Network Working Group
Request for Comments: 3840
Category: Standards Track

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia University
P. Kyzivat
Cisco Systems

August 2004

Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This specification defines mechanisms by which a Session Initiation Protocol (SIP) user agent can convey its capabilities and characteristics to other user agents and to the registrar for its domain. This information is conveyed as parameters of the Contact header field.

Table of Contents

1. Introduction

Session Initiation Protocol (SIP) [1] user agents vary widely in their capabilities and in the types of devices they represent. Frequently, it is important for another SIP element to learn the capabilities and characteristics of a SIP UA. Some of the applications of this information include:

- One user agent, a PC-based application, is communicating with another that is embedded in a limited-function device. The PC would like to be able to "grey out" those components of the user interface that represent features or capabilities not supported by its peer. To do that, there needs to be a way to exchange capability information within a dialog.

- A user has two devices at their disposal. One is a videophone, and the other, a voice-only wireless phone. A caller wants to interact with the user using video. As such, they would like their call preferentially routed to the device which supports video. To do this, the INVITE request can contain parameters that express a preference for routing to a device with the specified capabilities [11].

- A network application would like to asynchronously send information to a user agent in a MESSAGE [16] request. However, before sending it, they would like to know if the UA has the capabilities necessary to receive the message. To do that, they would ideally query a user database managed by the domain which holds such information. Population of such a database would require that a UA convey its capabilities as part of its registration. Thus, there is a need for conveying capabilities in REGISTER requests.

SIP has some support for expression of capabilities. The Allow, Accept, Accept-Language, and Supported header fields convey some information about the capabilities of a user agent. However, these header fields convey only a small part of the information that is needed. They do not provide a general framework for expression of capabilities. Furthermore, they only specify capabilities indirectly; the header fields really indicate the capabilities of the UA as they apply to this request. SIP also has no ability to convey characteristics, that is, information that describes a UA.

As a result, this specification provides a more general framework for an indication of capabilities and characteristics in SIP. Capability and characteristic information about a UA is carried as parameters of the Contact header field. These parameters can be used within REGISTER requests and responses, OPTIONS responses, and requests and responses that create dialogs (such as INVITE).

2. Terminology

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 BCP 14, RFC 2119 [2] and indicate requirement levels for compliant implementations.

3. Definitions

Feature: As defined in RFC 2703 [17], a piece of information about the media handling properties of a message passing system component or of a data resource. For example, the SIP methods supported by a UA represent a feature.

Feature Tag: As defined in RFC 2703 [17], a feature tag is a name that identifies a feature. An example is "sip.methods".

Media Feature: As defined in RFC 2703, [17], a media feature is information that indicates facilities assumed to be available for the message content to be properly rendered or otherwise presented. Media features are not intended to include information that affects message transmission.

In the context of this specification, a media feature is information that indicates facilities for handling SIP requests, rather than specifically for content. In that sense, it is used synonymously with feature.

Feature Collection: As defined in RFC 2533 [4], a feature collection is a collection of different media features and associated values. This might be viewed as describing a specific rendering of a specific instance of a document or resource by a specific recipient.

Feature Set: As defined in RFC 2703 [17], a feature set is information about a sender, recipient, or other participant in a message transfer which describes the set of features that it can handle. Where a ’feature’ describes a single identified attribute of a resource, a ’feature set’ describes a full set of possible attributes.

Feature Parameters: A set of SIP header field parameters that can appear in the Contact header field. The feature parameters represent an encoding of a feature set. Each set of feature parameters maps to a feature set predicate.

Capability: As defined in RFC 2703 [17], a capability is an attribute of a sender or receiver (often the receiver) which indicates an ability to generate or process a particular type of message content. A capability is distinct from a characteristic in that a capability may or may not be utilized in any particular call, whereas a characteristic is a non-negotiable property of a UA. SIP itself will often negotiate whether or not capabilities are used in a call.

Characteristic: A characteristic is like a capability, but describes an aspect of a UA which is not negotiable. As an example, whether or not a UA is a mobile phone is a characteristic, not a capability. The semantics of this specification do not differentiate between capability and characteristic, but the distinction is useful for illustrative purposes. Indeed, in the text below, when we say "capability", it refers to both capabilities and characteristics, unless the text explicitly says otherwise.

Filter: A single expression in a feature set predicate.

Simple Filter: An expression in a feature set predicate which is a comparison (equality or inequality) of a feature tag against a feature value.

Disjunction: A boolean OR operation across some number of terms.

Conjunction: A boolean AND operation across some number of terms.

Predicate: A boolean expression.

Feature Set Predicate: From RFC 2533 [4], a feature set predicate is a function of an arbitrary feature collection value which returns a Boolean result. A TRUE result is taken to mean that the corresponding feature collection belongs to some set of media feature handling capabilities defined by this predicate.

Contact Predicate: The feature set predicate associated with a URI registered in the Contact header field of a REGISTER request. The contact predicate is derived from the feature parameters in the Contact header field.

4. Usage of the Content Negotiation Framework

This specification makes heavy use of the terminology and concepts in the content negotiation work carried out within the IETF, and documented in several RFCs. The ones relevant to this specification are RFC 2506 [3], which provides a template for registering media feature tags, RFC 2533 [4], which presents a syntax and matching algorithm for media feature sets, RFC 2738 [5], which provides a minor update to RFC 2533, and RFC 2703 [17], which provides a general framework for content negotiation.

