

Capability description for group cooperation  
draft-ott-mmusic-cap-00.txt

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

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/lid-abstracts.txt>

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

Abstract

This document presents a notation for describing potential and specific configurations of end systems in multiparty collaboration sessions. The objective is to define a configuration description framework that can be used to define end system capabilities, to calculate a set of appropriate common capabilities based on the descriptions of all (end) systems and to express a selected media description for use in session descriptions. One application for this framework would be multiparty multimedia conferencing, an application area where multiple tools have to be configured on conference startup (and/or during the conference) concerning media encoding types and other parameters. Other applications are IP Telephony and media gateway control.

This document is intended for discussion in 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 [confctrl@isi.edu](mailto:confctrl@isi.edu) and/or the authors.

## 1. Introduction

### 1.1. Background

#### 1.1.1. Motivation

Multiparty multimedia conferencing is one application that requires the 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. Currently the parameter negotiation is either done by out of band means or, for loosely coupled conferences, parameters are simply fixed by the initiator of a conference. In the latter scenario no negotiation is required because only those participants with media tools that support the predefined settings can join a media session and/or a conference.

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

To allow for spontaneous conferences, the process of defining a conference's parameter set must therefore be performed either at conference start (for closed conferences) or maybe (potentially) even repeatedly every time a new participant joins an active conference. The latter approach may not be appropriate for every type of conference: 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 exclude the main source from media sessions. But conferences with equal ``rights'' for participants that are open for new participants do need dynamic capability negotiation, for example a telephone call that is extended to a 3-parties conference at some time during the session.

#### 1.1.2. Current practices in the IETF community

Capability and session descriptions play different roles in applications of IETF conferencing standards and are currently almost always specified as SDP (Session Description Protocol) [[11](#)] session descriptions. In session announcements with SAP (Session Announcement Protocol) [[12](#)] they are used to define media encodings and parameters

the participants or the active source.

Within the context of SIP (Session Initiation Protocol) capability descriptions can be expressed in different session description languages, one of them SDP. For example, in a SIP-INVITE message for a unicast session, the session description enumerates the media types and formats that the caller is willing to use and thus expresses the capabilities of the caller's end system. The SDP content is however not only used to express a caller's preferences but is also used to configure communication channels in a somewhat crude way. For example, if a callee does not want to send or receive data on a offered stream he has to set the port number of that stream to zero in its media description that he sends as a reply to the caller. The use of SDP as a capability description and negotiation mechanism has lead to a whole set of conventions and requirements that have to be considered by implementations because SDP itself is not powerful enough for this purpose. This is clearly not a defect of SDP which has never been designed to be a complete capability description and negotiation mechanism. SDP has been developed in the context of SAP to describe simple static media sets.

The misuse of SDP reveals a lack of a powerful, yet simple way to perform capability description and negotiation in a conference setup or reconfiguration phase in the current IETF conferencing model.

## [1.2.](#) Purpose

The configuration negotiation framework consists of three components:

- o A language that allows expressing capability descriptions, potential configurations, unambiguously;
- o an algorithm that compares different capability descriptions and produces an appropriate ``collapsed'' subset that can be used as a common set of potential configurations; and
- o a concrete capability name and value range specification for specific applications.

This documents specifies ways to express potential and concrete configurations as well as rules to combine, constrain, and collapse these configurations. How a particular component's potential configurations are gained, what relationship exists to system

capabilities, and similar meta-discussions are beyond the scope of this document.

It is also not the purpose of this document to specify a complete framework including mandatory protocols for capability exchange. Names and value ranges for different applications should be defined in a follow-up document and registered with the IANA.

Besides modeling and rules, this document specifies a syntax for

expressing configurations and describes a basic and a concise representation format as well as an XML-based notation. A number of appendices provide mappings to other specification formats (in particular SDP and H.245) as far as possible and also give an overview of semantic definitions for configurations for audio codecs.

