Network Working Group
Request for Comments: 2938
Updates: 2533
Category: Standards Track

G. Klyne
Content Technologies
L. Masinter
AT&T

September 2000

Identifying Composite Media Features

Status of this Memo

Copyright Notice

Abstract

In RFC 2533, an expression format is presented for describing media feature capabilities as a combination of simple media feature tags.

This document describes an abbreviated format for a composite media feature set, based upon a hash of the feature expression describing that composite.

Table of Contents

1. Introduction

In "A Syntax for Describing Media Feature Sets" [1], an expression format is presented for describing media feature capabilities as a combination of simple media feature tags [2].

This document proposes an abbreviated format for a composite media feature set, based upon a hash of the feature expression describing that composite.

This memo extends and builds upon the expression syntax described in RFC 2533 [1], and it is assumed that the reader is familiar with the interpretation of feature set expressions described there.

1.1 Organization of this document

Section 2 sets out some of the background and goals for feature set references.

Section 3 presents a syntax for feature set references, and describes how they are related to feature set expressions.

1.2 Terminology and document conventions

This section defines a number of terms and other document conventions, which are used with specific meaning in this memo. The terms are listed in alphabetical order.

dereference
: the act of replacing a feature set reference with its corresponding feature set expression. Also called "resolution".

feature set
: some set of media features described by a media feature assertion, as described in "A Syntax for Describing Media Feature Sets" [1]. (See that memo for a more formal definition of this term.)

feature set expression
: a string that describes some feature set, formulated according to the rules in "A Syntax for Describing Media feature sets" [1] (and possibly extended by other specifications).

feature set reference
: a brief construct that references some feature set. (See also: "dereference".)

feature set tag
: a name that conforms to the syntax of a feature tag [2] that is used to denote a feature set rather than a single feature.

resolution
: (See "dereference").

This specification uses syntax notation and conventions described in RFC 2234, "Augmented BNF for Syntax Specifications: ABNF" [3].

> NOTE: Comments like this provide additional nonessential information about the rationale behind this document. Such information is not needed for building a conformant implementation, but may help those who wish to understand the design in greater depth.

2. Motivation and goals

The range of media feature capabilities of a message handling system can be quite extensive, and the corresponding feature set expression [1] can reach a significant size.

A requirement has been identified to allow recurring feature sets to be identified by a single reference value, which can be combined with other elements in a feature set expression. It is anticipated that mechanisms will be provided that allow the recipient of such a feature set reference to discover the corresponding feature set expression, but any such mechanism is beyond the scope of this specification.

Thus, the goals for this proposal are:

- to provide an abbreviated form for referencing an arbitrary feature set expression.
- the meaning of (i.e., the corresponding feature set expression) a feature set reference should be independent of any particular mechanism that may be used to dereference it.
- to be able to verify whether a given feature set expression corresponds to some feature set reference without having to perform an explicit dereferencing operation (i.e., without incurring additional network traffic).

- for protocol processors that conform to RFC 2533 [1] to be able to sensibly handle a feature set reference without explicit knowledge of its meaning (i.e., the introduction of feature set references should not break existing feature expression processors). That is, the applicable interpretation and processing rules of RFC 2533 [1] apply equally to expressions containing feature set references.

  NOTE: This proposal does not attempt to address the "override" or "default" problem. (Where a feature set may be referenced and selectively modified.)

Some circumstances in which such an abbreviated form might be used include:

- A media feature expression that contains a repeated sub-expression. If the sub-expression is quite large, space can be saved by writing it out once, then using the abbreviated form to reference it.

- A capability that is common to a range of devices, such as a given class of fax machine where are large number of feature tags are involved, but only a small number of common feature sets. If the recipient understands, or can discover, that some abbreviation stands for a given feature set then feature expression size can be reduced by using the abbreviation.

  If feature set abbreviations are used in this way, it may be that they can be interpreted by a simple table lookup rather than full feature expression parsing. (Making this useful in practice will depend on crafting the feature subsets appropriately.)