In case the reader does not have the time to read those specifications, Appendix A provides a brief overview of the concepts and terminology in those documents that is critical for understanding this specification.

Since the content negotiation work was primarily meant to apply to documents or other resources with a set of possible renderings, it is not immediately apparent how it is used to model SIP user agents. A feature set is composed of a set of feature collections, each of which represents a specific rendering supported by the entity described by the feature set. In the context of a SIP user agent, a feature collection represents an instantaneous modality. That is, if you look at the run time processing of a SIP UA and take a snapshot in time, the feature collection describes what it is doing at that very instant.

This model is important, since it provides guidance on how to determine whether something is a value for a particular feature tag, or a feature tag by itself. If two properties can be exhibited by a UA simultaneously so that both are present in an instantaneous modality, they need to be represented by separate media feature tags. For example, a UA may be able to support some number of media types - audio, video, and control. Should each of these be different values for a single "media-types" feature tag, or should each of them be a separate boolean feature tag? The model provides the answer. Since, at any instance in time, a UA could be handling both audio and video, they need to be separate media feature tags. However, the SIP methods supported by a UA can each be represented as different values for the same media feature tag (the "sip.methods" tag), because fundamentally, a UA processes a single request at a time. It may be multi-threading, so that it appears that this is not so, but at a purely functional level, it is true.

Clearly, there are weaknesses in this model, but it serves as a useful guideline for applying the concepts of RFC 2533 to the problem at hand.

5. Computing Capabilities

To construct a set of Contact header field parameters that indicate capabilities, a UA constructs a feature predicate for that contact. This process is described in terms of RFC 2533 [4] (and its minor update, RFC 2738 [5]) syntax and constructs, followed by a conversion to the syntax used in this specification. However, this represents a logical flow of processing. There is no requirement that an implementation actually use RFC 2533 syntax as an intermediate step.

A UA MAY use any feature tags that are registered through IANA in the SIP tree (Established in Section 12.1), IETF, or global trees [3]; this document registers several into the SIP tree. The feature tags discussed in this specification are referred to as base tags. While other tags can be used, in order to identify them as feature parameters (as opposed to parameters for another SIP extension), they are encoded with a leading "+" sign in the Contact header field. It is also permissible to use the URI tree [3] for expressing vendor-specific feature tags. Feature tags in any other trees created through IANA MAY also be used.

When using the "sip.methods" feature tag, a UA MUST NOT include values that correspond to methods not standardized in IETF standards track RFCs. When using the "sip.events" feature tag, a UA MUST NOT include values that correspond to event packages not standardized in IETF standards track RFCs. When using the "sip.schemes" feature tag, a UA MUST NOT include values that correspond to schemes not standardized in IETF standards track RFCs. When using the "sip.extensions" feature tag, a UA MUST NOT include values that correspond to option tags not standardized in IETF standards track RFCs.

Note that the "sip.schemes" feature tag does not indicate the scheme of the registered URI. Rather, it indicates schemes that a UA is capable of sending requests to, should such a URI be received in a web page or Contact header field of a redirect response.

It is RECOMMENDED that a UA provide complete information in its contact predicate. That is, it SHOULD provide information on as many feature tags as possible. The mechanisms in this specification work best when user agents register complete feature sets. Furthermore, when a UA registers values for a particular feature tag, it MUST list all values that it supports. For example, when including the "sip.methods" feature tag, a UA MUST list all methods it supports.

The contact predicate constructed by a UA MUST be an AND of terms (called a conjunction). Each term is either an OR (called a disjunction) of simple filters or negations of simple filters, or a single simple filter or negation of a single filter. In the case of a disjunction, each filter in the disjunction MUST indicate feature values for the same feature tag (i.e., the disjunction represents a set of values for a particular feature tag), while each element of the conjunction MUST be for a different feature tag. Each simple filter can be an equality, or in the case of numeric feature tags, an inequality or range. If a string (as defined in RFC 2533 [4]) is used as the value of a simple filter, that value MUST NOT include the "<" or ">" characters, the simple filter MUST NOT be negated, and it MUST be the only simple filter for that particular feature tag. This contact predicate is then converted to a list of feature parameters, following the procedure outlined below.

The contact predicate is a conjunction of terms. Each term indicates constraints on a single feature tag, and each term is represented by a separate feature parameter that will be present in the Contact header field. The syntax of this parameter depends on the feature tag. Each forward slash in the feature tag is converted to a single quote, and each colon are converted to an exclamation point. For the base tags - that is, those feature tags documented in this specification (sip.audio, sip.automata, sip.class, sip.duplex, sip.data, sip.control, sip.mobility, sip.description, sip.events, sip.priority, sip.methods, sip.extensions, sip.schemes, sip.application, sip.video, language, type, sip.isfocus, sip.actor and sip.text), the leading "sip.", if present, is stripped. For feature tags not in this list, the leading "sip." MUST NOT be stripped if present, and indeed, a plus sign ("+") MUST be added as the first character of the Contact header field parameter. The result is the feature parameter name. As a result of these rules, the base tags appear "naked" in the Contact header field - they have neither a "+" nor a "sip." prefix. All other tags will always have a leading "+" when present in the Contact header field, and will additionally have a "sip." if the tag is in the SIP tree.