### [1.3.](#) Relation to other Developments

A few other generic or application specific models have been developed that deal with capability description and/or capability negotiation.

[RFC 2295](#) (Transparent Content Negotiation in HTTP) [3] proposes a negotiation mechanism layered on top of HTTP that allows for automatically selecting the ``best'' version of documents that are accessible by a single URI. A server can describe the properties of each variant of a document associated with ``quality degradation factors''. The content negotiation process will either allow the client to select the appropriate version according a variant list provided by the server or the server itself may choose a document version relying on Accept-headers that are included in the client's request.

The Resource Description Framework (RDF) [4] provides a specification model for properties of Web resources and aims at automating processing Web resources with respect to resource discovery, cataloging, resource selection and other applications.

CC/PP [5] is an on-going development that is creating a framework for describing user preferences and device capabilities that uses RDF to express those descriptions. In the CC/PP model a user agent can provide capability profiles that enable servers and proxies to customize content accordingly.

The IETF Content Negotiation (conneg) working group is developing a

collection of media features for display, print and fax [6], a registration procedure for feature tags (the names of capability properties) [7] as well as description and negotiation models [8] [9] for media features and capabilities. One of conneg's goals is to develop a ``tag independent negotiation'' process that can work without knowing the meaning of feature tags.

Whereas TCN, RDF and CC/PP focus on describing/negotiating capabilities for client/server scenarios such as the WWW, where a server provides content with certain properties and a client has certain preferences/capabilities, the conneg approach is more general. The conneg framework provides the abstraction of ``feature sets'' that are media feature collections. Feature sets can either be interpreted as a set of variants that a server can provide as data formats or as a set of capabilities of a receiver. Content negotiation in this model would be to find a non-empty feature set that is compatible with both the sender's and the receiver's original

feature set.

H.245, the multimedia control protocol employed across all newer H.32x Recommendations for tightly-coupled multimedia conferencing (particularly included H.323) provides the concept of capability specification and exchange between terminals and uses the same description mechanisms to define particular instantiations of media streams in a conference. For capability description purposes, H.245 provides means to express all the capabilities supported by a system (``AlternativeCapabilitySets'') as well as to describe permitted combinations of these capability sets to be instantiated at the same time. Capability exchange is defined on a peer-to-peer basis, common (``collapsed'') capabilities are calculated by some central entity that controls the mode of operation in a multipoint conference. This calculation requires the central entity to understand the (semantics of) the individual endpoints' capability descriptions.

T.124 specifies a framework for exchanging and collapsing capabilities. This framework specifies a core set of rules (minimum, maximum, logical AND) and capability types as well as a naming scheme, but leaves definition of specific semantics to the application protocols. This concept makes the framework extensible and enables entities to calculate a common set of supported capabilities without having to understand their semantics. Also, T.124 distinguishes between capability descriptions and particular instantiations for application sessions. In addition to these collapsing capabilities, T.124 supports the notion of non- collapsing

capabilities to which the collapsing process is not applied.

Capability negotiation for groups of senders and receivers as presented in this document can be viewed as a specialization of the general conneg approach that focuses on simplicity for capability descriptions. Some expressional power of the conneg framework is abandoned in favor of simplicity.

#### [1.4.](#) Terminology for requirement specifications

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC 2119](#) [1] and indicate requirement levels for compliant implementations.

## [2.](#) Requirements and Concepts

### [2.1.](#) System Model

Any (computer) system has 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 machine. We term features enabled and

restricted by these resources "system capabilities".

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

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

Example: In lecture multicast conferences one component might be the voice transmission for the lecturer, another the transmission of video pictures showing the lecturer and the third the transmission of presentation material that are different components in a conference.

Depending on system capabilities, user preferences and other technical and political constraints, different configurations can be chosen to accomplish the ``deployment'' of these components.

Each component can be characterized at least by (a) its intended use (i.e. the function it shall provide) and (b) a 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 an copy of the slides into a distributed electronic whiteboard. For each of these cases, additional parameters may exist, leading to additional configurations (see below).