Examples of such usage are given in section 4 of this memo.

This memo does not specify how a program that receives a feature set abbreviation should discover the corresponding feature set expression: see section 3.2.

3. Composite feature representation

This specification hinges on two central ideas:

- the use of auxiliary predicates (introduced in RFC 2533 [1]) to form the basis of a feature set identifier, and

- the use of a token based on a hash function computed over the referenced feature set expression.

A key reason to use a hash function to generate an identifier is to define a global name space without requiring a central naming authority. New feature set tags can be introduced by any party following the appropriate rules of formulation, without reference to any centralized authority.

Local resolution services may be needed to map feature set tags to their corresponding feature set expressions, but these are not able to vary the meaning of any given tag. Failure of a resolution service to return the correct expression is detectable by a calling application, which should reject any incorrect value supplied.

> NOTE: where a feature set reference is used, its meaning is defined by substitution of the referenced feature expression into the referencing expression. When all references have been thus replaced, the result is interpreted as a normal feature expression.
>
> In particular, if a referenced feature expression contains some feature tag that is also constrained by the referencing expression, the constraints are interpreted per RFC 2533 [1], without regard for their origin. E.g., (using some notation introduced below):
>
> ```
> (& (pix-x=100) (pix-y<=300)
>    (h.SBB5REAOMHC09CP2GM4V07PQP0) )
> ```
>
> where (h.SBB5REAOMHC09CP2GM4V07PQP0) resolves to:
>
> ```
> (& (pix-x<=200) (pix-y<=150) )
> ```
>
> yields a result equivalent to:
>
> ```
> (& (pix-x=100) (pix-y<=150) )
> ```

3.1 Feature set hashed reference format

This specification introduces a special form of auxiliary predicate name with the following syntax:

```
fname       = "h." 1*BASE32DIGIT
BASE32DIGIT = DIGIT
            / "A" / "B" / "C" / "D" / "E" / "F" / "G" / "H"
            / "I" / "J" / "K" / "L" / "M" / "N" / "O" / "P"
            / "Q" / "R" / "S" / "T" / "U" / "V"
```

The sequence of base-32 digits represents the value of a hash function calculated over the corresponding feature set expression (see following sections). Note that the above syntax allows upper- or lower-case letters for base-32 digits (per RFC 2234 [3]).

Thus, within a feature set expression, a hashed feature set reference would have the following form:

```
(h.123456789abcdefghijklmnopq)
```

3.1.1 Hash value calculation

The hash value is calculated using the MD5 algorithm [6] over the text of the referenced feature set expression subjected to certain normalizations. The feature expression must conform to the syntax given for 'filter' in RFC 2533 [1]:

```
filter = "(" filtercomp ")" *( ";" parameter )
```

The steps for calculating a hash value are:

1. Whitespace normalization: all spaces, CR, LF, TAB and any other layout control characters that may be embedded in the feature expression string, other than those contained within quoted strings, are removed (or ignored for the purpose of hash value computation).

2. Case normalization: all lower case letters in the feature expression, other than those contained within quoted strings, are converted to upper case. That is, unquoted characters with US-ASCII values 97 to 122 (decimal) are changed to corresponding characters in the range 65 to 90.

3. Hash computation: the MD5 algorithm, described in RFC 1321 [6], is applied to the normalized feature expression string (represented as a sequence of octets containing US-ASCII character codes; see also section 5).

   The result obtained in step 3 is a 128-bit (16 octet) value that is converted to a base-32 representation to form the feature set reference.