The value of the feature parameter depends on the term of the conjunction. If the term is a boolean expression with a value of true, i.e., (sip.audio=TRUE), the contact parameter has no value. If the term of the conjunction is a disjunction, the value of the contact parameter is a quoted string. The quoted string is a comma separated list of strings, each one derived from one of the terms in the disjunction. If the term of the conjunction is a negation, the value of the contact parameter is a quoted string. The quoted string begins with an exclamation point (!), and the remainder is constructed from the expression being negated.

The remaining operation is to compute a string from a primitive filter. If the filter is a simple filter that is performing a numeric comparison, the string starts with an octothorpe (#), followed by the comparator in the filter (=, >=, or <=), followed by the value from the filter. If the value from the filter is expressed in rational form (X / Y), then X and Y are divided, yielding a decimal number, and this decimal number is output to the string.

> RFC 2533 uses a fractional notation to describe rational numbers. This specification uses a decimal form. The above text merely converts between the two representations. Practically speaking, this conversion is not needed since the numbers are the same in either case. However, it is described in case implementations wish to directly plug the predicates generated by the rules in this section into an RFC 2533 implementation.

If the filter is a range (foo=X..Y), the string is equal to X:Y, where X and Y have been converted from fractional numbers (A / B) to their decimal equivalent.

If the filter is an equality over a token or boolean, then that token or boolean value ("TRUE" or "FALSE") is output to the string.

If the filter is an equality over a quoted string, the output is a less than (<), followed by the quoted string, followed by a greater than (>).

As an example, this feature predicate:

```
(& (sip.mobility=fixed)
   (| (! (sip.events=presence)) (sip.events=message-summary))
   (| (language=en) (language=de))
   (sip.description="PC")
   (sip.newparam=TRUE)
   (rangeparam=-4..5125/1000))
```

would be converted into the following feature parameters:

```
mobility="fixed";events="!presence,message-summary";language="en,de"
   ;description="<PC>";+sip.newparam;+rangeparam="#-4:+5.125"
```

These feature tags would then appear as part of the Contact header field:

```
Contact: <sip:user@pc.example.com>
          ;mobility="fixed";events="!presence,message-summary"
          ;language="en,de";description="<PC>"
          ;+sip.newparam;+rangeparam="#-4:+5.125"
```

Notice how the leading "sip." was stripped from the sip.mobility, sip.events and sip.description feature tags before encoding them in the Contact header field. This is because these feature tags are amongst the base tags listed above. It is for this reason that these feature tags were not encoded with a leading "+" either. However, the sip.newparam feature tag was encoded with both the "+" and its leading "sip.", and the rangeparam was also encoded with a leading "+". This is because neither of these feature tags are defined in this specification. As such, the leading "sip." is not stripped off, and a "+" is added.

6. Expressing Capabilities in a Registration

When a UA registers, it can choose to indicate a feature set associated with a registered contact. Whether or not a UA does so depends on what the registered URI represents. If the registered URI represents a UA instance (the common case in registrations), a UA compliant to this specification SHOULD indicate a feature set using the mechanisms described here. If, however, the registered URI represents an address-of-record, or some other resource that is not representable by a single feature set, it SHOULD NOT include a feature set. As an example, if a user wishes to forward calls from sip:user1@example.com to sip:user2@example.org, it could generate a registration that looks like, in part:

```
REGISTER sip:example.com SIP/2.0
To: sip:user1@example.com
Contact: sip:user2@example.org
```

In this case, the registered contact is not identifying a UA, but rather, another address-of-record. In such a case, the registered contact would not indicate a feature set.

However, in some cases, a UA may wish to express feature parameters for an address-of-record. One example is an AOR which represents a multiplicity of devices in a home network, and routes to a proxy server in the user’s home. Since all devices in the home are for personal use, the AOR itself can be described with the ;class="personal" feature parameter. A registration that forwards calls to this home AOR could make use of that feature parameter. Generally speaking, a feature parameter can only be associated with an address-of-record if all devices bound to that address-of-record share the exact same set of values for that feature parameter.

Similarly, in some cases, a UA can exhibit one characteristic or another, but the characteristic is not known in advance. For example, a UA could represent a device that is a phone with an embedded answering machine. The ideal way to treat such devices is to model them as if they were actually a proxy fronting two devices - a phone (which is never an answering machine), and an answering machine (which is never a phone). The registration from this device would be constructed as if it were an AOR, as per the procedures above. Generally, this means that, unless the characteristic is identical between the logical devices, that characteristic will not be present in any registration generated by the actual device.

The remainder of this section assumes that a UA would like to associate a feature set with a contact that it is registering. This feature set is constructed and converted to a series of Contact header field parameters, as described in Section 5, and those feature parameters are added to the Contact header field value containing the URI to which the parameters apply. The Allow, Accept, Accept-Language and Allow-Events [9] header fields are allowed in REGISTER requests, and also indicate capabilities. However, their semantic in REGISTER is different, indicating capabilities, used by the registrar, for generation of the response. As such, they are not a substitute or an alternate for the Contact feature parameters, which indicate the capabilities of the UA generally speaking.

The REGISTER request MAY contain a Require header field with the value "pref" if the client wants to be sure that the registrar understands the extensions defined in this specification. This means that the registrar will store the feature parameters, and make them available to elements accessing the location service within the domain. In the absence of the Require header field, a registrar that does not understand this extension will simply ignore the Contact header field parameters.

If a UA registers against multiple separate addresses-of-record, and the contacts registered for each have different capabilities, a UA MUST use different URIs in each registration. This allows the UA to uniquely determine the feature set that is associated with the request URI of an incoming request.