Two configurations are considered different regardless 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 system 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.

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.

A configuration consists of any number of properties and is uniquely identified by a tag. Potential configurations can be grouped into alternatives each of which indicates a possible mode of operation of a component.

In a conference, each involved peer contributes to the formation of a component's configuration -- by specifying its own features and limitations during the capability exchange process. Based upon all systems' input, a set of common capabilities -- potential configurations -- is calculated through the collapsing process.

The collapsing process may be influenced by additional constraints that may be expressed on the possible combinations of alternatives -- between multiple instances of the same component as well as across (instances of) different components. Also, user preferences may be taken into account -- during the collapsing process as well as when deciding on which potential configuration is to be instantiated as the actual configuration for a component.

## [2.2.](#) Definition of terms

From the system model described above, the following core terms can be extracted:

- o conference component

An element of a multiparty multimedia conference that can appear

as a media stream and has a set of potential configurations.

- o configuration



A set of named attributes, expressing constraints to a system's capabilities.

- o capability

Resources or system features that influence the selection of useful configurations for components.

- o alternative

When comparing different potential configurations, one potential configuration is an alternative to other configurations.

- o property

A property is a label-value pair.

The capability description language specified in this document is called CAP.

### [2.3.](#) Description language

The objective of a capability description language is to allow the definition of supported media types, encodings and features of an end system. The language must be unambiguous, easily parsable and allow for concise definitions to minimize the transport overhead for a capability negotiation phase during a conference. It should also be extensible and not fixed to certain features, because new encodings must be supported without changes to the language definition.

To ensure the unambiguousness it is however required to have a common understanding on the meaning of identifiers and values. E.g. if two end systems used different names for the audio encoding ``GSM'' a capability negotiation would not lead to the desired result. The need for well-known identifiers and the need for extensibility require to separate the definition of identifiers and values from the definition of the description language itself. Identifiers and values should therefore be standardized and registered.

### [2.4.](#) Collapsing Algorithm

The objective of the collapsing algorithm is to take capability description sets from each end system in order to find a set of media-types, encodings and features that are supported by all end system, or, if this is not possible, to find a subset that would exclude as few systems as possible.

The procedure described above would be the default algorithm. In certain scenarios where some end systems are privileged it must be possible to ensure that the result of the collapsing process does not exclude those privileged systems. It must therefore be possible to parameterize the process with the policy to be applied.

### [3.](#) Specification of the Description Language

Two, semantically equivalent, notations are introduced. The first notation is simple but leads to verbose capability descriptions and the second notation is more complex but allows for concise descriptions.[\[1\]](#) This specification also defines how to translate descriptions using the concise notation to the other, simpler, format.

Please note that all tags and values are just examples and not a subject of this specification.

#### [3.1.](#) Basic Description Language

In the basic description language a end system's capability description is a set of alternatives. An alternative is a set of constraints for certain parameters. A constraint can be understood as a restriction because it limits the capability alternative according to the constraint's meaning.

A constraint is constituted of three components:

-----+-----	-----+-----
tag	name of the constraint
operator	defines the type of the constraint
value	a value for the constraint operator
-----+-----	-----+-----

Table 1: Components of a Constraint

A constraint that limits the capability of an end system to a maximum transfer rate of 64 kbit/s (say in a description of audio receiver capabilities) would be written as follows:

```
bps <= 64000;
```

with bps as the tag, <= as the operator and 64000 as the value of this constraint (plus a semicolon as a end-of-statement-symbol).

A complete alternative (a set of constraints) would be written as:

---

[\[1\]](#) A third, XML-based notation is included in [appendix A](#).

```

media = audio;
mode = receive | send;
channels = 1;
encoding = g711;
compression = mulaw;
sampling_rate = 8000 | 11025 | 16000;

```

This example exhibits another way of expressing constraints using the = operator. The = operator can be used to define a set of supported values in a single constraint. The value of the = operator's value is actually a list of names separated by `|'. In the definition of the media constraint it is shown how a single name is used as a value for the = operator, which has the meaning that (for the respective alternative ) only the media-type audio is supported.