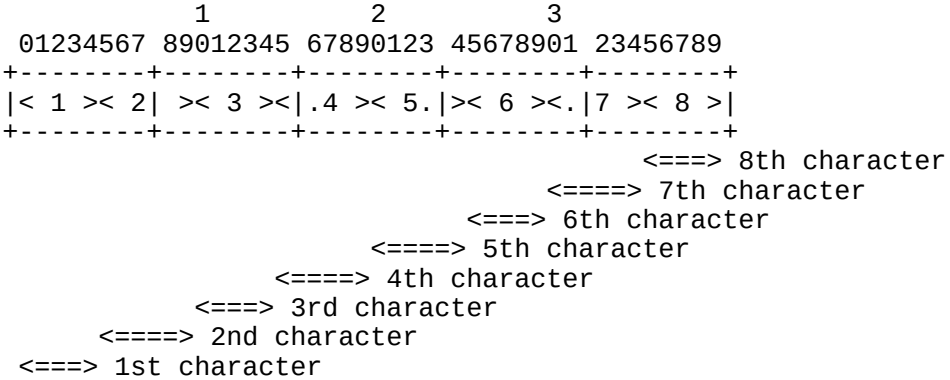
   NOTE: under some circumstances, removal of ALL whitespace may result in an invalid feature expression string. This should not be a problem as this is done only for the purpose of calculating a hash value, and significantly different feature expressions are expected to differ in ways other than their whitespace.

   NOTE: case normalization is deemed appropriate since feature tag and token matching is case insensitive.

3.1.2 Base-32 value representation

RFC 1321 [6] describes how to calculate an MD5 hash value that is a sequence of 16 octets. This is then required to be coded as a base-32 value, which is a sequence of base-32 digit characters.

Each successive character in a base-32 value represents 5 successive bits of the underlying octet sequence. Thus, each group of 8 characters represents a sequence of 5 octets (40 bits):



The value (i.e. sequence of bits) represented by each base-32 digit character is indicated by the following table:

"0"	0	"A"	10	"K"	20	"U"	30
"1"	1	"B"	11	"L"	21	"V"	31
"2"	2	"C"	12	"M"	22		
"3"	3	"D"	13	"N"	23		
"4"	4	"E"	14	"O"	24		
"5"	5	"F"	15	"P"	25		
"6"	6	"G"	16	"Q"	26		
"7"	7	"H"	17	"R"	27		
"8"	8	"I"	18	"S"	28		
"9"	9	"J"	19	"T"	29		

When encoding a base-32 value, each full group of 5 octets is represented by a sequence of 8 characters indicated above. If a group of less than 5 octets remain after this, they are encoded using as many additional characters as may be needed: 1, 2, 3 or 4 octets are encoded by 2, 4, 5 or 7 characters respectively. Any spare bits represented by the base-32 digit characters are selected to be zero.

When decoding a base-32 value, the reverse mapping is applied: each full group of 8 characters codes a sequence of 5 octets. A final group of 2, 4, 5 or 7 characters codes a sequence of 1, 2, 3 or 4 octets respectively. Any spare bits represented by the final group of characters are discarded.

Thus, for a 128-bit (16 octet) MD5 hash value, the first 15 octets are coded as 24 base 32 digit characters, and the final octet is coded by two characters.

> NOTE: Base64 representation (per MIME [4]) would be more compact (21 rather than 26 characters for the MD5 128-bit hash value), but an auxiliary predicate name is defined (by [1]) to have the same syntax as a feature tag, and the feature tag matching rules (per [2]) state that feature tag matching is case insensitive.
>
> Base36 representation was considered (i.e., using all letters "A"-"Z") but was not used because this would require extended precision multiplication and division operations to encode and decode the hash values.

3.2 Resolving feature set identifiers

This memo does not mandate any particular mechanism for dereferencing a feature set identifier. It is expected that specific dereferencing mechanisms will be specified for any application or protocol that uses them.

The following sections describe some ways that feature set dereferencing information may be incorporated into a feature set expression. These are based on auxiliary predicate definitions within a "where" clause [1].

When a hashed feature set reference is used, conformance to the hashing rules takes precedence over any other determination of the feature expression. Any expression, however obtained, may not be substituted for the hash-based reference unless it yields the correct hash value.

3.2.1 Query protocol

A protocol providing request/response type queries (e.g., HTTP, LDAP, etc.) might be set up to provide a resolution service.