As an example, a voicemail server that is a UA that supports audio and video media types and is not mobile would construct a feature predicate like this:

```
(& (sip.audio=TRUE)
   (sip.video=TRUE)
   (sip.actor=msg-taker)
   (sip.automata=TRUE)
   (sip.mobility=fixed)
   (| (sip.methods=INVITE) (sip.methods=BYE) (sip.methods=OPTIONS)
      (sip.methods=ACK) (sip.methods=CANCEL)))
```

These would be converted into feature parameters and included in the REGISTER request:

```
REGISTER sip:example.com SIP/2.0
From: sip:user@example.com;tag=asd98
To: sip:user@example.com
Call-ID: hh89as0d-asd88jkk@host.example.com
CSeq: 9987 REGISTER
Max-Forwards: 70
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds8
Contact: <sip:user@host.example.com>;audio;video
  ;actor="msg-taker";automata;mobility="fixed"
  ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
Content-Length: 0
```

Note that a voicemail server is usually an automata and a message taker.

When a UAC refreshes its registration, it MUST include its feature parameters in that refresh if it wishes for them to remain active. Furthermore, when a registrar returns a 200 OK response to a REGISTER request, each Contact header field value MUST include all of the feature parameters associated with that URI.

7. Indicating Feature Sets in Remote Target URIs

Target refresh requests and responses are used to establish and modify the remote target URI in a dialog. The remote target URI is conveyed in the Contact header field. A UAC or UAS MAY add feature parameters to the Contact header field value in target refresh requests and responses for the purpose of indicating the capabilities of the UA. To do that, it constructs a set of feature parameters according to Section 5. These are then added as Contact header field parameters in the request or response.

The feature parameters can be included in both initial requests and mid-dialog requests, and MAY change mid-dialog to signal a change in UA capabilities.

There is overlap in the callee capabilities mechanism with the Allow, Accept, Accept-Language, and Allow-Events [9] header fields, which can also be used in target refresh requests. Specifically, the Allow header field and "sip.methods" feature tag indicate the same information. The Accept header field and the "type" feature tag indicate the same information. The Accept-Language header field and the "language" feature tag indicate the same information. The Allow-Events header field and the "sip.events" feature tag indicate the same information. It is possible that other header fields and feature tags defined in the future may also overlap. When there exists a feature tag that describes a capability that can also be represented with a SIP header field, a UA MUST use the header field to describe the capability. A UA receiving a message that contains both the header field and the feature tag MUST use the header field, and not the feature tag.

8. OPTIONS Processing

When a UAS compliant to this specification receives an OPTIONS request, it MAY add feature parameters to the Contact header field in the OPTIONS response for the purpose of indicating the capabilities of the UA. To do that, it constructs a set of feature parameters according to Section 5. These are then added as Contact header field parameters in OPTIONS response. Indeed, if feature parameters were included in the registration generated by that UA, those same parameters SHOULD be used in the OPTIONS response.

The guidelines in Section 7 regarding the overlap of the various callee capabilities feature tags with SIP header fields applies to the generation of OPTIONS responses as well. In particular, they apply when a Contact header field is describing the UA which generated the OPTIONS response. When a Contact header field in the OPTIONS response is identifying a different UA, there is no overlap.

9. Contact Header Field

This specification extends the Contact header field. In particular, it allows for the Contact header field parameters to include feature-param. Feature-param is a feature parameter that describes a feature of the UA associated with the URI in the Contact header field. Feature parameters are identifiable because they either belong to the well known set of base feature tags, or they begin with a plus sign.

```
feature-param    =  enc-feature-tag [EQUAL LDQUOT (tag-value-list
                    / string-value ) RDQUOT]
enc-feature-tag  =  base-tags / other-tags
base-tags        =  "audio" / "automata" /
                    "class" / "duplex" / "data" /
                    "control" / "mobility" / "description" /
                    "events" / "priority" / "methods" /
                    "schemes" / "application" / "video" /
                    "language" / "type" / "isfocus" /
                    "actor" / "text" / "extensions"
other-tags      =  "+" ftag-name
ftag-name       =  ALPHA *( ALPHA / DIGIT / "!" / "'" /
                   "." / "-" / "%" )
```

```
   tag-value-list  =  tag-value *("," tag-value)
   tag-value       =  ["!"] (token-nobang / boolean / numeric)
   token-nobang    =  1*(alphanum / "-" / "." / "%" / "*"
                      / "_" / "+" / "`" / "'" / "~" )
   boolean         =  "TRUE" / "FALSE"
   numeric         =  "#" numeric-relation number
   numeric-relation  =  ">=" / "<=" / "=" / (number ":")
   number          =  [ "+" / "-" ] 1*DIGIT ["." 0*DIGIT]
   string-value    =  "<" *(qdtext-no-abkt / quoted-pair ) ">"
   qdtext-no-abkt  =  LWS / %x21 / %x23-3B / %x3D
                           / %x3F-5B / %x5D-7E / UTF8-NONASCII
```

Note that the tag-value-list uses an actual comma instead of the COMMA construction because it appears within a quoted string, where line folding cannot take place.

The production for qdtext can be found in RFC 3261 [1].

There are additional constraints on the usage of feature-param that cannot be represented in a BNF. There MUST only be one instance of any feature tag in feature-param. Any numbers present in a feature parameter MUST be representable using an ANSI C double.

