

mmusic
Internet-Draft
Expires: August 21, 2005

Kutscher
Ott
Bormann
TZI, Universitaet Bremen
February 20, 2005

Session Description and Capability Negotiation
draft-ietf-mmusic-sdpng-08.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [RFC 3668](#).

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

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

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

To view the list Internet-Draft Shadow Directories, see
<http://www.ietf.org/shadow.html>.

Copyright Notice

Copyright (C) The Internet Society (2005). All Rights Reserved.

Abstract

This document defines a language for describing multimedia sessions with respect to configuration parameters and capabilities of end-systems. The description language is independent of specific application scenarios (session announcement, session setup for interactive communication etc.) and is not limited to specific media types, capabilities, or configuration parameters.

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.

Internet-Draft

SDPng

February 2005

Document Revision

\$Revision: 6.21 \$

Table of Contents

1.	Introduction	3
2.	Terminology and System Model	7
3.	Overview	11
3.1	Outline of the Negotiation Process	11
3.2	SDPng Data Types	13
3.3	Application-specific Vocabulary	15
4.	SDPng Syntax	16
4.1	SDPng Base Syntax	16
4.2	Capabilities	18
4.2.1	Tokens	19
4.2.2	Token Sets	19
4.2.3	Numerical Values	19
4.2.4	Numerical Ranges	19
4.2.5	Sample SDPng cap Element	20
4.2.6	Referencing Capability Elements	21
4.3	Definitions	21
4.4	Configurations	23
4.5	Constraints	25
4.6	Session Information	25
4.7	Summary of SDPng XML-Syntax	26
5.	Usage of SDPng in Different Application Scenarios	28
5.1	Broadcast/Announcement Scenarios	28
5.2	Real-Time-Streaming	29
5.3	Two-Way Session Setup	31
5.4	Multi-Party-Conferencing with Negotiation	38
6.	IANA Considerations	39
7.	Open Issues	40
8.	Acknowledgements	41
	References	42
	Authors' Addresses	43
A.	Formal Syntax Specifications	44
A.1	SDPng Base DTD	44
A.2	SDPng XML-Schema Specification	45
B.	Sample Package Definitions	51
B.1	Sample RTP Package Definition	51
B.2	Sample Audio Package Definition	52

B.3	Sample Video Package Definition	52
C.	Sample SDPng Description	54
D.	Change History	58
	Intellectual Property and Copyright Statements	61

[1.](#) Introduction

Different applications in the multimedia realtime communication domain require a means for session description and capability negotiation. For example, for establishing an interactive audio communication session between two participants, end-systems must be dynamically configured with respect to codec types, codec parameters, transport protocol parameters and other configuration details. All these parameters can be viewed as the configuration for a session, and a session description language is used to describe this configuration unambiguously.

While the fundamental requirement to describe session parameters applies to all application scenarios, there are differences in how configurations are obtained and distributed among participants. For broadcast applications, e.g., sessions that are announced using SAP or IMG protocols, the sender is typically distributing a session description to potential receivers, describing the technical parameters (e.g., multicast addresses, port numbers, codecs etc.) and providing additional information about the session such as meta-information (about the content) and scheduling information. While a sender might provide alternative variants of one specific broadcast event (different language versions, different media codec configurations for different quality levels etc.) there is typically no negotiation process between sender and receiver in order to obtain the optimal configuration for a specific receiver. Instead, the sender may provide different potential configurations and the receiver would select the most appropriate one.

Real-time-streaming applications such as "video-on-demand" provide similar but slightly different requirements. Still the sender is typically providing a set of different potential configurations, out of which the receiver may select the most appropriate one, so there is not really a negotiation of content and its representation (on the session description level). But, differently to the aforementioned

scenarios, the receiver must tell the sender the endpoint address, i.e., where media data should be sent to. Depending on the control protocol, this may be specified using the session description language or other means (as it is done with RTSP).

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 spontaneously initiated conferences 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 conferences with a fixed set of participants) 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" was initially only made implicitly. While numerous extensions to SDP were developed to account for various aspects of interactive session establishment (the offer/answer model in [RFC 3264](#), grouping of media sessions in [RFC 3388](#), and simple capability declaration in [RFC 3407](#), among others), their expressiveness is naturally constrained by the simple (and very limited) SDP syntax.

SDPng, the session description languages specified in this document, provide a common, XML-based framework for session description and capability description for the aforementioned application scenarios. It allows for distributing "fixed" configuration descriptions in broadcast application scenarios but also supports the dynamic negotiation of session parameters for interactive multimedia

conferences.

In order to support a broad range of different applications, SDPng itself is completely application-agnostic. The base specification does not define any application-specific vocabulary, e.g., media types, codecs and their configuration parameters etc., but is intended as an extensible framework that provides the necessary extensibility mechanisms for supporting both future applications and future transport and encoding mechanisms.

With respect to capability negotiation, SDPng is based on the following principles:

Capability negotiation is an optional feature, which MAY be employed for specific applications, e.g., session setup for interactive communication.

SDPng provides the framework for describing capabilities and linking them to configurations of application sessions, but the base specification does not prescribe negotiation algorithms such as feature matching.

The SDPng base specification provides the necessary support for application-independent capability negotiation, i.e., an SDPng processor that receives capability descriptions from one or multiple participants does not need to understand the capability semantics in order to process the different capability descriptions. For this purpose, SDPng provides a well-defined set of data types and defines a value representation that allows SDPng processors to infer data types without requiring access to schema definitions and without requiring application-specific knowledge.

For most application classes, especially for broadcast applications, the session description will have to provide information about the session that is not used to configure end-systems but rather to facilitate content navigation for human users. For example, This meta-information can include author/source details, additional information about the whole session or individual media sessions, scheduling information and other, arbitrary information that is related to the session but not directly required to achieve interoperability between end-systems. SDPng provides fundamental mechanisms that support the inclusion of meta-information:

Descriptions of application components (media sessions) and alternative configurations can be labeled to enable later referencing, i.e., for associating meta-information. For example, in a multi-language TV broadcast session, the language information could be provided by a meta-information fragment that assigns

language tags to media session descriptions by referencing them.

SDPng itself does not define vocabulary for specifying meta-information, but allows for including arbitrary meta-information fragments, e.g., MPEG-7 descriptions. These description may either be included or inline or be referenced by URIs.

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](#). [Section 5](#) describes how SDPng

can be used in different application scenarios.

[Appendix A](#) provide formal specifications of SDPng such as XML DTD and Schema definitions, [Appendix B](#) provides some sample package definitions, [Appendix C](#) provides a sample SDPng description, and [Appendix D](#) 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, available licenses, and quality of implementation may constrain the possible video encoding algorithms.

In multiparty multimedia conferences, participants employ different "components" in conducting the conference; similarly, a multimedia (rather than plain TV) broadcast may comprise several components that make up the full presentation.

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 policy 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:

- o 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;
- o 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:

- o A multimedia session (broadcast, streaming or conference) consists of one or more conference components for multimedia "interaction".
- o 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).
- o A configuration is a set of parameters that are required to implement a certain variation (realization) of a certain

Internet-Draft

SDPng

February 2005

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 [\[8\]](#) 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 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).