Thus, a query to a server associated with the capabilities could be performed on the feature set identifier. The response returned would be a CONNEG expression; e.g.,

```
(h.SBB5REAOMHC09CP2GM4V07PQP0)
where
(h.SBB5REAOMHC09CP2GM4V07PQP0) :- (& (pix-x<=200) (pix-y<=150) )
end
```

or just:

```
(& (pix-x<=200) (pix-y<=150) )
```

This result would be combined with the original expression to obtain a result not including the hash based predicate.

This process might be further enhanced by using URN resolution mechanisms (e.g., DNS NAPTR [10]) to discover the resolution protocol and server.

3.2.2 Inline feature set details

In this case, a reference is resolved by including its definition inline in an expression.

The feature set expression associated with a reference value may be specified directly in a "where" clause, using the auxiliary predicate definition syntax [1]; e.g.,

```
(& (dpi=100) (h.SBB5REAOMHC09CP2GM4V07PQP0) )
where
(h.SBB5REAOMHC09CP2GM4V07PQP0) :- (& (pix-x<=200) (pix-y<=150) )
end
```

This form might be used on request (where the request mechanism is defined by the invoking application protocol), or when the originator believes the recipient may not understand the reference.

It is an error if the inline feature expression does not yield the hash value contained in auxiliary predicate name.

> NOTE: viewed in isolation, this format does not have any obvious value, in that the (h.xxx) form of auxiliary predicate could be replaced by any arbitrary name.
>
> It is anticipated that this form might be used as a follow-up response in a sequence along the lines of:
>
> ```
> A> Capabilities are:
>   (& (dpi=100) (h.SBB5REAOMHC09CP2GM4V07PQP0) )
> B> Do not understand:
>   (h.SBB5REAOMHC09CP2GM4V07PQP0)
> ```

```
        A> Capabilities are:
          (& (dpi=100) (h.SBB5REAOMHC09CP2GM4V07PQP0) )
          where
            (h.SBB5REAOMHC09CP2GM4V07PQP0) :- (& (pix-x<=200)
              (pix-y<=150) )
          end
```

4. Examples

The following are some examples of feature set expressions containing feature set references:

```
(& (dpi=100) (h.SBB5REAOMHC09CP2GM4V07PQP0) )

(& (dpi=100) (h.SBB5REAOMHC09CP2GM4V07PQP0) )
where
(h.SBB5REAOMHC09CP2GM4V07PQP0) :-
  (& (pix-x<=200) (pix-y<=150) )
end

(h.QGEOPMCF02P09QC016CEPU22FO)
where
(h.QGEOPMCF02P09QC016CEPU22FO) :-
 (| (& (ua-media=continuous) (dpi=200) (dpi-xyratio=200/100)
       (color=Binary) (paper-size=B4) (image-coding=MH) )
    (& (ua-media=continuous) (dpi=200) (dpi-xyratio=200/100)
       (color=Binary) (paper-size=B4) (image-coding=MR) )
    (& (ua-media=stationery) (dpi=300) (dpi-xyratio=1)
       (color=Binary) (paper-size=A4) (image-coding=JBIG) )
    (& (ua-media=transparency) (dpi=300) (dpi-xyratio=1)

       (color=Binary) (paper-size=A4) (image-coding=JBIG) ) )
end
```

The following examples are based on Internet fax work, and show how a feature-hash might be used to express the commonly-used features. A form of Internet fax system that is expected to be quite common is a so-called "simple mode" system, whose capabilities are described by the following feature expression:

```
(& (image-file-structure=TIFF-minimal)
  (MRC-mode=0)
  (color=Binary)
  (image-coding=MH) (MRC-mode=0)
  (| (& (dpi=204) (dpi-xyratio=[204/98,204/196]) )
     (& (dpi=200) (dpi-xyratio=[200/100,1]) ) )
  (size-x<=2150/254)
  (paper-size=A4)
```

```
      (ua-media=stationery) )
```

This might be expressed by the hash-based feature set identifier:

```
   (h.MSB955PVIRT1QOHET9AJT5JM3O)
```