The following production updates the one in RFC 3261 [1] for contact-params:

```
   contact-params    =  c-p-q / c-p-expires / feature-param
                        / contact-extension
```

10. Media Feature Tag Definitions

This specification defines an initial set of media feature tags for use with this specification. This section serves as the IANA registration for these feature tags, which are made into the SIP media feature tag tree. New media feature tags are registered in the IETF or global trees based on the process defined for feature tag registrations [3], or in the SIP tree based on the process defined in Section 12.1.

Any registered feature tags MAY be used with this specification. However, several existing ones appear to be particularly applicable. These include the language feature tag [6], which can be used to specify the language of the human or automata represented by the UA, and the type feature tag [7], which can be used to specify the MIME types that a SIP UA can receive in a SIP message. The audio, video, application, data, and control feature tags in the SIP tree (each of which indicate a media type, as defined in RFC 2327 [8]) are different. They do not indicate top level MIME types which can be received in SIP requests. Rather, they indicate media types that can be used in media streams, and as a result, match up with the types defined in RFC 2327 [8].

If a new SDP media type were to be defined, such as "message", a new feature tag registration SHOULD be created for it in the SIP tree. The name of the feature tag MUST equal "sip." concatenated with the name of the media type, unless there is an unlikely naming collision between the new media type and an existing feature tag registration. As a result, implementations can safely construct caller preferences and callee capabilities for the new media type before it is registered, as long as there is no naming conflict.

If a new media feature tag is registered with the intent of using that tag with this specification, the registration is done for the unencoded form of the tag (see Section 5). In other words, if a new feature tag "foo" is registered in the IETF tree, the IANA registration would be for the tag "foo" and not "+foo". Similarly, if a new feature tag "sip.gruu" is registered in the SIP tree, the IANA registration would be for the tag "sip.gruu" and not "+sip.gruu" or "gruu". As such, all registrations into the SIP tree will have the "sip." prefix.

The feature tags in this section are all registered in the SIP media feature tag tree created by Section 12.1.

10.1. Audio

Media feature tag name: sip.audio

ASN.1 Identifier: 1.3.6.1.8.4.1

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports audio as a streaming media type.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a phone that can support audio.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.2. Application

Media feature tag name: sip.application

ASN.1 Identifier: 1.3.6.1.8.4.2

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports application as a streaming media type. This feature tag exists primarily for completeness. Since so many MIME types are underneath application, indicating the ability to support applications provides little useful information.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application, for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a phone that can support a media control application.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.3. Data

Media feature tag name: sip.data

ASN.1 Identifier: 1.3.6.1.8.4.3

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports data as a streaming media type.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a phone that can support a data streaming application.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.4. Control

Media feature tag name: sip.control

ASN.1 Identifier: 1.3.6.1.8.4.4

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports control as a streaming media type.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a phone that can support a floor control application.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.5. Video

Media feature tag name: sip.video

ASN.1 Identifier: 1.3.6.1.8.4.5

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports video as a streaming media type.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a phone that can support video.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.6. Text

Media feature tag name: sip.text

ASN.1 Identifier: 1.3.6.1.8.4.6

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports text as a streaming media type.

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Routing a call to a phone that can support text.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.7. Automata

Media feature tag name: sip.automata

ASN.1 Identifier: 1.3.6.1.8.4.7

Summary of the media feature indicated by this tag: The sip.automata feature tag is a boolean value that indicates whether the UA represents an automata (such as a voicemail server, conference server, IVR, or recording device) or a human.

Values appropriate for use with this feature tag: Boolean. TRUE indicates that the UA represents an automata.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Refusing to communicate with an automata when it is known that automated services are unacceptable.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.8. Class

Media feature tag name: sip.class

ASN.1 Identifier: 1.3.6.1.8.4.8

Summary of the media feature indicated by this tag: This feature tag indicates the setting, business or personal, in which a communications device is used.

Values appropriate for use with this feature tag: Token with an equality relationship. Typical values include:

business: The device is used for business communications.

personal: The device is used for personal communications.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application, for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing between a business phone and a home phone.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.9. Duplex

Media feature tag name: sip.duplex

ASN.1 Identifier: 1.3.6.1.8.4.9

Summary of the media feature indicated by this tag: The sip.duplex media feature tag indicates whether a communications device can simultaneously send and receive media ("full"), alternate between sending and receiving ("half"), can only receive ("receive-only") or only send ("send-only").

Values appropriate for use with this feature tag: Token with an equality relationship. Typical values include:

full: The device can simultaneously send and receive media.

half: The device can alternate between sending and receiving media.

receive-only: The device can only receive media.

send-only: The device can only send media.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to communicate with a broadcast server, as opposed to a regular phone, when making a call to hear an announcement.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.10. Mobility

Media feature tag name: sip.mobility

ASN.1 Identifier: 1.3.6.1.8.4.10

Summary of the media feature indicated by this tag: The sip.mobility feature tag indicates whether the device is fixed (meaning that it is associated with a fixed point of contact with the network), or mobile (meaning that it is not associated with a fixed point of contact). Note that cordless phones are fixed, not mobile, based on this definition.

Values appropriate for use with this feature tag: Token with an equality relationship. Typical values include:

fixed: The device is stationary.

mobile: The device can move around with the user.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to communicate with a wireless phone instead of a desktop phone.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.11. Description

Media feature tag name: sip.description

ASN.1 Identifier: 1.3.6.1.8.4.11

Summary of the media feature indicated by this tag: The sip.description feature tag provides a textual description of the device.

