Network Working Group
Request for Comments: 5147
Updates: 2046
Category: Standards Track

E. Wilde
UC Berkeley
M. Duerst
Aoyama Gakuin University

April 2008

# URI Fragment Identifiers for the text/plain Media Type

## Status of This Memo



## Abstract

This memo defines URI fragment identifiers for text/plain MIME entities. These fragment identifiers make it possible to refer to parts of a text/plain MIME entity, either identified by character position or range, or by line position or range. Fragment identifiers may also contain information for integrity checks to make them more robust.

Table of Contents

## 1. Introduction

This memo updates the text/plain media type defined in RFC 2046 [3] by defining URI fragment identifiers for text/plain MIME entities. This makes it possible to refer to parts of a text/plain MIME entity. Such parts can be identified by either character position or range, or by line position or range. Integrity checking information can be added to a fragment identifier to make it more robust, enabling applications to detect changes of the entity.

This section gives an introduction to the general concepts of text/plain MIME entities and URI fragment identifiers, and it discusses the need for fragment identifiers for text/plain and deployment issues. Section 2 discusses the principles and methods on which this memo is based. Section 3 defines the syntax, and Section 4 discusses processing of text/plain fragment identifiers. Section 5 shows some examples.

### 1.1. What Is text/plain?

Internet Media Types (often referred to as "MIME types"), as defined in RFC 2045 [2] and RFC 2046 [3], are used to identify different types and sub-types of media. RFC 2046 [3] and RFC 3676 [6] specify the text/plain media type, which is used for simple, unformatted text. Quoting from RFC 2046 [3]: "Plain text does not provide for or allow formatting commands, font attribute specifications, processing instructions, interpretation directives, or content markup. Plain text is seen simply as a linear sequence of characters, possibly interrupted by line breaks or page breaks".

The text/plain media type does not restrict the character encoding; any character encoding may be used. In the absence of an explicit character encoding declaration, US-ASCII [13] is assumed as the default character encoding. This variability of the character encoding makes it impossible to count characters in a text/plain MIME entity without taking the character encoding into account, because there are many character encodings using more than one octet per character.

The biggest advantage of text/plain MIME entities is their ease of use and their portability among different platforms. As long as they use popular character encodings (such as US-ASCII or UTF-8 [12]), they can be displayed and processed on virtually every computer system. The only remaining interoperability issue is the representation of line endings, which is discussed in Section 4.1.

## 1.2. What Is a URI Fragment Identifier?

URIs are the identification mechanism for resources on the Web. The URI syntax specified in RFC 3986 [7] optionally includes a so-called "fragment identifier", separated by a number sign ('#'). The fragment identifier consists of additional reference information to be interpreted by the user agent after the retrieval action has been successfully completed. The semantics of a fragment identifier are a property of the data resulting from a retrieval action, regardless of the type of URI used in the reference. Therefore, the format and interpretation of fragment identifiers is dependent on the media type of the retrieval result.

The most popular fragment identifier is defined for text/html (defined in RFC 2854 [10]) and makes it possible to refer to a specific element (identified by the value of a 'name' or 'id' attribute) of an HTML document. This makes it possible to reference a specific part of a Web page, rather than a Web page as a whole.

## 1.3. Why text/plain Fragment Identifiers?

Referring to specific parts of a resource can be very useful because it enables users and applications to create more specific references. Users can create references to the part they really are interested in or want to talk about, rather than always pointing to a complete resource. Even though it is suggested that fragment identification methods are specified in a media type's MIME registration (see [15]), many media types do not have fragment identification methods associated with them.

Fragment identifiers are only useful if supported by the client, because they are only interpreted by the client. Therefore, a new fragment identification method will require some time to be adopted by clients, and older clients will not support it. However, because the URI still works even if the fragment identifier is not supported (the resource is retrieved, but the fragment identifier is not interpreted), rapid adoption is not highly critical to ensure the success of a new fragment identification method.

Fragment identifiers for text/plain, as defined in this memo, make it possible to refer to specific parts of a text/plain MIME entity, using concepts of positions and ranges, which may be applied to characters and lines. Thus, text/plain fragment identifiers enable users to exchange information more specifically, thereby reducing the time and effort that is necessary to manually search for the relevant part of a text/plain MIME entity.

The text/plain format does not support the embedding of links, so in most environments, text/plain resources can only serve as targets for links, and not as sources. However, when combining the text/plain fragment identifiers specified in this memo with out-of-line linking mechanisms such as XLink [14], it becomes possible to "bind" link resources to text/plain resources and thereby "embed" links into text/plain resources. Thus, the text/plain fragment identifiers specified in this memo open a path for text/plain files to become bidirectionally navigable resources in hypermedia systems such as the Web.

