SIP
Internet-Draft
Expires: April 21, 2004

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia University
P. Kyzivat
Cisco Systems
October 22, 2003

Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)
draft-ietf-sip-callee-caps-01

Status of this Memo

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 [15] request. However, before sending it, they would like to know if the UA has the capabilites 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 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 RFC 2119 [2] and indicate requirement levels for compliant implementations.

3. Definitions

Feature: As defined in RFC 2703 [16], 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 [16], a feature tag is a name that identifies a feature. An example is ``sip.methods''.

Media Feature: As defined in RFC 2703, [16], 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 [16], 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 [16], 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 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 [16] 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 of 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 which 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 (Section 12.1), IETF, or global trees [3]; this document registers several into the SIP and IETF trees. 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 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), and 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. 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. The name of this parameter depends on the feature tag. Any forward slashes in the feature tag are converted to a single quote, and any colons are converted to an exclamation point. If the feature tag name is not amongst the base tags specified in Section 9, a plus sign is added to the front of the feature parameter name. The plus sign MUST NOT be added if the feature tag name is amongst the base tags. If the feature tag name is amongst the base tags, and it is present in the SIP registry, the leading "sip." is removed from the name. The result is the feature parameter name.

The value of the feature parameter depends on the the term of the conjunction. If the term is a boolean expression with value of true, i.e., (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 (i.e., no and, or, or nots). 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 use 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, feature predicate:

```
(& (sip.mobility=fixed)
   (| (! (sip.events=presence)) (sip.events=winfo))
   (| (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,winfo";language="en,de"
 ;description="<PC>";+sip.newparam;+rangeparam="#-4:+5.125"
```

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 mutliplicity 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 "sip.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 it is not known in advance which. 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 the Contact header field value containing the URI that the parameters apply to.

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 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 is so that the UA can uniquely determine the feature set that is associated with the request URI of an incoming request.

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

```
(& (audio=TRUE)
   (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.

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 the 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.

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"
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 ">"
```

Note that the tag-value-list uses an actual comma instead of the COMMA construction. Thats 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 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 split between the IETF and SIP media feature tag trees. New media feature tags SHOULD be registered with IANA, based on the process defined for feature tag registrations [3]. Those registrations can occur in either the IETF tree, or the SIP tree, depending on the scope of applicability of the media feature tags.

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 of the media formats supported by the UA. However, the usage of the audio, video, application, data and control feature tags (each of which indicate a media type, as defined in RFC 2327 [8]) supported by the UA are preferred to indicating support for specific media formats. When the type feature tag is present, there SHOULD also be a feature tag present for the its top-level MIME type with a value of TRUE. In other words, if a UA indicates in a registration that it supports the video/H263 MIME type, it should also indicate that it supports video generally:

```
Contact: sip:192.0.2.1;type="video/H263";video="TRUE"
```

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 IETF tree. The name of the feature tag MUST equal that 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 of this, 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 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".

10.1 Feature Tags in the IETF Tree

The feature tags in this section are all registered into the IETF

media feature tag tree.

10.1.1 Audio

Media feature tag name: audio

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports audio as a 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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.1.2 Application

Media feature tag name: application

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports application as a 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. In most cases, the concrete MIME type is a better parameter to use in a predicate representing a preference.

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 supports gaming application.

Related standards or documents: RFC XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.1.3 Data

Media feature tag name: data

ASN.1 Identifier: New assignment by IANA.

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

Related standards or documents: RFC XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.1.4 Control

Media feature tag name: control

ASN.1 Identifier: New assignment by IANA.

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

Related standards or documents: RFC XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.1.5 Video

Media feature tag name: video

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: This feature tag indicates that the device supports video as a 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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2 Feature Tags in the SIP Tree

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

10.2.1 Automata

Media feature tag name: sip.automata

ASN.1 Identifier: New assignment by IANA.

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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.2 Class

Media feature tag name: sip.class

ASN.1 Identifier: New assignment by IANA.

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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.3 Duplex

Media feature tag name: sip.duplex

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: The sip.duplex media feature tag lists 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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.4 Mobility

Media feature tag name: sip.mobility

ASN.1 Identifier: New assignment by IANA.

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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.5 Description

Media feature tag name: sip.description

ASN.1 Identifier: New assignment by IANA.

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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.6 Event Packages

Media feature tag name: sip.events

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: The event packages [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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.7 Priority

Media feature tag name: sip.priority

ASN.1 Identifier: New assignment by IANA.

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.

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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.8 Methods

Media feature tag name: sip.methods

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: The sip.methods (note the plurality) feature tag indicates the SIP methods 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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.9 Extensions

Media feature tag name: sip.extensions

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: The sip.extensions feature tag is a list of SIP extensions (each of which is defined by an option-tag registered with IANA) that are 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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.10 Schemes

Media feature tag name: sip.schemes

ASN.1 Identifier: New assignment by IANA.

Summary of the media feature indicated by this tag: The set of URI schemes [10] that are 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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.11 Actor

Media feature tag name: sip.actor

ASN.1 Identifier: New assignment by IANA.

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 XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

10.2.12 Is Focus

Media feature tag name: sip.isfocus

ASN.1 Identifier: New assignment by IANA.

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 it has connected to is a conference server.

Related standards or documents: RFC XXXX [[Note to IANA: Please replace XXXX with the RFC number of this specification.]]

11. Security Considerations

Feature sets contained in REGISTER requests can reveal sensitive information about a user or UA (for example, the languages spoken). If this information is sensitive, confidentiality SHOULD be provided by using the SIPS URI scheme, as described in RFC 3261 [1].

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 RFC2506 [3]. The name of this tree is the "SIP Media Feature Tag Registration Tree", and its prefix is "sip.". It is used for 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 requires publication of a standards track 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.

12.2 Media Feature Tags

This specification registers a number of new Media feature tags according to the procedures of RFC 2506 [3]. Some of these registrations (those of Section 10.1) are made into the IETF tree for media feature tags, and the others (Section 10.2) are made into the newly created SIP tree for media feature tags.

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.

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., "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.

Informative References

[11] Rosenberg, J., Schulzrinne, H. and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", draft-ietf-sip-callerprefs-09 (work in progress), July 2003.

[12] Mahy, R., "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)", draft-ietf-sipping-mwi-03 (work in progress), July 2003.

[13] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-00 (work in progress), May 2003.

[14] Howes, T. and M. Smith, "LDAP: String Representation of Search Filters", draft-ietf-ldapbis-filter-04 (work in progress), March 2003.

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

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

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: pkzivat@cisco.com

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 either baz equal to C, or baz equal to 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.

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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