Package

A package is an 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. Overview](#)

SDPng is a description language for both potential configurations (i.e. capabilities) of participants in multimedia conference 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 itself is an application-independent framework that defines a description syntax that enable an (optional) capability negotiation process. It should be noted, that SDPng can be used without capability negotiation and that the specific negotiation algorithm is not specified in this document.

A capability description for an endpoint is a set of individual capabilities, each of which provides a fixed type, e.g., a numeric value or a list value. The set of types and the corresponding abstract negotiation rules are defined in this memo. The SDPng data types are relevant to all SDPng application domains, while the processing rules are only applicable to application domains that rely on capability negotiation.

In the following, we provide a conceptual overview of the negotiation process in [Section 3.1](#) and describe the different capability types and the corresponding abstract negotiation rules in [Section 3.2](#).

[3.1 Outline of the Negotiation Process](#)

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 [\[13\]](#) it must be possible to determine an actual configuration in a single request/response cycle.

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 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 " $CAP(A) \times CAP(B) = CAP(AB)$ ".
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 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 B.
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.

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, therefore they have to be distinguished from other configuration information.

Note also that, in case of multicast/broadcast scenarios, the sender may simply provide the full description of the alternative configurations including (transport) parameters. The potential receivers will decide based upon this information whether or not they are capable of receiving the respective media sessions and pick the most suitable configuration.

[3.2](#) SDPng Data Types

The description of actual configurations and the capability negotiation process rely on a fixed set of data types with corresponding processing rules. The following types are defined:

1. Tokens (text strings)

Example:

`<audio:encoding>PCMU</audio:encoding>`

Processing rule:

Ascertain identity (compare strings)

2. Token lists

Example:

`<audio:sampling-rate>8000 16000</audio:sampling-rate>`

Processing rule:

Determine common subset

3. Numbers

Example:

```
<audio:bitrate val="64"/>
```

Processing rule:
Ascertain equality

4. Numerical ranges

Example:

```
<audio:bitrate min="6" max="64"/>
```

Processing rule:
Determine common subrange

SDPng distinguishes between optional and mandatory capability definitions, with different processing rules for the negotiation process. Optional definitions are used for capabilities that can be provided by an entity but do not have to be supported by all participants. For example, an audio codec could provide optional codec parameters. The use of these parameters needs to be declared by a session description, but if the parameter is not understood by all implementations, a session can be established nevertheless. As a result, the failure of a single processing step for a definition that has been marked as "optional" does not lead to a failure of the capability negotiation as a whole.

A mandatory capability on the other hand has to be supported by all participants. For example, the specification of an audio codec for an audio capability is mandatory, and for obtaining an interoperable configuration, all participants must support the same audio codec or set of audio codecs.

In addition to capabilities, a SDPng description can also provide parameters that are not negotiable, e.g., transport parameters. In SDPng, there is a distinction between capability definitions (that

are subject to a negotiation process) and parameters that are

specified by each participant. In a description of alternative configurations for a specific component, capabilities and parameters can be referred to and describe the configurations.

[3.3](#) Application-specific Vocabulary

While the SDPng specification defines the fundamental types, abstract processing rules and the syntax definition for SDPng descriptions, it does not define any application-specific vocabulary.

Application-specific vocabulary is defined in SDPng packages. An SDPng package defines a schema for application specific capability and parameter descriptions. Based on the description types specified by the SDPng base specification, a package definition specifies the capability and parameter definitions allowed for a specific application, the types of definitions and additional attributes, e.g., whether a definition element is optional with respect to the capability negotiation or not.

The SDPng base specification does define some fundamental requirements for definition elements that are specified in package definitions, for example XML attributes for elements. [Appendix A.2](#) provides an XML Schema definition that specifies some base types to be used for package definitions.

In order to allow for an application independent processing of SDPng description documents, SDPng descriptions are standalone, i.e., the package definition is not required to process a corresponding SDPng document. All information, e.g., the type of definitions and additional attributes are contained in the SDPng document itself. An SDPng implementation can thus be processed without access to the package definition.

4. SDPng Syntax

This section specifies the SDPng base syntax. An SDPng description is an XML document consisting of up to five parts:

Capabilities

Definitions

Configurations

Constraints

Session Information

The Capabilities section provides a list of individual capabilities. In a capability negotiation process, these capabilities are matched against corresponding definitions of other participants' capability descriptions. This section is OPTIONAL for a SDPng description document.

The Definitions section provides definitions of commonly used parameters for later referencing. This section is OPTIONAL for SDPng descriptions.

The Configurations section provides the description of the different conference components (applications in a conference). Each component description can provide a list of alternative configurations. This section MUST be present in any SDPng description.

The Constraints section provides constraints on combinations of configurations. This section is OPTIONAL for SDPng descriptions.

The Session Information section provides meta information on the conferences and on individual components. This section is OPTIONAL for SDPng documents.

4.1 SDPng Base Syntax

An SDPng description is an XML document. The document root element MUST be an element of type "sdpng". The XML vocabulary for the SDPng base specification resides in the XML namespace "http://www.iana.org/sdpng". The root element of an SDPng description MUST define an XML default namespace "http://www.iana.org/sdpng". In addition, the "sdpng" element MUST map the namespace prefix "sdpng" to the namespace name "http://www.iana.org/sdpng". The "sdpng" element type

provides the child elements "cap", "def", "cfg", "constraints", and "info" for the different sections of the SDPng description. The

default namespace is also applied to these elements.

The encoding of the XML document MUST be UTF-8 ([RFC2279](#), [16]).

The following figure depicts the overall SDPng document structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng">
  <cap>
    [...]
  </cap>
  <def>
    [...]
  </def>
  <cfg>
    [...]
  </cfg>
  <constraints>
    [...]
  </constraints>
  <info>
    [...]
  </info>
</sdpng>
```

[Appendix A.1](#) provides a XML DTD that defines a corresponding document type.

Note that the elements for the optional sections "Capabilities", "Definitions", "Constraints", and "Session-Level Information" are OPTIONAL.

Application-specific vocabulary resides in its own namespace. For each namespace name of an SDPng package, a namespace prefix MUST be declared in the start tag of the "sdpng" element. The following figure depicts the declaration of namespace prefixes for two package namespaces:

```

<?xml version="1.0" encoding="UTF-8"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng"
      xmlns:rtp="http://www.iana.org/sdpng/rtp"
      xmlns:audio="http://www.iana.org/sdpng/audio">
  [...]
</sdpng>

```

[4.2](#) Capabilities

A section for capability descriptions is an XML element that can provide a list of child elements. The element type is called "cap"(in the "sdpng" namespace). Each child element represents an individual capability.

Each capability 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, e.g., for the participant of a conference. If a prefix is specified, it MUST be separated by a colon (':') from the name. The namespace MUST be declared in the respective element or in ancestor elements, e.g., the root "sdpng" element. The following figure depicts a capability element inside a "cap" element. Note that the child elements of "audio:codec" and the other sections of the SDPng description are not shown.