### 1.4. Incremental Deployment

As long as text/plain fragment identifiers are not supported universally, it is important to consider the implications of incremental deployment. Clients (for example, Web browsers) not supporting the text/plain fragment identifier described in this memo will work with URI references to text/plain MIME entities, but they will fail to locate the sub-resource identified by the fragment identifier. This is a reasonable fallback behavior, and in general, users should take into account the possibility that a program interpreting a given URI will fail to interpret the fragment identifier part. Since fragment identifier evaluation is local to the client (and happens after retrieving the MIME entity), there is no reliable way for a server to determine whether a requesting client is using a URI containing a fragment identifier.

### 1.5. Notation Used in This Memo

The capitalized key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [4].

## 2. Fragment Identification Methods

The identification of fragments of text/plain MIME entities can be based on different foundations. Since it is not possible to insert explicit, invisible identifiers into a text/plain MIME entity (for example, as used in HTML documents, implemented through dedicated attributes), fragment identification has to rely on certain inherent properties of the MIME entity. This memo specifies fragment identification using four different methods, which are character positions and ranges, and line positions and ranges, augmented by an integrity check mechanism for improving the robustness of fragment identifiers.

When interpreting character or line numbers, implementations MUST take the character encoding of the MIME entity into account, because character count and octet count may differ for the character encoding being used. For example, a MIME entity using the UTF-16 encoding (as specified in RFC 2781 [11]) uses two octets per character in most cases, and sometimes four octets per character. It can also have a leading BOM (Byte-Order Mark), which does not count as a character and thus also affects the mapping from a simple octet count to a character count.

## 2.1. Fragment Identification Principles

Fragment identification can be done by combining two orthogonal principles, which are positions and ranges, and characters and lines. This section describes the principles themselves, while Section 2.2 describes the combination of the principles.

### 2.1.1. Positions and Ranges

A position does not identify an actual fragment of the MIME entity, but a position inside the MIME entity, which can be regarded as a fragment of length zero. The use case for positions is to provide pointers for applications that may use them to implement functionalities such as "insert some text here", which needs a position rather than a fragment. Positions are counted from zero; position zero being before the first character or line of a text/plain MIME entity. Thus, a text/plain MIME entity having one character has two positions, one before the first character (position zero), and one after the first character (position 1).

Since positions are fragments of length zero, applications SHOULD use other methods than highlighting to indicate positions, the most obvious way being the positioning of a cursor (if the application supports the concept of a cursor).

Ranges, on the other hand, identify fragments of a MIME entity that have a length that may be greater than zero. As a general principle for ranges, they specify both a lower and an upper bound. The start or the end of a range specification may be omitted, defaulting to the first or last position of the MIME entity, respectively. The end of a range must have a value greater than or equal to the start. A range with identical start and end is legal and identifies a range of length zero, which is equivalent to a position.

Applications that support a concept such as highlighting SHOULD use such a concept to indicate fragments of lengths greater than zero to the user.

For positions and ranges, it is implicitly assumed that if a number is greater than the actual number of elements in the MIME entity, then it is referring to the last element of the MIME entity (see Section 4 for details).

#### 2.1.2. Characters and Lines

The concept of positions and ranges can be applied to characters or lines. In both cases, positions indicate points between these entities, while ranges identify zero or more of these entities by indicating positions.

Character positions are numbered starting with zero (ignoring initial BOM marks or similar concepts that are not part of the actual textual content of a text/plain MIME entity), and counting each character separately, with the exception of line endings, which are always counted as one character (see Section 4.1 for details).

Line positions are numbered starting with zero (with line position zero always being identical with character position zero); Section 4.1 describes how line endings are identified. Fragments identified by lines include the line endings, so applications identifying line-based fragments MUST include the line endings in the fragment identification they are using (e.g., the highlighted selection). If a MIME entity does not contain any line endings, then it consists of a single (the first) line.

### 2.2. Combining the Principles

In the following sections, the principles described in the preceding section (positions/ranges and characters/lines) are combined, resulting in four use cases. The schemes mentioned below refer to the fragment identifier syntax, described in detail in Section 3.

#### 2.2.1. Character Position

To identify a character position (i.e., a fragment of length zero between two characters), the 'char' scheme followed by a single number is used. This method identifies a position between two characters (or before the first or after the last character), rather than identifying a fragment consisting of a number of characters. Character position counting starts with zero, so the character position before the first character of a text/plain MIME entity has the character position zero, and a MIME entity containing n distinct characters has n+1 distinct character positions, the last one having the character position n.

### 2.2.2. Character Range

To identify a fragment of one or more characters (a character range), the 'char' scheme followed by a range specification is used. A character range is a consecutive region of the MIME entity that extends from the starting character position of the range to the ending character position of the range.

### 2.2.3. Line Position

To identify a line position (i.e., a fragment of length zero between two lines), the 'line' scheme followed by a single number is used. This method identifies a position between two lines (or before the first or after the last line), rather than identifying a fragment consisting of a number of lines. Line position counting starts with zero, so the line position before the first line of a text/plain MIME entity has the line position zero, and a MIME entity containing n distinct lines has n+1 distinct line positions, the last one having the line position n.

### 2.2.4. Line Range

To identify a fragment of one or more lines (a line range), the 'line' scheme followed by a range specification is used. A line range is a consecutive region of the MIME entity that extends from the starting line position of the range to the ending line position of the range.