Another operator that is not shown in the example is the operator >= that can be used to express minimum constraints. Table 2 provides an overview of the operators:

+---+-----+	
<=	maximum
>=	minimum
=	selection of fixed values
+---+-----+	

Table 2: Operators for Capability Constraints

The reason why the sampling\_rate constraint is expressed with a = and not with a <= operator is that defining the rate capability as a maximum constraint with a value of 16000 would allow any value less than 16000 as a valid parameter which would not match the application specific semantics in this case.[\[2\]](#)

The example above contains one alternative of a capability description. It could be used as a complete description expressing that the end system does not support more than this specific alternative. Most end system however support more variants of audio parameters, requiring the definition of more alternatives. E.g. supporting ``GSM'' as a second encoding would lead to the following capability description:

```

tag: audio/g711
media = audio;
mode = receive | send;
channels = 1;

```

```
encoding = g711;
compression = mulaw;
sampling_rate = 8000 | 11025 | 16000;
```

-----  
[2] Most codecs do not support arbitrary sampling rates.

Ott/Kutscher/Bormann

[Page 10]

---

INTERNET-DRAFT Capability description for group cooperation

June 1999

```
tag: audio/gsm
media = audio;
mode = receive | send;
channels = 1;
encoding = gsm;
compression = half | full | enhanced_full;
```

This description expresses that the end system supports one media type ``audio'' and two audio encodings ``g711'' and ``gsm'', each with certain other constraints. This way of defining capabilities is very redundant as many constraints are the same for both alternatives. It is important to know all the constraints of an alternative for a later negotiation phase (see below) but for writing and transferring capability descriptions another notation that expresses common constraints and allows for more concise definition is useful.

The = operator is actually already used to aggregate several constraints into one: A hypothetical even more primitive notation could translate each alternative containing a = constraint into a set of alternatives each containing a ``equality constraint'' for one value of the = value list. E.g. for the GSM alternative there would be 3 alternatives for each compression type (each variant again would require an alternative for receive and for send mode in this example). This has not been done in this example in order to avoid the obvious verbosity. Every alternative containing a = constraint with n values can however unrolled to n different alternatives if this granularity is required.

Each alternative also contains a tag that allows to reference it later in simultaneous capability specifications. Due to the possibility to aggregate alternatives with = constraints several specific codec parameters for a media codec can be subsumed under one common tag like in the example above. This allows to handle common cases, where this is desired, efficiently. Again, if more granularity is needed for specific applications, = constraints can be unrolled.

The ABNF[2] specification for the basic description language is as

[Page 11]

June 1999

Note that the specification does currently not provide ``non-collapsing'' attributes, i.e. attributes that are not considered in collapsing rules, except for tags. Another syntactic element for those attribute will be added in the future.

### 3.2.1. Syntax

```
media: audio {
    mode = receive | send;
    channels = 1;
    encoding: g711 {
        compression = mulaw;
        sampling_rate = 8000 | 11025 | 16000;
    } || encoding: gsm {
        compression = half | full | enhanced_full;
    };
};
```

An alternative group contains those constraints (and subgroups) that are specific to an alternative and cannot be expressed in the common part. A group is enclosed by curly brackets and follows a group-tag (like ``encoding: g711'' in the example). A group-tag is semantically a ``=' constraint (with one value) but is used in the concise notation to introduce a new subgroup of constraints.