```

<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng">
  <cap>
    <audio:codec name="avp:pcmu">
      [One or more feature elements]
    </audio:codec>

    [...]
  </cap>
</sdpng>

```

Each capability element provides a set of features. Each feature is

represented by a child element. The element types are defined in package definitions. XML Namespaces are used to disambiguate element types and to allow for extensibility. Each feature element can provide a "range" of values -- not only a single value. For example, a feature element 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": A feature element can specify a set of tokens or a numerical range.

Each feature that is represented by an XML feature element has a well-defined type that is specified in the package definition. The type determines the representation of the element values in XML so that type information is encoded implicitly in the description document. For example, a token set can be identified by the presence of whitespace separated list of token as element content of the respective feature element. Each feature element MAY provide an

attribute "status". If this attribute is present it MUST provide one of the following values:

opt:

This element describes an optional feature (as described by [Section 3.2](#)).

The three different features types (as described in [Section 3.2](#)) are represented as described in the following sections. [Section 4.2.5](#) provides a complete example.

[4.2.1](#) Tokens

Token elements provide a single token as element content. The token is of type Nmtoken (name token) as defined by [9]. The following example depicts a feature element of type token.

```
<audio:encoding>PCMU</audio:encoding>
```

Boolean values SHOULD be represented as token elements with a values of either "true" or "false".

[4.2.2](#) Token Sets

Token set elements provide a token list as element content. The token

is of type Nmtokens (name tokens) as defined by [9]. The following example depicts a feature element of type token set.

```
<audio:sampling>8000 16000 32000</audio:sampling>
```

[4.2.3 Numerical Values](#)

Elements for numbers provide an attribute "val" with a numerical value. The following example depicts a feature element of type numerical value.

```
<audio:bitrate val="64"/>
```

[4.2.4 Numerical Ranges](#)

Elements for numerical ranges can provide an attribute "min" and an attribute "max". Both attributes provide a numerical value. At least one of these attributes MUST be present. The following example depicts a feature element of type numerical range.

```
<audio:bitrate min="6" max="64"/>
```

[4.2.5 Sample SDPng cap Element](#)

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
  xmlns:sdpng="http://www.iana.org/sdpng">
  <cap>
    <audio:codec name="avp:pcmu">
      <audio:encoding>PCMU</audio:encoding>
      <audio:channels>1 2</audio:channels>
      <audio:sampling>8000 16000</audio:sampling>
      <audio:bitrate min="6" max="64"/>
      <audio:silence-suppression status="opt">
        true
      </audio:silence-suppression>
    </audio:codec>
```

[...]

```
</cap>
</sdpng>
```

Capability elements MAY also provide elements from different XML namespaces. For example, a video-codec capability MAY be described with elements declaring general video capabilities, and this element MAY provide a list of additional codec specific feature elements, as depicted in the following example:

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
       xmlns:sdpng="http://www.iana.org/sdpng">
  <cap>
    <video:codec name="h263+-enhanced">
      <video:encoding>H.263+</video:encoding>
      <video:resolution>QCIF</video:resolution>
      <video:framerate max="30"/>
      <h263plus:A>foo</h263plus:A>
      <h263plus:B>bar</h263plus:B>
    </video:codec>

    [...]
  </cap>
</sdpng>
```

[4.2.6](#) Referencing Capability Elements

The capability elements of a "cap" element can be referenced in later sections of the SDPng document. The fundamental model is that capability elements specify individual capabilities (without transport and other non-negotiable parameters) and that these elements are later augmented in Definitions and Configurations sections.

When referencing a capability element, e.g., the element video:codec, the same element name (general identifier) is used. The referencing

element MUST provide an attribute "ref", and the value of this attribute SHOULD provide the value of the attribute "name" of the referenced element.

The referencing element MAY also provide additional feature elements (that have not been provided by the referenced capability element). The referencing element MAY also provide feature elements that have already been provided by the referenced element.

The referencing element MAY provide an attribute "name". The semantics of a reference are defined in the corresponding sections where references to definitions are used, i.e., in [Section 4.3](#) and in [Section 4.4](#).

[4.3](#) Definitions

The Definitions section is an optional section that can provide definitions of fixed parameters that are not negotiable such as transport parameters. An SDPng description document MAY provide a "def" element that can provide a set of definitions as child elements.

Each child element of a "def" element provides an element type specified in a package definition. Such child elements are referred to as "definition elements". Definition elements can provide a set of child elements, each of which specifies a specific configuration value. Syntactically, these child elements MUST be "feature elements" as specified in [Section 4.2](#). Child elements of a definition element MUST be of type Token or of type Numerical Value. A definition element MUST provide an attribute "name" that is used to specify a unique name in the scope of the current SDPng description. A definition element MAY provide an attribute "ref" that is used to reference a capability element as specified in [Section 4.2](#).

The following example depicts a def element with one definition element of type "rtp:udp". This element is used to specify fixed parameters of an RTP session -- the allowable parameters would have

been specified in a corresponding SDPng RTP package.

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
```



```

xmlns:sdpng="http://www.iana.org/sdpng">

<cap>
  <audio:codec name="avp:pcmu">[...]</audio:codec>
  <rtp:udp name="rtpudpip6">[...]</rtp:udp>
</cap>

<def>
  <rtp:udp name="rtp-cfg1" ref="rtp:rtpudpip6">
    <rtp:ip-addr>::1</rtp:ip-addr>
    <rtp:rtp-port>9456</rtp:rtp-port>
    <rtp:pt>1</rtp:pt>
  </rtp:udp>
</def>

<cfg>
  [...]
</cfg>
<constraints>
  [...]
</constraints>
<info>
  [...]
</info>

</sdpng>

```

A definition element SHOULD reference a capability element provided in the "cap" element, as depicted in the example. In the example, the definition named "rtp-cfg1" provides RTP transport parameters and references the RTP capability named "rtp:rtpudpip6". The semantics of referencing the capability element are as follows:

- o An implementation MUST process the newly defined element by adopting the individual feature elements of the referenced capability element.
- o For feature elements that are present in both the capability element and the description element, the feature elements of the definition element take precedence over the feature elements of the capability element.

[4.4](#) Configurations

The 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.

An SDPng document MUST provide a "cfg" element that represents the Configurations section. The "cfg" element provides one or more "component" element describing alternative configurations for the component. The "cfg" element SHOULD provide at least one "component" element. Each "component" element MUST provide an attribute "name" that identifies the component uniquely in the scope of the SDPng description. Each "component" element MAY provide an attribute status of type CDATA that MAY be used to specify application specific status information.

Each "component" element MUST provide one or more "alt" element, each of which describes an alternative configuration for the component. Each "alt" element MUST provide an attribute "name" that provides a unique identification for the alternative in the scope of the SDPng description. In addition, each "alt" element MUST also provide an attribute "media" for specifying the media type for this particular alternative. Currently defined values for this attribute are "audio", "video", "application", "data", and "control". The semantics of these values are described in [\[2\]](#).

Each "alt" element MUST provide one or more XML elements that describe the configuration parameters for the particular alternative configuration. The elements are defined by SDPng package specification and definitions from different packages can be mixed. The type of the elements and their order is application dependent.

Each definition element that is contained in an "alt" element MAY provide an attribute "ref". The "ref" attribute is used to specify a reference to a capability element (from a "cap" section) or to a definition element (from a "def" section). The value of an "ref" element MUST provide the value of a "name" attribute of an existing capability or definition element. A definition element MAY provide child elements (for the specification of additional feature and configuration parameters) but it MAY also be an empty element. The semantics of referencing the capability element are as follows:

- o An implementation MUST process the newly defined element by adopting the individual feature elements of the referenced capability or definition element.