Values appropriate for use with this feature tag: String with an equality relationship.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Indicating that a device is of a certain make and model.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.12. Event Packages

Media feature tag name: sip.events

ASN.1 Identifier: 1.3.6.1.8.4.12

Summary of the media feature indicated by this tag: Each value of the sip.events (note the plurality) feature tag indicates a SIP event package [9] supported by a SIP UA. The values for this tag equal the event package names that are registered by each event package.

Values appropriate for use with this feature tag: Token with an equality relationship. Values are taken from the IANA SIP Event types namespace registry.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to communicate with a server that supports the message waiting event package, such as a voicemail server [12].

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.13. Priority

Media feature tag name: sip.priority

ASN.1 Identifier: 1.3.6.1.8.4.13

Summary of the media feature indicated by this tag: The sip.priority feature tag indicates the call priorities the device is willing to handle. A value of X means that the device is willing to take requests with priority X and higher. This does not imply that a phone has to reject calls of lower priority. As always, the decision on handling of such calls is a matter of local policy.

Values appropriate for use with this feature tag: An integer. Each integral value corresponds to one of the possible values of the Priority header field as specified in SIP [1]. The mapping is defined as:

non-urgent: Integral value of 10. The device supports non-urgent calls.

normal: Integral value of 20. The device supports normal calls.

urgent: Integral value of 30. The device supports urgent calls.

emergency: Integral value of 40. The device supports calls in the case of an emergency situation.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to communicate with the emergency cell phone of a user.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.14. Methods

Media feature tag name: sip.methods

ASN.1 Identifier: 1.3.6.1.8.4.14

Summary of the media feature indicated by this tag: Each value of the sip.methods (note the plurality) feature tag indicates a SIP method supported by this UA. In this case, "supported" means that the UA can receive requests with this method. In that sense, it has the same connotation as the Allow header field.

Values appropriate for use with this feature tag: Token with an equality relationship. Values are taken from the Methods table defined in the IANA SIP parameters registry.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to communicate with a presence application on a PC, instead of a PC phone application.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.15. Extensions

Media feature tag name: sip.extensions

ASN.1 Identifier: 1.3.6.1.8.4.15

Summary of the media feature indicated by this tag: Each value of the sip.extensions feature tag (note the plurality) is a SIP extension (each of which is defined by an option-tag registered with IANA) that is understood by the UA. Understood, in this context, means that the option tag would be included in a Supported header field in a request.

Values appropriate for use with this feature tag: Token with an equality relationship. Values are taken from the option tags table in the IANA SIP parameters registry.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to communicate with a phone that supports quality of service preconditions instead of one that does not.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.16. Schemes

Media feature tag name: sip.schemes

ASN.1 Identifier: 1.3.6.1.8.4.16

Summary of the media feature indicated by this tag: Each value of the sip.schemes (note the plurality) media feature tag indicates a URI scheme [10] that is supported by a UA. Supported implies, for example, that the UA would know how to handle a URI of that scheme in the Contact header field of a redirect response.

Values appropriate for use with this feature tag: Token with an equality relationship. Values are taken from the IANA URI scheme registry.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Choosing to get redirected to a phone number when a called party is busy, rather than a web page.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.17. Actor

Media feature tag name: sip.actor

ASN.1 Identifier: 1.3.6.1.8.4.17

Summary of the media feature indicated by this tag: This feature tag indicates the type of entity that is available at this URI.

Values appropriate for use with this feature tag: Token with an equality relationship. The following values are defined:

principal: The device provides communication with the principal that is associated with the device. Often this will be a specific human being, but it can be an automata (for example, when calling a voice portal).

attendant: The device provides communication with an automaton or person that will act as an intermediary in contacting the principal associated with the device, or a substitute.

msg-taker: The device provides communication with an automaton or person that will take messages and deliver them to the principal.

information: The device provides communication with an automaton or person that will provide information about the principal.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Requesting that a call not be routed to voicemail.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

10.18. Is Focus

Media feature tag name: sip.isfocus

ASN.1 Identifier: 1.3.6.1.8.4.18

Summary of the media feature indicated by this tag: This feature tag indicates that the UA is a conference server, also known as a focus, and will mix together the media for all calls to the same URI [13].

Values appropriate for use with this feature tag: Boolean.

The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device, such as a phone or PDA.

Examples of typical use: Indicating to a UA that the server to which it has connected is a conference server.

Related standards or documents: RFC 3840

Security Considerations: Security considerations for this media feature tag are discussed in Section 11.1 of RFC 3840.

11. Security Considerations

11.1. Considerations for Media Feature Tags

This section discusses security considerations for the media feature tags, including, but not limited to, this specification.

The media feature tags defined in Section 10 reveal sensitive information about a user or the user agent they are describing. Some of the feature tags convey capability information about the agent - for example, the media types it can support, the SIP methods it can support, and the SIP extensions it can support. This capability information might be used for industrial espionage, for example, and so its protection may be important. Other attributes, such as the mobility, priority, and isfocus attributes, reveal characteristics of the user agent. These attributes are more sensitive than the capability information. They describe the way in which a user agent is utilized by a user, and thus reveal information about user preferences and the ways in which they want calls handled. Some feature tags, such as languages, reveal information about the user themself. As a result, applications which utilize these media feature tags SHOULD provide a means for ensuring their confidentiality.

