP, including examples.

draft-ietf-mmusic-sdpng-02.txt

* Added a section on formal specification mechanisms.

draft-ietf-mmusic-sdpng-01.txt

* renamed section "Syntax Proposal" to "Syntax Definition Mechanisms". More text on DTD vs. schema. Edited the example description.

* updated example definitions in section "Definitions" and "Components & Configurations"

* section "Session Attributes" replaces section "Session"

* new appendix on audio codec definitions

Full Copyright Statement

Acknowledgement

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