- o For feature elements that are present in both the capability/

definition element and the current definition element, the feature elements of the current definition element take precedence over the feature elements of the referenced element.

It should be noted that the inclusion of definition by reference is NOT MANDATORY. For certain applications such as session announcement, it may be sufficient to provide the configuration data directly inside an "alt" element.

The following example depicts the description of a single configuration for a component named "interactive-audio". The description of the configuration references the "avp:pcmu" audio codec definition from the "cap" element and the "rtp-cfg1" RTP session definition from the "def" element. In this example, both elements of the "alt" element are empty elements that adopt the specified values from the referenced elements.

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng">

  <cap>
    <audio:codec name="avp:pcmu">[...]</audio:codec>
    <rtp:udp name="rtpudpip6">[...]</rtp:udp>
  </cap>

  <def>
    <rtp:udp name="rtp-cfg1" ref="rtp:rtpudpip6">
      <rtp:ip-addr>::1</rtp:ip-addr>
      <rtp:rtp-port>9456</rtp:rtp-port>
      <rtp:pt>1</rtp:pt>
    </rtp:udp>
  </def>

  <cfg>
    <component name="interactive-audio" media="audio" status="active">
      <alt name="alt1">
        <audio:codec ref="avp:pcmu"/>
        <rtp:udp ref="rtp-cfg1"/>
      </alt>
    </component>
  </cfg>
</sdpng>
```

```
</alt>
</component>

</cfg>
<constraints>
  [...]
</constraints>
<info>
```

```
  [...]
</info>

</sdpng>
```

[4.5](#) Constraints

The Constraints section allows to express constraints on the combination of configurations that apply across different components. This feature is intended for specialized devices with strict limitations on, e.g., parallel codec instantiations due to limited DSP resources. The SDPng base specification does not define the use of constraints. Instead, the usage of constraints will be specified in a separate document.

The "constraints" element of an SDPng description is OPTIONAL.

[4.6](#) Session Information

The Session Information section is represented by an "info" element and is intended for meta information on the conference itself and on the individual components. In an SDPng description document, the "info" element MAY provide one or multiple "part" element, each of which may contain arbitrary meta-description content.

The "info" element is OPTIONAL in an SDPng description document.

Each "part" element inside an "info" element MUST provide a "type" attribute that specifies the type of the meta-information content, e.g., "MPEG-7", "TV-Anytime", "MPEG-21", "SDP".

Each "part" element inside an "info" element MAY provide a "ref" attribute that can be used to specify an URI for the meta-information content. If a "ref" attribute is provided the "part" element MUST be empty.

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng">

  <cap>
    <audio:codec name="avp:pcmu">[...]</audio:codec>
    <rtp:udp name="rtpudpip6">[...]</rtp:udp>
  </cap>
```

```
<def>
  <rtp:udp name="rtp-cfg1" ref="rtp:rtpudpip6">
    <rtp:ip-addr>::1</rtp:ip-addr>
    <rtp:rtp-port>9456</rtp:rtp-port>
    <rtp:pt>1</rtp:pt>
  </rtp:udp>
</def>

<cfg>
  <component name="interactive-audio" media="audio" status="active">
    <alt name="alt1">
      <audio:codec ref="avp:pcmu"/>
      <rtp:udp ref="rtp-cfg1"/>
    </alt>
  </component>
</cfg>
<constraints>
  [...]
</constraints>
<info>
  <part type="SDP">
    o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
    s=SDP Seminar
    i=A Seminar on the session description protocol
    u=http://www.example.com/seminars/sdp.pdf
```

```
e=j.doe@example.com (Jane Doe)
t=2873397496 2873404696
</part>
</info>

</sdpng>
```

[4.7](#) Summary of SDPng XML-Syntax

The SDPng base specification defines the following XML element types that reside in the SDPng namespace designated by the namespace name "http://www.iana.org/sdpng":

- o sdpng
- o cap
- o def
- o cfg

- o component
- o alt
- o constraints
- o info

[Appendix A.1](#) provides an XML DTD that specifies the content model of the SDPng base elements.

[5.](#) Usage of SDPng in Different Application Scenarios

This informative section describes how SDPng can be used in different application scenarios, namely broadcast/announcement, real-time-streaming, two-way-session-setup, and multiparty conferencing with negotiation.

[5.1](#) Broadcast/Announcement Scenarios

Broadcast and multicast application scenarios rely on session announcements to communicate information about multimedia sessions and their associated parameters. Historically, in the Mbone, the

Session Announcement Protocol (SAP) [RF2974] was used to convey static session descriptions via multicast but the same information could also be advertized by means of email, NetNews, or web pages. For some years, digital television used Electronic Program Guides (EPGs) to convey programming information (movie schedule, metadata, channel information) to large audiences. More recently, streaming services for (3G and beyond) mobile networks make use of similar concepts to announce streaming services. As a response to these needs, the generalized framework of Internet Media Guides (IMGs) has been devised to address conveying scheduling and media information to potentially large receiver groups. This subsection addresses the use of SDPng with SAP and IMG-based session announcements.

SAP and IMGs are 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 communicated via SAP or IMGs to join the announced media sessions. While this is supported by MIME types identifying SDPng contents with implied semantics, the IMG framework explicitly suggests interactive retrieval using HTTP. Furthermore, IMG has the concept of asynchronous notifications/updates to existing SDPng descriptions: it makes use of a (SIP-based) notification mechanism that allows interested parties to monitor the state of session descriptions and receive asynchronous updates whenever the description changes.

SAP makes extensive use of the SDP session level attributes to provide a (limited) set of descriptive metadata for the session, including scheduling and subject information. Quite a bit of this information is application-specific and is therefore not defined in the baseline SDPng spec. In particular, IMGs are expected to be used with more encompassing metadata description formats (such as MPEG-7) which will carry the respective information so that these need not be replicated in SDPng itself.

While SAP only supports full SDP descriptions (which are minimal in size), IMGs allows in addition to the (potentially large)

(collections of) SDPng descriptions and associated metadata to carry delta information or pointers to contents (URIs). The structured format of the XML-based SDP goes well with both concepts and allows to identify subparts of SDPng messages where and perform operations on them via well-defined means.

[5.2](#) Real-Time-Streaming

Real-time streaming provides an interactive way to offer real-time media to users. In contrast to the announcement/broadcast based described in the previous section, where the communication is mostly unidirectional and interested receivers just "tune" into the respective media session, with RTSP an interactive session setup exists. This also informs the media server that there are parties interested in a particular media stream and provides the opportunity to the client to obtain the most appropriate variant of a media stream.

Similar to SAP and IMGs, RTSP has, from its intended usage, a clear distinction between offering a set of Potential Configurations (by the server) and choosing one out of these (by the client). There is no capability negotiation process involved: the server provides a complete SDPng document describing all Components making up a presentation and includes detailed codec and transport parameters for each of there. The client may only pick one out of alternatives for each of the offered Components but has no further option to negotiate parameters in depth. Where some additional exchange is necessary -- e.g. for the client's transport addresses and security parameters --, the respective parameters are no encoded in SDPng; instead, additional RTSP header fields and parameters are field for this purpose.