The example above contains three groups: The top-level group ``media: audio'' and two second-level groups ``encoding: g711'' and ``encoding: gsm''. Groups on the same hierarchy level (siblings) are connected by ``||''. Groups can be nested to arbitrary levels and there is no limit for the number of siblings in a hierarchy. The next example shows how the ``encoding: g711'' group can be split-up into 2 subgroups:

```
media: audio {
    mode = receive | send;
    channels = 1;
    encoding: g711 {
        compression: mulaw {
            sampling_rate = 8000 | 11025 | 16000;
        } || compression: alaw {
            sampling_rate = 8000 | 11025 | 32000;
        };
    } || encoding: gsm {
        compression = half | full | enhanced_full;
    };
};
```

Note that there are no explicit tags allowed for the concise notation. Instead group tags serve as implicit tags components that can be composed to unique tags for each expressed alternative. A alternative can be uniquely specified by joining the group tags of all enclosing groups. The specification example above would thus define three alternatives: audio/g711/mulaw, audio/g711/alaw and audio/gsm. Tag concatenation uses "/" (slash) as a delimiting character.

The ABNF[2] specification for the concise description language is as follows (as an extension to the ABNF of the basic language, see [section 3.1](#)):

```
+-----+
| caps      = 1*(group LWSP *("||" LWSP group) *WSP |
|           ";" )                                     |
+-----+
```

group	=	group-tag *WSP "{" LWSP *constraint	
		[caps] LWSP "}"	
group-tag	=	name ":" *WSP tag	

+-----+

### [3.2.2.](#) Translation to Basic Notation

Transforming a capability description from concise to basic notation MUST be done by applying the following algorithm, starting at the outermost hierarchy level and transforming subgroups recursively:

Ott/Kutscher/Bormann

[Page 13]

INTERNET-DRAFTCapability description for group cooperation

June 1999

```

transform group-tag to = constraint;
push group-tag to tag stack;
adopt all other constraints within the group;

for each group in this level {
    add adopted constraints and transformed group-tag
    to every alternative obtained from transforming the
subgroups
    recursively resulting in a set of alternatives;
    if (is innermost group) {
        construct tag by concatenating all group-tags
from tag stack
        and add it to alternative;
    }
    pop tag stack;
}

```

Two innermost subgroups at the same hierachy level are thus converted to two alternatives. An Example:

```

A: B {
    C <= 1;
    D: E {
        F <= 2;
        G:H {
            I <= 3 ;
        } || J: K {
            L <=4;
        }
    } || M: L {
        N <= 5;
    }
}

```

would be transformed into the following set of alternatives:

```
tag: B/E/H
A = B;
C <= 1;
D = E;
F <= 2;
G = H;
I <= 3;
```

```
tag: B/E/K
A = B;
C <= 1;
D = E;
F <= 2;
J = K;
L <= 4;
```

```
tag: B/L
A = B;
C <= 1;
M = L;
N <= 5;
```

### [3.2.3.](#) Translation from Basic to Concise Format

The mapping from basic to concise representation is not unique by itself: In principle, for all alternatives constraints with common values can be factored out. Depending on the constraints that are chosen for outer groups the results will differ. Nevertheless it would be possible to define an algorithm that will guarantee uniqueness, for example by defining certain tags as implicit outer-level tags (e.g. ``media'') and by demanding that those constraints with the largest number of equal values in many alternatives will appear in the outermost groups. Conflicts could be avoided by imposing a lexicographic ordering on the tags. Only ``=' constraints with one parameter can be chosen for group tags.

## [4.](#) Specification of constraints for simultaneous capabilities

For some applications it is not sufficient to be able to express the capability to support a list of media types and codec parameters. Instead constraints of how many instances of codecs of different types can be active at a given time must also be specified as an input parameter for a negotiation/selection process.



For example a gateway may be able to handle either 5 GSM streams or, alternatively, 5 G.711 streams at the same time but not both GSM and G.711 at the same time.

The specification presented here enables the definitions of such constraints by the tagging mechanism. Alternative capability can be referred to in rules expressing those simultaneous constraints using their tags. The specification of such a definition language is however not subject of this draft and will have to be defined elsewhere.

## [5.](#) Specification of the Collapsing Process

The collapsing process generates a set of alternatives, according to the collapsing policy and the set of alternatives that are used as the input to this process.