The following example describes capabilities of a full-color Internet fax system. Note a number of feature values are applicable in common with '(color=grey)' and '(color=full)':

```
   (& (image-file-structure=TIFF)
      (MRC-mode=0)
      (| (& (color=Binary)
            (image-coding=[MH,MR,MMR])
            (| (& (dpi=204) (dpi-xyratio=[204/98,204/196]) )
               (& (dpi=200) (dpi-xyratio=[200/100,1]) )
               (& (dpi=300) (dpi-xyratio=1) ) ) )
         (& (color=grey)
            (image-coding=JPEG)
            (image-coding-constraint=JPEG-T4E)
            (color-levels<=256)
            (color-space=CIELAB)
            (color-illuminant=D50)
            (CIELAB-L-min>=0)
            (CIELAB-L-max<=100)
            (dpi=[100,200,300]) (dpi-xyratio=1) )
         (& (color=full)
            (image-coding=JPEG)
            (image-coding-constraint=JPEG-T4E)
            (color-subsampling=["1:1:1","4:1:1"])
            (color-levels<=16777216)
            (color-space=CIELAB)
            (color-illuminant=D50)
            (CIELAB-L-min>=0)
            (CIELAB-L-max<=100)
            (CIELAB-a-min>=-85)
            (CIELAB-a-max<=85)
            (CIELAB-b-min>=-75)
            (CIELAB-b-max<=125)
            (dpi=[100,200,300]) (dpi-xyratio=1) ) )
      (size-x<=2150/254)
      (paper-size=[letter,A4,B4]) )
      (ua-media=stationery) )
```

Separating out the common capabilities yields:

```
  (& (image-file-structure=TIFF)
     (MRC-mode=0)
     (| (& (color=Binary)
           (image-coding=[MH,MR,MMR])
           (| (& (dpi=204) (dpi-xyratio=[204/98,204/196]) )
              (& (dpi=200) (dpi-xyratio=[200/100,1]) )
              (& (dpi=300) (dpi-xyratio=1) ) ) )
        (& (color=grey)
           (color-levels<=256)
           (h.QVSEM8V2LMJ8VOR7V682J7079O) )
        (& (color=full)
           (color-subsampling=["1:1:1","4:1:1"])
           (color-levels<=16777216)
           (CIELAB-a-min>=-85)
           (CIELAB-a-max<=85)
           (CIELAB-b-min>=-75)
           (CIELAB-b-max<=125)
           (h.QVSEM8V2LMJ8VOR7V682J7079O) ) )
     (size-x<=2150/254)
     (paper-size=[letter,A4,B4]) )
     (ua-media=stationery) )
  where
  (h.QVSEM8V2LMJ8VOR7V682J7079O) :-
     (& (image-coding=JPEG)
        (image-coding-constraint=JPEG-T4E)
        (color-space=CIELAB)
        (color-illuminant=D50)
        (CIELAB-L-min>=0)
        (CIELAB-L-max<=100)
        (dpi=[100,200,300]) (dpi-xyratio=1) )
  end
```

5. Internationalization Considerations

Feature set expressions and URI strings are currently defined to consist of only characters from the US-ASCII repertoire [1,5]; under these circumstances this specification is not impacted by internationalization considerations (other than any already applicable to URIs [5]).

But, if future revisions of the feature set syntax permit non-US-ASCII characters (e.g. within quoted strings), then some canonical representation must be defined for the purposes of calculating hash values. One choice might be to use a UTF-8 equivalent representation as the basis for calculating the feature set hash. Another choice

might be to leave this as an application protocol issue (but this could lead to non-interoperable feature sets between different protocols).

Another conceivable issue is that of up-casing the feature expression in preparation for computing a hash value. This does not apply to the content of strings so is not likely to be an issue. But if changes are made that do permit non-US-ASCII characters in feature tags or token strings, consideration must be given to properly defining how case conversion is to be performed.