Hence, SDPng is only used to describe alternatives to gain access to streaming media out of which the client has to choose. No interaction takes place at the SDPng level.

It should be noted that SAP or IMG-based announcements may also be used to point users to RTSP servers. In such a case, the "negotiation" is a two-stage process: the media discovery takes place using SAP or IMG and leads to the RTSP server. The actual streaming invocation process then takes place interactively between the RTSP client and server as described in this section.

```
C->M: DESCRIBE rtsp://foo/audio-play RTSP/1.0
      CSeq: 1
```

```
M->C: RTSP/1.0 200 OK
      CSeq: 1
```

Content-Type: application/sdp
Content-Length: ...

```
<sdpng xmlns="http://www.iana.org/sdpng"
  xmlns:audio="http://www.iana.org/sdpng/audio"
  xmlns:rtp="http://www.iana.org/sdpng/rtp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.iana.org/sdpng sdpng-base.xsd
http://www.iana.org/sdpng/audio sdpng-audio-pkg.xsd
http://www.iana.org/sdpng/rtp sdpng-rtp-pkg.xsd"

  owner="A@example.com" id="98765432" version="1"
>

<cap>
  <audio:codec name="avp:pcmu">
    <audio:encoding>PCMU</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
  </audio:codec>
  <audio:codec name="avp:gsm">
    <audio:encoding>GSM</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
  </audio:codec>
</cap>
<def>
  <rtp:udp name="rtp-cfg1" ref="rtpudp4">
    <rtp:ip-addr>192.168.47.11</rtp:ip-addr>
    <rtp:rtp-port>51400</rtp:rtp-port>
  </rtp:udp>
</def>
<cfg>
  <component>
    <alt>
      <audio:codec ref="avp:pcmu"/>
      <rtp:udp ref="rtp-cfg1">
        <rtp:pt>0</rtp:pt>
      </rtp:udp>
    </alt>
    <alt>
      <audio:codec ref="avp:gsm"/>
      <rtp:udp ref="rtp-cfg1">
        <rtp:pt>3</rtp:pt>
      </rtp:udp>
    </alt>
  </component>
</cfg>
```

Internet-Draft

SDPng

February 2005

</sdpng>

```
C->M: SETUP rtsp://foo/audio-play RTSP/1.0
      CSeq: 2
      Transport: RTP/AVP;unicast;client_port=8000-8001

M->C: RTSP/1.0 200 OK
      CSeq: 2
      Transport: RTP/AVP;unicast;client_port=8000-8001;
                server_port=51400-51401
      Session: 12345678
```

To be continued with PLAY and, after the audio track has completed, finished with TEARDOWN.

[5.3](#) Two-Way Session Setup

The major application of SDP today is its use with the Session Initiation Protocol (SIP) [[RFC3261](#)]. Session descriptions are used to configure the media streams between two parties involved in a multimedia call.

SIP is used to establish and modify multimedia sessions, and SDPng may be carried at least in SIP INVITE, ACK and UPDATE 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, generalized in the offer/answer model documented in [RFC 3264](#).

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 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 distinguish between Potential and Actual Configurations at the syntax, this has no implications on the SDPng signaling itself but is merely up to interpretation.

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 corresponding 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
```

<?xml version="1.0" encoding="UTF-8"?>

```

<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:audio="http://www.iana.org/sdpng/audio"
      xmlns:rtp="http://www.iana.org/sdpng/rtp"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.iana.org/sdpng sdpng-base.xsd
http://www.iana.org/sdpng/audio sdpng-audio-pkg.xsd
http://www.iana.org/sdpng/rtp sdpng-rtp-pkg.xsd"

  owner="A@example.com" id="98765432" version="1"
>

<cap>
  <audio:codec name="avp:pcmu">

```

```

      <audio:encoding>PCMU</audio:encoding>
      <audio:channels>1</audio:channels>
      <audio:sampling>8000</audio:sampling>
    </audio:codec>
    <audio:codec name="avp:gsm">
      <audio:encoding>GSM</audio:encoding>
      <audio:channels>1</audio:channels>
      <audio:sampling>8000</audio:sampling>
    </audio:codec>
  </cap>
  <def>
    <rtp:udp name="rtp-cfg1" ref="rtpudpip4">
      <rtp:ip-addr>192.168.47.11</rtp:ip-addr>
      <rtp:rtp-port>51400</rtp:rtp-port>
    </rtp:udp>
  </def>
  <cfg>
    <component>
      <alt>
        <audio:codec ref="avp:pcmu"/>
        <rtp:udp ref="rtp-cfg1">
          <rtp:pt>0</rtp:pt>
        </rtp:udp>
      </alt>
      <alt>
        <audio:codec ref="avp:gsm"/>
        <rtp:udp ref="rtp-cfg1">
          <rtp:pt>3</rtp:pt>

```

```

                                </rtp:udp>
                        </alt>
                </component>
        </cfg>
</sdpng>

```

=====

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

```
<?xml version="1.0" encoding="UTF-8"?>

<sdpng xmlns="http://www.iana.org/sdpng"
  xmlns:audio="http://www.iana.org/sdpng/audio"
  xmlns:rtp="http://www.iana.org/sdpng/rtp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.iana.org/sdpng sdpng-base.xsd
http://www.iana.org/sdpng/audio sdpng-audio-pkg.xsd
http://www.iana.org/sdpng/rtp sdpng-rtp-pkg.xsd"

  owner="B@example.com" id="4595647" version="1"
>

<cap>
  <audio:codec name="avp:pcmu">
    <audio:encoding>PCMU</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
  </audio:codec>
  <audio:codec name="avp:gsm">
    <audio:encoding>GSM</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
  </audio:codec>
```

```
</cap>
<def>
  <rtp:udp name="rtp-cfg1" ref="rtpudp4">
    <rtp:ip-addr>192.168.47.12</rtp:ip-addr>
    <rtp:rtp-port>60006</rtp:rtp-port>
  </rtp:udp>
</def>
<cfg>
  <component>
    <alt>
      <audio:codec ref="avp:gsm"/>
      <rtp:udp ref="rtp-cfg1">
        <rtp:pt>3</rtp:pt>
      </rtp:udp>
```