### [5.1.](#) Finding compatible alternatives

Ott/Kutscher/Bormann

[Page 15]

---

INTERNET-DRAFTCapability description for group cooperation      June 1999

The general collapsing process tries to find a set of alternatives that are supported by every end system. This must be accomplished by comparing each alternative of an end system's alternative set with each alternative of every other alternative set.

The process of collapsing two alternatives works as follows:

```

    find intersection of constraints of the two alternatives by
    keeping all constraint with same names and operators;
    for all constraints in the intersection {
        find according constraint (same name and operator) in
second set;
        if(operator=='<=') {
            calculate minimum of both constraint values and
add
            maximum constraint with that value to result set;
        }
        if(operator=='>=') {
            calculate maximum of both constraint values and
add
            minimum constraint with that value to result set;
        }
        if(operator=='=') {
            Build intersection of tags in both constraints;
            add = constraint with a value of the
```

```
                                intersection to the result set;
                                }
                                }
}
```

Tags are ignored in the collapsing process. If the result set of alternatives contains = constraints with empty value lists the collapsing of these two alternatives has failed and the resulting set must be discarded.

## [5.2.](#) Other policies

Other collapsing policies will have to be defined.

## [6.](#) Composed Configurations

For certain configurations it is required to compose configurations by combining or referencing other configurations. Sample application could be redundant and FEC encodings. A full specification how this can be accomplished will have to be defined. The general outline would be to use the structuring and referencing mechanisms (tagged alternative) to express the required constraints for the respective encodings.

## [7.](#) Security Considerations

Security considerations will also have to be defined.

Ott/Kutscher/Bormann

[Page 16]

---

INTERNET-DRAFT Capability description for group cooperation

June 1999

## [8.](#) Authors' Addresses

Joerg Ott <jo@tzi.org>  
Universitaet Bremen, TZI, MZH 5180  
Bibliothekstr. 1  
D-28359 Bremen  
Germany  
voice +49 421 201-7028  
fax +49 421 218-7000

Dirk Kutscher <dku@tzi.org>  
Universitaet Bremen, TZI, MZH 5160  
Bibliothekstr. 1  
D-28359 Bremen  
Germany  
voice +49 421 218-7595  
fax +49 421 218-7000

Carsten Bormann <cabo@tzi.org>  
Universitaet Bremen, TZI, MZH 5180  
Bibliothekstr. 1  
D-28359 Bremen  
Germany  
voice +49 421 218-7024  
fax +49 421 218-7000

## 9. References

- [1] S. Bradner, ``Key words for use in RFCs to Indicate Requirement Levels'', [RFC 2119](#), March 1997
- [2] D. Crocker, P. Overell, ``Augmented BNF for Syntax Specifications: ABNF'', [RFC 2234](#), November 1997
- [3] K. Holtman, A. Mutz, ``Transparent Content Negotiation in HTTP'', [RFC 2295](#), March 1998
- [4] O. Lassila, R. Swick, ``Resource Description Framework (RDF) Model and Syntax Specification'', W3C Proposed Recommendation, January 1999, work in progress, <http://www.w3.org/TR/1999>
- [5] F. Reynolds, J. Hjelm, S. Dawkins, S. Singhal, ``Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation'', W3C Note 30, November 1998, work in progress, <http://www.w3.org/TR/1998/NOTE-CPP-19981130>
- [6] L. Massinter, K. Holtman, A. Mutz, D. Wing, ``Media Features for

Ott/Kutscher/Bormann

[Page 17]

---

INTERNET-DRAFTCapability description for group cooperation      June 1999

Display, Print, and Fax'', Internet Draft [draft-ietf-conneg-media-features-05.txt](#), January 1998, Work in Progress