6. Security Considerations

For the most part, security considerations are the same as those that apply for capability identification in general [1,2,9].

A possible added consideration is that use of a specific feature set identifier may reveal more information about a system than is necessary for a transaction at hand.

7. Acknowledgements

Ideas here have been improved by early discussions with Martin Duerst, Al Gilman and Ted Hardie. Useful suggestions for improvement were provided by Maurizio Codogno.

8. References

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

[2] Mutz, A. and T. Hardie, "Media Feature Tag Registration Procedure", RFC 2506, March 1999.

[3] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[4] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part 1: Format of Internet message bodies", RFC 2045, November 1996.

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

[6] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[7] "Applied Cryptography"
Bruce Schneier
John Wiley and Sons, 1996 (second edition)
ISBN 0-471-12845-7 (cloth)
ISBN 0-471-11709-9 (paper)

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

[9] "Numerical Recipes"
William H Press, Brian P Flannery, Saul A Teukolski and William T Vetterling
Cambridge University Press (1986)
ISBN 0 521 30811 9
(The Gamma function approximation is presented in chapter 6 on "Special Functions". There have been several later editions of this book published, so the chapter reference may change.)

[10] Daniel, R. and M. Mealling, "Resolution of Uniform Resource Identifiers using the Domain Name System", RFC 2168, June 1997.

[11] Java source code of feature set matching algorithm, with feature set hash computation option. Linked from <http://www.imc.org/ietf-medfree/>

9. Authors' Addresses

Graham Klyne
Content Technologies Ltd.
1220 Parkview,
Arlington Business Park
Theale
Reading, RG7 4SA
United Kingdom

Phone: +44 118 930 1300
Fax: +44 118 930 1301
EMail: GK@ACM.ORG

Larry Masinter
AT&T Labs
75 Willow Road
Menlo Park, CA 94025

Phone: +1-650-463-7059
EMail: LMM@acm.org
http://larry.masinter.net

10. Appendix A: The birthday paradox

NOTE: this entire section is commentary, and does not affect the feature set reference specification in any way.

The use of a hash value to represent an arbitrary feature set is based on a presumption that no two distinct feature sets will yield the same hash value.

There is a small but distinct possibility that two different feature sets will indeed yield the same hash value.

We assume that the 128-bit hash function distributes hash values for feature sets, even those with very small differences, randomly and evenly through the range of 2^128 (approximately 3*10^38) possible values. This is a fundamental property of a good digest algorithm like MD5. Thus, the chance that any two distinct feature set expressions yield the same hash is less than 1 in 10^38. This is negligible when compared with, say, the probability that a receiving system will fail having received data conforming to a negotiated feature set.

But when the number of distinct feature sets in circulation increases, the probability of repeating a hash value increases surprisingly. This is illustrated by the "birthday paradox": given a random collection of just 23 people, there is a greater than even chance that there exists some pair with the same birthday. This topic is discussed further in sections 7.4 and 7.5 of Bruce Schneier's "Applied Cryptography" [7].

The table below shows the "birthday paradox" probabilities that at least one pair of feature sets has the same hash value for different numbers of feature sets in use.

Number of feature sets in use	Probability of two sets with the same hash value
1	0
2	3E-39
10	1E-37
1E3	1E-33
1E6	1E-27
1E9	1E-21
1E12	1E-15
1E15	1E-9
1E18	1E-3

> The above probability computations are approximate, being performed using logarithms of a Gamma function approximation by Lanczos [9]. The probability formula is 'P=1-(m!/((m-n)! m^n))', where 'm' is the total number of possible hash values (2^128) and 'n' is the number of feature sets in use.

If original feature set expressions are generated manually, or only in response to some manually constrained process, the total number of feature sets in circulation is likely to remain very small in relation to the total number of possible hash values.

The outcome of all this is: assuming that the feature sets are manually generated, even taking account of the birthday paradox effect, the probability of incorrectly identifying a feature set using a hash value is still negligibly small when compared with other possible failure modes.

11. Full Copyright Statement

Acknowledgement

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