```
                </alt>
            </component>
    </cfg>
</sdpng>
```

=====

ACK from A to B omitted

In the INVITE message, A sends B a description document that specifies exactly one component with two alternatives (the PCMU and GSM audio streams). The alternatives make reference to the capability section where the two codec types are defined. All required transport parameters are already contained in the respective descriptions but they are kept separate from the capability part. The <def> element contains a definition for the RTP media sessions so that this needs not be repeated in the configuration of the single component. Note that the semantics of the component is not explicitly specified (in an <info> element) but rather implied.

In the 200 OK message, B sends an updated description document to A. 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 offer/answer model allows 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 applicable to realize these components. The answerer indicates which components it supports

(e.g. receiving a offer including audio and video, it may disallow the video session and go with an 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 for each of these ("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 basic defined in [RFC 3264](#) for generating offers and answers apply. The following considerations specifically apply when using offer/answer with SDPng (instead of SDP) documents:

- o For each component to be used, all necessary parameters MUST be given for at least one configuration per component, i.e. transport addresses and payload formats MUST be specified along with the capabilities.
- o Matching of components is done based upon their identification in the info part of the SDPng document using predefined identifiers for certain session types.
For simple sessions, where applications can implicitly derive the semantics of the the offered components, no such explicit mapping is necessary. In this case, i.e. if the entire "<info>" element or the respective elements in the "<info>" element are absent, the order of appearance in the SDPng document is relevant as it is with SDP.
- o For each component, the answerer performs a capability matching process as per then application's requirements
For all components that are acceptable, the answerer determines whether or not to accept the offer.
If the answerer decides to accept the offer for a certain component, it MUST accept at least one of the potential

configurations for the respective component. It SHOULD indicate this by setting the "status" attribute of the component and of the selected configuration(s) to "active" (but it MAY also omit the status attribute in both cases).

It is RECOMMENDED that the answerer selects exactly one configuration for each component as "active".

- o The answerer MAY refuse individual configurations for a component from the offer in two ways.
If the configuration shall not be used at all during a session, e.g. because the answerer does not support it or because the answerer does not want to use this configuration at all, the answerer MUST set the "status" attribute of the respective component to "unused". In this case, the answerer MAY omit all the elements contained in the respective configuration's elements. This is equivalent to setting the port parameter to "0" in SDP. If a configuration shall be accepted (i.e. the respective capability shall be indicated) but no media session shall be instantiated (not even on hold!), the answerer MUST set the "status" attribute of the respective configuration to "available" and omit all media-session-specific parameters the configuration.
- o The answerer MAY refuse entire components that the offerer has included in two ways.
If a component shall not be used at all during a session -- e.g. because the answerer does not support any of the configurations listed or because the answerer does not want to use this component at all -- the answerer MUST set the "status" attribute of the respective component's to "unused". In this case, the answerer MAY omit all the elements contained in the respective component elements. This is equivalent to setting the port parameter to "0" in SDP.
If a component shall be accepted (i.e. the respective capability shall be indicated) but no media session shall be instantiated (not even on hold!), the answerer MUST set the "status" attribute of the respective component to "available", omit all media-session-specific parameters from all acceptable configurations for the respective component.
- o For each component, the alternative potential configurations MUST be listed in the order of preference.
Within a configuration, alternatives (e.g. different codecs) MUST also be listed in the order of preference.
The considerations of [RFC 3264](#) to simply arriving at symmetric codec use apply.

If a component shall be put on hold, the status attribute of the component MUST be set to "sendonly", "recvonly", or "inactive", as

Internet-Draft

SDPng

February 2005

appropriate. In this case, the status attributes of all the contained configurations that were previously active MUST be set to indicate "sendonly", "recvonly", or "inactive", as appropriate. The rules from [RFC 3264](#) for putting media streams on hold SHALL apply.

[5.4](#) Multi-Party-Conferencing with Negotiation

The independent capability collapsing properties of SDPng allow to calculate the "intersection" of any number of SDPng documents independently and easily derive a common subset. Details are TBD in a separate document.

Internet-Draft

SDPng

February 2005

6. IANA Considerations

The IANA should set up a registry for XML namespaces for SDPng and SDPng package definitions.

The SDP parameter registry (<http://www.iana.org/assignments/sdp-parameters>) should be converted to SDPng package definitions.

7. Open Issues

Meta-Information for SDPng description (author, version etc.)?

Revise usage of terminology (potential configuration, actual configuration)

Do we need an explicit mechanism to declare the used packages?
E.g., <pkg ref="http://www.iana.org/sdpng/audio"/>

Data model for audio package: sampling-rate vs. RTP clock rate

Bib. references: distinguish normative and informational

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

[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. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", [RFC 3550](#), July 2003.
- [4] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", [RFC 3551](#), July 2003.

- [5] 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.
- [6] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", [RFC 2733](#), December 1999.
- [7] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", [RFC 2354](#), June 1998.
- [8] Handley, M., Perkins, C. and E. Whelan, "Session Announcement Protocol", [RFC 2974](#), October 2000.
- [9] 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.
- [10] World Wide Web Consortium (W3C), "Namespaces in XML", Status W3C Recommendation, Version <http://www.w3.org/TR/1999/REC-xml-names-19990114>, January 1999.
- [11] 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.
- [12] 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.
- [13] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with SDP", [RFC 3264](#), June 2002.

- [14] Hollenbeck, S., Rose, M. and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", [BCP 70](#), [RFC 3470](#), January 2003.
- [15] Klyne, G., "A Syntax for Describing Media Feature Sets", [RFC 2533](#), March 1999.
- [16] Yergeau, F., "UTF-8, a transformation format of ISO 10646", [RFC](#)

[2279](#), January 1998.

- [17] Holtman, K., Mutz, A. and T. Hardie, "Media Feature Tag Registration Procedure", [BCP 31](#), [RFC 2506](#), 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

[A.1](#) SDPng Base DTD

The following DTD specifies the SDPng base syntax. DTDs are not XML-Namespace aware, therefore the following DTD is for informational purposes only. Moreover, the content models for the element types "cap" and "def" have to be empty in this DTD as the specific element types for the allowed child elements are not defined by the base specification but by independent package definitions. Common requirements for these element types such as the "name" attribute cannot be expressed with XML DTDs.

```
<!ELEMENT sdpng (cap?, def?, cfg, constraints?, info?)>
```

```
<!ELEMENT cap ANY>
```

```
<!ELEMENT def ANY>
```

```
<!ELEMENT cfg (component+)>
```

```
<!ELEMENT component (alt+)>
```

```
<!ATTLIST component
  name    CDATA #REQUIRED
  media   CDATA #IMPLIED
  status  CDATA #IMPLIED
>
```

```
<!ELEMENT alt ANY>
```

```
<!ATTLIST alt
  name CDATA    #REQUIRED
  status CDATA #IMPLIED
>
```

```
<!ELEMENT constraints ANY>
```

```
<!ELEMENT info (part+)>
```

```
<!ELEMENT part ANY>
```

```
<!ATTLIST part
  type CDATA #REQUIRED
  ref  CDATA #IMPLIED
>
```

[A.2](#) SDPng XML-Schema Specification

```
<xsd:schema
  xmlns:sdpng="http://www.iana.org/sdpng"
  targetNamespace="http://www.iana.org/sdpng"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
>
```

```
<!--
```

A data type for the "status" attribute

```
  status=mandatory: feature match MUST be successful
  status=opt:       optional feature, feature match MAY fail
-->
```

```
<xsd:simpleType name="status">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="mandatory"/>
    <xsd:enumeration value="opt"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!-- Base type for definition elements -->
```

```
<xsd:complexType name="Definition" abstract="true">
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
  <xsd:attribute name="ref" type="xsd:string" use="optional"/>
</xsd:complexType>
```

```
<!--
```

Data type for the content model of mandatory feature elements of type token