- [7] K. Holtman, A. Mutz, T. Hardie, ``Media Feature Tag Registration Procedure'', Internet Draft [draft-ietf-conneg-feature-reg-03.txt](#), July 1998, Work in Progress
- [8] G. Klyne, ``A syntax for describing media feature sets'', Internet Draft [draft-ietf-conneg-feature-syntax-04.txt](#), December 1998, Work in Progress
- [9] G. Klyne, ``An algebra for describing media feature sets'', Internet Draft [draft-ietf-conneg-feature-algebra-03.txt](#), August 1998, Work in Progress

- [10] G. Klyne, ``W3C Composite Capability/Preference Profiles'', Internet-Draft [draft-ietf-conneg-W3C-ccpp-01.txt](#), December 1998, Work in progress
- [11] M. Handley, ``SDP: Session Description Protocol'', [RFC 2327](#), April 1998
- [12] M. Handley, C. Perkins, E. Whelan, ``Session Announcement Protocol'', Internet-Draft [draft-ietf-mmusic-sap-v2-01.txt](#), June 1999, Work in progress

Ott/Kutscher/Bormann

[Page 18]

---

INTERNET-DRAFT Capability description for group cooperation      June 1999

## Appendix A: XML-DTD for the description language

A XML-DTD for XML documents representing concise CAP descriptions:

```
<!ELEMENT cap (media*)>
<!ATTLIST cap
  version CDATA "1.0"
>

<!ELEMENT media (property|min|max|one.of|group)+>
<!ATTLIST media
  type CDATA #REQUIRED
>

<!ELEMENT group (property|min|max|one.of|group)+>
<!ATTLIST group
  name CDATA #REQUIRED
  val CDATA #REQUIRED
>

<!ELEMENT property (#PCDATA)>
<!ATTLIST property
  name CDATA #IMPLIED
>

<!ELEMENT min EMPTY>
<!ATTLIST min
  name CDATA #REQUIRED
  val CDATA #REQUIRED
>

<!ELEMENT max EMPTY>
<!ATTLIST min
```

```

    name CDATA #REQUIRED
    val CDATA #REQUIRED
  >

  <!ELEMENT one.of (property)+>
  <!--ATTLIST one.of
    name CDATA #REQUIRED
  >

```

The example explained above represented in XML:

```

<?xml version="1.0"?>
<cap version="1.0">
  <media type="audio">
    <one.of name="mode">
      <property>receive</property>
      <property>send</property>
    </one.of>
    <property name="channels">1</property>
    <group name="encoding" val="g711">
      <group name="compression" val="mulaw">
        <one.of name="sampling_rate">
          <property>8000</property>
          <property>11025</property>
        </one.of>
      </group>
      <group name="compression" val="alaw">
        <one.of name="sampling_rate">
          <property>8000</property>
          <property>11025</property>
        </one.of>
      </group>
    </group>
    <group name="encoding" val="gsm">
      <one.of name="compression">
        <property>half</property>
        <property>full</property>
        <property>enhanced_full</property>
      </one.of>
    </group>
  </media>
</cap>

```

Note that = constraints with one alternative are represented as property elements for brevity while = constraints with multiple

alternatives are represented as one of elements with a property element (without name attribute) for each value.

## Appendix B: Mapping from/to SDP

Note that this appendix is still preliminary as it does not yet cover all the features provided by the capability description language presented in this document.

SDP allows for describing all parameters required for establishing a conference. The media parameters that can be interpreted as a caller's capabilities are only a subset of the session description. Other information like origin ('o=' field) or communication parameters are not related to a system's capability description (although they need to be expressible in a session description language, as well). An example SDP description:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=video 51374 RTP/AVP 98
a=rtpmap:98 X-H.263+
m=application 32416 udp wb
a=orient:portrait
```

Only the 'm=' and the respective 'a=' fields contain relevant information for a mapping to our capability description language. The first element of a 'm=' field is the media type that can be mapped to the tag of a top-level 'group-tag' in the concise description language. The second element of a 'm=' field, the transport port, is a communication parameter and can therefore be neglected for now. The third and fourth (and subsequent) elements define a transport protocol (that can be regarded as some kind of capability) and media formats (encodings). The 'm=' field may be followed by an 'a=' field that can contain arbitrary constraints on the media description, notably the rtpmap attribute, that maps a dynamic RTP payload type number to a media format (and additional

encoding parameters, depending on the concrete encoding). Further encoding specific parameters are specified using a ``a=fmtp'' attribute. All parameters of a ``a=fmtp'' attribute will be mapped to respective constraints in our description language. The concrete mapping is yet to be defined for some common uses of ``a=fmtp''.

For the sake of generality we must translate the implicit encoding parameters expressed in static RTP payload numbers to explicit descriptions and extract the relevant information from the ``a='' fields for dynamic payload types.

The example above could therefore be translated as:

```
media: audio {
    mode = receive | send
    encoding: g711 {
        transport = RTP
        compression = mulaw
        sampling_rate = 8000
        channels = 1
    }
}
media: video {
    mode = receive | send
    encoding: h261 {
        transport = RTP
    } || encoding: h263+ {
        transport = RTP
    }
}
media: application {
    type: wb {
        transport = UDP
        orientation = portrait
    }
}
```

The constraints inside the ``g711'' group have to be adopted from the payload types definition in [RFC 1890](#). The ``transport'' constraint could also be factored-out to the outer groups ``audio'' and ``video'' -- this is not relevant to the semantics of the description. Note that the empty group for ``h261'' and ``h263+'' can also be abbreviated as a ``= constraint'' if no specific constraints exist for those encodings.

The mapping process can thus defined as follows:

- 1) Each ``m=<media>'' format specification is mapped to a ``group'' nested in a ``group'' for the respective media. The tag for that group is inferred either from the static payload type or in case of dynamic payload types looked up from a corresponding ``a=rtpmap'' field. The corresponding registered payload type name leads to an encoding name (by a yet to be defined name map). A mapping for unregistered payload type names has to be defined, as well.
- 2) The transport of a ``m='' field becomes a ``=''' constraint in the ``group'' for the encoding
- 3) For registered payload type names the additional parameters as defined in [RFC 1890](#) such as sampling rate and number of channels are each translated into corresponding ``=''' constraints of the encoding group.

- 4) Translation of ``a=fmtp'' has to be defined...
- 5) All other ``a='' fields relating to a ``m='' and representing a single attribute-value mapping (like orient:portrait) are translated into single ``=''' constraints with one value.

Future versions of this specification will also define how integrate other SDP configuration parameters into CAP using non-collapsing parameters (see section xx) that are yet to be defined.

Translating a description written in the concise description language (back) to SDP again would rely on a well-defined mapping of encoding names:

- 1) CAP names for video or audio that cannot be translated into registered payload type names will be translated as dynamic payload types with a corresponding ``a=rtpmap'' field.
- 2) CAP groups with encoding names that can be mapped are either translated into ``m='' fields with static payload types if the encoding parameters (sampling rate and number of channels) conform to the specification of a static payload type or, if one of these parameters differ are translated to ``m='' fields with a dynamic payload type that will be defined in a subsequent ``a=rtpmap'' field.



- 3) For other media types the encoding groups will be translated to ``m=application'' fields with the encoding name as the fourth element.
- 4) The transport constraint of the CAP description is will be reflected in the ``m=''' field, as well.
- 5) Other = constraints with one value will be translated to ``a=''' fields.

## Appendix C: Integration into SDP

Instead of translating a CAPs specification into SDP media descriptions it can be more efficient to directly add it to a SDP description and thus retain the original specification. This can be done by using dynamic payload types:

```
m=audio <port> <transport-parameters> 98
```

```
a=rtpmap:98 X-CAP
```

```
a={ channels = 1; encoding: g711 { compression = mulaw};sampling_rate
= 8000 | 11025 | 16000; } || encoding: gsm { compression = half |
full | enhanced_full; }; }
```

## Appendix D: Mapping to H.245

TBD.