The media feature tags can be used in ways which affect application behaviors. For example, the SIP caller preferences extension [11] allows for call routing decisions to be based on the values of these parameters. Therefore, if an attacker can modify the values of these feature tags, they may be able to affect the behavior of applications. As a result of this, applications which utilize these media feature tags SHOULD provide a means for ensuring their integrity. Similarly, media feature tags should only be trusted as valid when they come from the user or user agent described by those feature tags. As a result, mechanisms for conveying feature tags SHOULD provide a mechanism for guaranteeing authenticity.

11.2. Considerations for Registrations

As per the general requirements in Section 11.1, when media feature tags are carried in a registration, authenticity, confidentiality, and integrity need to be provided. To accomplish this, registrations containing capability information SHOULD be made by addressing the registration to a SIPS URI (in other words, the Request URI of the request would be sips:example.com when creating a registration in the example.com domain). Furthermore, the registrar SHOULD challenge the UA using digest over TLS, to verify its authenticity. The combination of TLS and digest provide integrity, confidentiality, and authenticity, as required.

It is not necessary for the Contact in the registration to itself contain a sips URI, since the feature tags are not carried in incoming requests sent to the UA.

11.3. Considerations for OPTIONS Responses

When including information on capabilities in a response to an OPTIONS request, a UA SHOULD verify with the user (either through a user interface or though prior configuration) whether or not capability information should be divulged to the requester. If the identity of the requester cannot be cryptographically verified (using digest or the SIP identity enhancements [15]), the user SHOULD also be alerted to this fact, and be allowed to choose whether such information should be divulged.

If the user does wish to reveal capability information to the requester, and wishes to guarantee its confidentiality, but the request did not arrive using SIPS, the UAS SHOULD redirect the request to a sips URI. This will cause the UAC to send the OPTIONS request using SIPS instead, and therefore provide confidentiality of any responses sent over the secure connections.

Furthermore, S/MIME MAY be used in the OPTIONS response. In that case, the capability information would be contained only in the secured S/MIME body, and not in the header fields of the OPTIONS response.

11.4. Considerations for Dialog Initiating Messages

When a UAS generates a response that will initiate a dialog, and they wish to include capability information in the Contact header field, the same considerations as described in Section 11.3 apply.

When a UAC generates a request that will initiate a dialog, it SHOULD obtain permission from the user (either through a user interface or apriori configuration) before including capability information in the Contact header field of the request. Confidentiality and integrity of the information SHOULD be provided using SIPS. S/MIME MAY be used.

12. IANA Considerations

There are a number of IANA considerations associated with this specification.

12.1. SIP Media Feature Tag Registration Tree

This specification serves to create a new media feature tag registration tree, per the guidelines of Section 3.1.4 of RFC 2506 [3]. The name of this tree is the "SIP Media Feature Tag Registration Tree", and its prefix is "sip.". It is used for the registration of media feature tags that are applicable to the Session

Initiation Protocol, and whose meaning is only defined within that usage.

The addition of entries into this registry occurs through IETF consensus, as defined in RFC 2434 [18]. This requires the publication of an RFC that contains the registration. The information required in the registration is identical to the IETF tree. As such, specifications adding entries to the registry should use the template provided in Section 3.4 of RFC 2506. Note that all media feature tags registered in the SIP tree will have names with a prefix of "sip.". No leading "+" is used in the registrations in any of the media feature tag trees.

12.2. Media Feature Tags

This specification registers a number of new Media feature tags according to the procedures of RFC 2506 [3]. These registrations are all made in the newly created SIP tree for media feature tags. These registrations are:

sip.audio: The information for registering the sip.audio media feature tag is contained in Section 10.1.

sip.application: The information for registering the sip.application media feature tag is contained in Section 10.2.

sip.data: The information for registering the sip.data media feature tag is contained in Section 10.3.

sip.control: The information for registering the sip.control media feature tag is contained in Section 10.4.

sip.video: The information for registering the sip.video media feature tag is contained in Section 10.5.

sip.text: The information for registering the sip.text media feature tag is contained in Section 10.6.

sip.automata: The information for registering the sip.automata media feature tag is contained in Section 10.7.

sip.class: The information for registering the sip.class media feature tag is contained in Section 10.8.

sip.duplex: The information for registering the sip.duplex media feature tag is contained in Section 10.9.

sip.mobility: The information for registering the sip.mobility media feature tag is contained in Section 10.10.

sip.description: The information for registering the sip.description media feature tag is contained in Section 10.11.

sip.events: The information for registering the sip.events media feature tag is contained in Section 10.12.

sip.priority: The information for registering the sip.priority media feature tag is contained in Section 10.13.

sip.methods: The information for registering the sip.methods media feature tag is contained in Section 10.14.

sip.extensions: The information for registering the sip.extensions media feature tag is contained in Section 10.15.

sip.schemes: The information for registering the sip.schemes media feature tag is contained in Section 10.16.

sip.actor: The information for registering the sip.actor media feature tag is contained in Section 10.17.

sip.isfocus: The information for registering the sip.isfocus media feature tag is contained in Section 10.18.

12.3. SIP Option Tag

This specification registers a single SIP option tag, pref. The required information for this registration, as specified in RFC 3261 [1], is:

Name: pref

Description: This option tag is used in a Require header field of a registration to ensure that the registrar supports the caller preferences extensions.

13. Acknowledgments

The initial set of media feature tags used by this specification were influenced by Scott Petrack's CMA design. Jonathan Lennox, Bob Penfield, Ben Campbell, Mary Barnes, Rohan Mahy, and John Hearty provided helpful comments. Graham Klyne provided assistance on the usage of RFC 2533. Thanks to Allison Mankin for her comments and support, and to Ted Hardie for his guidance on usage of the media feature tags.