```
-->
```

```
<xsd:complexType name="token">
  <xsd:simpleContent>
    <xsd:extension base="xsd:NMTOKEN">
      <xsd:attribute name="status" type="sdpng:status" fixed="mandatory"/>
    </xsd:extension>
  </xsd:simpleContent>
```

</xsd:complexType>

Internet-Draft

SDPng

February 2005

<!--

Data type for the content model of optional feature elements of
type token

-->

<xsd:complexType name="opttoken">

<xsd:simpleContent>

<xsd:extension base="xsd:NMTOKEN">

<xsd:attribute name="status" type="sdpng:status" fixed="opt"/>

</xsd:extension>

</xsd:simpleContent>

</xsd:complexType>

<!--

Data type for the content model of mandatory feature elements of type
token list

-->

<xsd:complexType name="tokenlist">

<xsd:simpleContent>

<xsd:extension base="xsd:NMTOKENS">

<xsd:attribute name="status" type="sdpng:status" fixed="mandatory"/>

</xsd:extension>

</xsd:simpleContent>

</xsd:complexType>

<!--

Data type for the content model of optional feature elements of type
token list

-->

<xsd:complexType name="opttokenlist">

<xsd:simpleContent>

<xsd:extension base="xsd:NMTOKENS">

<xsd:attribute name="status" type="sdpng:status" fixed="opt"/>

</xsd:extension>

</xsd:simpleContent>

```
</xsd:complexType>
```

```
<!--
```

```
Data type for the content model of mandatory feature elements of type  
numerical value
```

```
-->
```

```
<xsd:complexType name="numval">
```

```
<xsd:attribute name="status" type="sdpng:status" fixed="mandatory"/>  
<xsd:attribute name="val" type="xsd:decimal"/>  
</xsd:complexType>
```

```
<!--
```

```
Data type for the content model of optional feature elements of  
type numerical value
```

```
-->
```

```
<xsd:complexType name="optnumval">  
  <xsd:attribute name="status" type="sdpng:status" fixed="opt"/>  
  <xsd:attribute name="val" type="xsd:decimal"/>  
</xsd:complexType>
```

```
<!--
```

```
Data type for the content model of mandatory feature elements of type  
numerical range
```

```
-->
```

```
<xsd:complexType name="numrange">  
  <xsd:attribute name="status" type="sdpng:status" fixed="mandatory"/>  
  <xsd:attribute name="min" type="xsd:decimal"/>  
  <xsd:attribute name="max" type="xsd:decimal"/>  
</xsd:complexType>
```

```
<!--
```

```
Data type for the content model of optional feature elements of  
type numerical range
```

```
-->
```

```

<xsd:complexType name="optnumrange">
  <xsd:attribute name="status" type="sdpng:status" fixed="opt"/>
  <xsd:attribute name="min" type="xsd:decimal"/>
  <xsd:attribute name="max" type="xsd:decimal"/>
</xsd:complexType>

```

```

<!-- Base type for definition elements -->

```

```

<xsd:complexType name="Constraint" abstract="true">
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
</xsd:complexType>

```

```

<!-- The base element for constraint element -->

```

```

<!-- FIXME: substitutionGroup? -->

```

```

<xsd:element name="constraint" type="sdpng:Constraint" abstract="true">
</xsd:element>

```

```

<!-- Base type for information elements -->

```

```

<xsd:complexType name="Information" abstract="true">
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="ref" type="xsd:string" use="optional"/>
</xsd:complexType>

```

```

<!-- The nformation part element -->

```

```

<xsd:element name="part" type="sdpng:Information" abstract="true">
</xsd:element>

```

```

<!-- ++++++ -->

```

```

<!--
SDPng document structure
-->

```

```

<xsd:element name="sdpng">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="sdpng:cap" minOccurs="0"/>
      <xsd:element ref="sdpng:def" minOccurs="0"/>
      <xsd:element ref="sdpng:cfg"/>
      <xsd:element ref="sdpng:constraints" minOccurs="0"/>
      <xsd:element ref="sdpng:info" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- The base element for capability and definition elements -->
<!-- FIXME: substitutionGroup? -->

  <xsd:element name="definition" type="sdpng:Definition" abstract="true">
  </xsd:element>

<!-- The optional "cap" element -->

  <xsd:element name="cap">
    <xsd:complexType mixed="false">

```

```

    <xsd:sequence>
      <xsd:element ref="sdpng:definition" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- The optional "def" element -->

  <xsd:element name="def">
    <xsd:complexType mixed="false">
      <xsd:sequence>
        <xsd:element ref="sdpng:definition" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

```

```

<!-- The mandatory "cfg" element -->

<xsd:element name="cfg">
  <xsd:complexType mixed="false">
    <xsd:sequence>
      <xsd:element ref="sdpng:component" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- The "component" element -->

<xsd:element name="component">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="sdpng:alt" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="media" type="xsd:string" use="optional"/>
    <xsd:attribute name="status" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="alt">
  <xsd:complexType mixed="false">
    <xsd:sequence>
      <xsd:element ref="sdpng:definition" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>

```

```

    <xsd:attribute name="status" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

```

<!-- The optional "constraints" element -->

```

```

<xsd:element name="constraints">
  <xsd:complexType mixed="false">
    <xsd:sequence>
      <xsd:element ref="sdpng:constraint" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```



```
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <!-- The optional "info" element -->

    <xsd:element name="info">
      <xsd:complexType mixed="false">
        <xsd:sequence>
          <xsd:element ref="sdpng:part" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

  </xsd:schema>
```

```

<xsd:schema
  xmlns:sdpng="http://www.iana.org/sdpng"
  xmlns:rtp="http://www.iana.org/sdpng/rtp"
  targetNamespace="http://www.iana.org/sdpng/rtp"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:import namespace="http://www.iana.org/sdpng" schemaLocation="sdpng-ba

  <xsd:complexType name="IPVersion">
    <xsd:simpleContent>
      <xsd:restriction base="sdpng:tokenlist">
        <xsd:enumeration value="IP4"/>
        <xsd:enumeration value="IP6"/>
      </xsd:restriction>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="RTPUDP">
    <xsd:complexContent>
      <xsd:extension base="sdpng:Definition">
        <xsd:all>
          <xsd:element name="network" type="rtp:IPVersion" minOccurs="0"/>
          <xsd:element name="ip-addr" type="sdpng:opttoken" minOccurs="0"/>
          <xsd:element name="rtp-port" type="sdpng:opttoken" minOccurs="0"/>
          <xsd:element name="rtcp-port" type="sdpng:opttoken" minOccurs="0"/>
          <xsd:element name="pt" type="sdpng:opttoken" minOccurs="0"/>
        </xsd:all>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="udp" type="rtp:RTPUDP" substitutionGroup="sdpng:definit

</xsd:schema>

```

[B.2](#) Sample Audio Package Definition

```
<xsd:schema
  xmlns:sdpng="http://www.iana.org/sdpng"
  xmlns:audio="http://www.iana.org/sdpng/audio"
  targetNamespace="http://www.iana.org/sdpng/audio"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:import namespace="http://www.iana.org/sdpng" schemaLocation="sdpng-ba

  <xsd:complexType name="Codec">
    <xsd:complexContent>
      <xsd:restriction base="sdpng:Definition">
        <xsd:all>
          <xsd:element name="encoding" type="sdpng:token" minOccurs="0"/>
          <xsd:element name="channels" type="sdpng:tokenlist" minOccurs="0"/>
          <xsd:element name="sampling" type="sdpng:tokenlist" minOccurs="0"/>
        </xsd:all>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="codec" type="audio:Codec" substitutionGroup="sdpng:defi

</xsd:schema>
```

[B.3](#) Sample Video Package Definition

```
<xsd:schema
  xmlns:sdpng="http://www.iana.org/sdpng"
  xmlns:video="http://www.iana.org/sdpng/video"
  targetNamespace="http://www.iana.org/sdpng/video"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:import namespace="http://www.iana.org/sdpng" schemaLocation="sdpng-ba

  <xsd:complexType name="Codec">
    <xsd:complexContent>
      <xsd:restriction base="sdpng:Definition">
        <xsd:all>
          <xsd:element name="encoding" type="sdpng:token" minOccurs="0"/>
```

```
<xsd:element name="sampling" type="sdpng:tokenlist" minOccurs="0"/>
```

```
    <xsd:element name="framerate" type="sdpng:opttokenlist" minOccurs="0"
    <xsd:element name="size" type="sdpng:opttokenlist" minOccurs="0"/>
    <xsd:element name="bitrate" type="sdpng:optnumrange" minOccurs="0"/>
    <xsd:element name="min-quant" type="sdpng:optnum" minOccurs="0"/>
    <xsd:element name="max-quant" type="sdpng:optnum" minOccurs="0"/>
    <xsd:element name="gop-size" type="sdpng:optnum" minOccurs="0"/>
  </xsd:all>
  </xsd:restriction>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="codec" type="video:Codec" substitutionGroup="sdpng:defi

</xsd:schema>
```

[Appendix C](#). Sample SDPng Description

```
<?xml version="1.0" encoding="UTF-8"?>

<sdpng xmlns="http://www.iana.org/sdpng"
  xmlns:audio="http://www.iana.org/sdpng/audio"
  xmlns:video="http://www.iana.org/sdpng/video"
  xmlns:rtp="http://www.iana.org/sdpng/rtp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.iana.org/sdpng sdpng-base.xsd
http://www.iana.org/sdpng/audio sdpng-audio-pkg.xsd
http://www.iana.org/sdpng/video sdpng-video-pkg.xsd
http://www.iana.org/sdpng/rtp sdpng-rtp-pkg.xsd"
>

  <cap>
    <audio:codec name="avp:pcmu">
      <audio:encoding>PCMU</audio:encoding>
      <audio:channels>1</audio:channels>
      <audio:sampling>8000</audio:sampling>
    </audio:codec>
    <audio:codec name="avp:gsm">
      <audio:encoding>GSM</audio:encoding>
      <audio:channels>1</audio:channels>
      <audio:sampling>8000</audio:sampling>
    </audio:codec>
    <audio:codec name="avp:g723">
      <audio:encoding>G723</audio:encoding>
      <audio:channels>1</audio:channels>
      <audio:sampling>8000</audio:sampling>
    </audio:codec>
    <audio:codec name="avp:dvi4">
      <audio:encoding>DVI4</audio:encoding>
```

```

        <audio:channels>1</audio:channels>
        <audio:sampling>8000 11025 16000 22050</audio:sampling>
</audio:codec>
<audio:codec name="avp:lpc">
    <audio:encoding>LPC</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g722">
    <audio:encoding>G722</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:l16">
    <audio:encoding>L16</audio:encoding>

```

```

        <audio:channels>1 2</audio:channels>
        <audio:sampling>44100</audio:sampling>
</audio:codec>
<audio:codec name="avp:qcelp">
    <audio:encoding>QCELP</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:cn">
    <audio:encoding>CN</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:mpa">
    <audio:encoding>MPA</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>32000 44100 48000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g728">
    <audio:encoding>G728</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g729">
    <audio:encoding>G728</audio:encoding>
    <audio:channels>1</audio:channels>

```

```

        <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g726-40">
    <audio:encoding>G726-40</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g726-32">
    <audio:encoding>G726-32</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g726-24">
    <audio:encoding>G726-24</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g726-16">
    <audio:encoding>G726-16</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>

```

```

<audio:codec name="avp:g729d">
    <audio:encoding>G729D</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:g729e">
    <audio:encoding>G729E</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:gsm-efr">
    <audio:encoding>GSM-EFR</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>8000</audio:sampling>
</audio:codec>
<audio:codec name="avp:l8">
    <audio:encoding>L8</audio:encoding>
    <audio:channels>1 2</audio:channels>
    <audio:sampling>8000 16000</audio:sampling>

```

```

</audio:codec>
<audio:codec name="avp:red">
    <audio:encoding>RED</audio:encoding>
    <audio:channels>1 2</audio:channels>
    <audio:sampling>8000 16000</audio:sampling>
</audio:codec>
<audio:codec name="avp:vdvi">
    <audio:encoding>RED</audio:encoding>
    <audio:channels>1</audio:channels>
    <audio:sampling>var</audio:sampling>
</audio:codec>

<video:codec name="avp:celb">
    <video:encoding>CelB</video:encoding>
    <video:framerate>4 6 8 12 16 20 24 30</video:framerate>
</video:codec>

<rtp:udp name="rtpudpip6">
    <rtp:network>IP6</rtp:network>
</rtp:udp>

</cap>
<def>
    <rtp:udp name="rtp-cfg1" ref="rtpudpip6">
        <rtp:ip-addr>::1</rtp:ip-addr>
        <rtp:rtp-port>9546</rtp:rtp-port>
    </rtp:udp>
</def>
<cfg>

```

```

    <component>
        <alt>
            <audio:codec ref="avp:pcmu"/>
            <rtp:udp ref="rtp-cfg1">
                <rtp:pt>0</rtp:pt>
            </rtp:udp>
        </alt>
    </component>
</cfg>
</sdpng>

```


[Appendix D](#). Change History

[draft-ietf-mmusic-sdpng-08.txt](#)

- * Changed introduction: emphasis on non-negotiation scenarios

(broadcast etc.), describe main concepts

- * element "cap" is now OPTIONAL.
- * new "status" attribute for "component" element type
- * new "part" element for "info" element (meta-information)
- * Removed section "Specification of the Capability Negotiation"
- * Removed appendix "Use of SDPng in Conjunction with other IETF Signaling Protocols"
- * Added section "Usage of SDPng in Different Application Scenarios"
- * Updated DTD and XML-Schema-Definition wrt to afore-mentioned changes

[draft-ietf-mmusic-sdpng-07.txt](#)

- * New document structure:
 1. Intro
 2. Terminology and System Model
 3. Overview
 4. SDPng Syntax Specification
 5. Negotiation Process
- * Changes to [Section 3](#): Describe all concepts
- * [Section 4](#) provides complete specification
- * Changed XML syntax: Represent tokens and token list as element content (not attributes)
- * a new element "def" for reusable definitions
- * Adapted section 5 accordingly

- * Sample DTD, schema definition and same SDPng document in appendix
- * Updated [section 5.1](#) (Offer/Answer)
- * Updated [appendix D](#) (Use of SDPng in conjunction with other IETF Signaling Protocols)

[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 a list of terms with explanation at the end of [Section 2](#).
- * Removed audio and RTP packages from appendix.
- * Added a section "Syntax Definition".
- * Added section "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. 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.

Internet-Draft

SDPng

February 2005

- * New section on referencing definitions.
- * 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

Internet-Draft

SDPng

February 2005

IPR Notice

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

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

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES,

EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

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