14. References

14.1. Normative References

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

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

[3] Holtman, K., Mutz, A., and T. Hardie, "Media Feature Tag Registration Procedure", BCP 31, RFC 2506, March 1999.

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

[5] Klyne, G., "Corrections to "A Syntax for Describing Media Feature Sets"", RFC 2738, December 1999.

[6] Hoffman, P., "Registration of Charset and Languages Media Features Tags", RFC 2987, November 2000.

[7] Klyne, G., "MIME Content Types in Media Feature Expressions", RFC 2913, September 2000.

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

[9] Roach, A.B., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[10] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

14.2. Informative References

[11] Rosenberg, J., Schulzrinne, H. and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.

[12] Mahy, R., "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)", RFC 3842, August 2004.

[13] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", Work in Progress, May 2003.

[14] Howes, T. and M. Smith, "LDAP: String Representation of Search Filters", Work in Progress, March 2003.

[15] Peterson, J., "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", Work in Progress, March 2003.

[16] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[17] Klyne, G., "Protocol-independent Content Negotiation Framework", RFC 2703, September 1999.

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

Appendix A. Overview of RFC 2533

This section provides a brief overview of RFC 2533 and related specifications that form the content negotiation framework. This section does not represent normative behavior. In the event of any conflict between the tutorial material here and the normative text in RFC 2533, RFC 2533 takes precedence.

A critical concept in the framework is that of a feature set. A feature set is information about an entity (in our case, a UA), which describes a set of features it can handle. A feature set can be thought of as a region in N-dimensional space. Each dimension in this space is a different media feature, identified by a media feature tag. For example, one dimension (or axis) might represent languages, another might represent methods, and another, MIME types. A feature collection represents a single point in this space. It represents a particular rendering or instance of an entity (in our case, a UA). For example, a "rendering" of a UA would define an instantaneous mode of operation that it can support. One such rendering would be processing the INVITE method, which carried the application/sdp MIME type, sent to a UA for a user that is speaking English.

A feature set can therefore be defined as a set of feature collections. In other words, a feature set is a region of N-dimensional feature-space, that region being defined by the set of points - feature collections - that make up the space. If a particular feature collection is in the space, it means that the rendering described by that feature collection is supported by the device with that feature set.

How does one represent a feature set? There are many ways to describe an N-dimensional space. One way is to identify mathematical functions which identify its contours. Clearly, that is too complex to be useful. The solution taken in RFC 2533 is to define the space with a feature set predicate. A feature predicate defines a relation over an N-dimensional space; its input is any point in that space (i.e., a feature collection), and is true for all points that are in the region thus defined.

RFC 2533 describes a syntax for writing down these N-dimensional boolean functions, borrowed from LDAP [14]. It uses a prolog-style syntax which is fairly self-explanatory. This representation is called a feature set predicate. The base unit of the predicate is a filter, which is a boolean expression encased in round brackets. A filter can be complex, where it contains conjunctions and disjunctions of other filters, or it can be simple. A simple filter is one that expresses a comparison operation on a single media feature tag.

For example, consider the feature set predicate:

```
(& (foo=A)
   (bar=B)
   (| (baz=C) (& (baz=D) (bif=E))))
```

This defines a function over four media features - foo, bar, baz, and bif. Any point in feature space with foo equal to A, bar equal to B, and baz equal to either C or D, and bif equal to E, is in the feature set defined by this feature set predicate.

Note that the predicate doesn't say anything about the number of dimensions in feature space. The predicate operates on a feature space of any number of dimensions, but only those dimensions labeled foo, bar, baz, and bif matter. The result is that values of other media features don't matter. The feature collection {foo=A,bar=B,baz=C,bop=F} is in the feature set described by the predicate, even though the media feature tag "bop" isn't mentioned. Feature set predicates are therefore inclusive by default. A feature collection is present unless the boolean predicate rules it out. This was a conscious design choice in RFC 2533.

RFC 2533 also talks about matching a preference with a capability set. This is accomplished by representing both with a feature set. A preference is a feature set - its a specification of a number of feature collections, any one of which would satisfy the requirements of the sender. A capability is also a feature set - its a specification of the feature collections that the recipient supports. There is a match when the spaces defined by both feature sets overlap. When there is overlap, there exists at least one feature collection that exists in both feature sets, and therefore a modality or rendering desired by the sender which is supported by the recipient.

This leads directly to the definition of a match. Two feature sets match if there exists at least one feature collection present in both feature sets.

Computing a match for two general feature set predicates is not easy. Section 5 of RFC 2533 presents an algorithm for doing it by expanding an arbitrary expression into disjunctive normal form. However, the feature set predicates used by this specification are constrained. They are always in conjunctive normal form, with each term in the conjunction describing values for different media features. This makes computation of a match easy. It is computed independently for each media feature, and then the feature sets overlap if media features specified in both sets overlap. Computing the overlap of a single media feature is very straightforward, and is a simple matter of computing whether two finite sets overlap.

Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027
US

EMail: schulzrinne@cs.columbia.edu
URI: http://www.cs.columbia.edu/~hgs

Paul Kyzivat
Cisco Systems
1414 Massachusetts Avenue
BXB500 C2-2
Boxboro, MA 01719
US

EMail: pkyzivat@cisco.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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