## 2.3. Fragment Identifier Robustness

It is easily possible that a modification of the referenced resource will break a fragment identifier. If applications want to create more robust fragment identifiers, they may do so by adding integrity-check information to fragment identifiers. Such information is used to detect changes in the resource. Applications can then warn users about the possibility that a fragment identifier might have been broken by a modification of the resource.

Fragment identifiers are interpreted by clients, and therefore integrity-check information is defined on MIME entities rather than on the resource itself. This means that the integrity-check information is specific to a certain entity. Specifically, content encodings and/or content transfer encodings must be removed before using integrity-check information.

Integrity-check information may specify the character encoding that has been used when creating the information, and if such a specification is present, clients MUST check whether the character

encoding specified and the character encoding of the retrieved MIME entity are equal, and clients MUST NOT use the integrity check information if these values differ. However, clients MAY choose to transcode the retrieved MIME entity in the case of differing character encodings, and after doing so, apply integrity checks. Please note that this method is inherently unreliable because certain characters or character sequences may have been lost or normalized due to restrictions in one of the character encodings used.

## 3. Fragment Identification Syntax

The syntax for the text/plain fragment identifiers is straightforward. The syntax defines four schemes, 'char', 'line', and integrity check (which can either be 'length' or 'md5'). The 'char' and 'line' schemes can be used in two different variants, either the position variant (with a single number), or the range variant (with two comma-separated numbers). An integrity check can either use the 'length' or the 'md5' scheme to specify a value. 'length' in this case serves as a very weak but easy to calculate integrity check.

The following syntax definition uses ABNF as defined in RFC 5234 [9], including the rules DIGIT and HEXDIG. The mime-charset rule is defined in RFC 2978 [5].

NOTE: In the descriptions that follow, specified text values MUST be used exactly as given, using exactly the indicated lower-case letters. In this respect, the ABNF usage differs from [9].

```
text-fragment   =  text-scheme 0*( ";" integrity-check )
text-scheme     =  ( char-scheme / line-scheme )
char-scheme     =  "char=" ( position / range )
line-scheme     =  "line=" ( position / range )
integrity-check =  ( length-scheme / md5-scheme )
                     [ "," mime-charset ]
position        =  number
range           =  ( position "," [ position ] ) / ( "," position )
number          =  1*( DIGIT )
length-scheme   =  "length=" number
md5-scheme      =  "md5=" md5-value
md5-value       =  32HEXDIG
```

### 3.1. Integrity Checks

An integrity check can either specify a MIME entity's length, or its MD5 fingerprint. In both cases, it can optionally specify the character encoding that has been used when calculating the integrity

check, so that clients interpreting the fragment identifier may check whether they are using the same character encoding for their calculations. For lengths, the character encoding can be necessary because it can influence the character count. As an example, Unicode includes precomposed characters for writing Vietnamese, but in the windows-1258 encoding, also used for writing Vietnamese, some characters have to be encoded with separate diacritics, which means that two characters will be counted. Applying Unicode terminology, this means that the length of a text/plain MIME entity is computed based on its "code points". For MD5 fingerprints, the character encoding is necessary because the MD5 algorithm works on the binary representation of the text/plain resource.

To allow future changes to this specification to address developments in cryptography, implementations MUST ignore new types of integrity checks, with names other than 'length' and 'md5'. If several integrity checks are present, an application can use whatever integrity checks it understands, and among these, those integrity checks that provide an appropriate trade-off between performance and the need for integrity checking. Please see Section 4.3 for further details.

The length of a text/plain MIME entity is calculated by using the principles defined in Section 2.1.2. The MD5 fingerprint of a text/plain MIME entity is calculated by using the algorithm presented in [1], encoding the result in 32 hexadecimal digits (using uppercase or lowercase letters) as a representation of the 128 bits that are the result of the MD5 algorithm. Calculation of integrity checks is done after stripping any potential content-encodings or content-transfer-encodings of the transport mechanism.

## 4. Fragment Identifier Processing

Applications implementing support for the mechanism described in this memo MUST behave as described in the following sections.

### 4.1. Handling of Line Endings in text/plain MIME Entities

In Internet messages, line endings in text/plain MIME entities are represented by CR+LF character sequences (see RFC 2046 [3] and RFC 3676 [6]). However, some protocols (such as HTTP) additionally allow other conventions for line endings. Also, some operating systems store text/plain entities locally with different line endings (in most cases, Unix uses LF, MacOS traditionally uses CR, and Windows uses CR+LF).

Independent of the number of bytes or characters used to represent a line ending, each line ending MUST be counted as one single

character. Implementations interpreting text/plain fragment identifiers MUST take into account the line ending conventions of the protocols and other contexts that they work in.

As an example, an implementation working in the context of a Web browser supporting http: URIs has to support the various line ending conventions permitted by HTTP. As another example, an implementation used on local files (e.g., with the file: URI scheme) has to support the conventions used for local storage. All implementations SHOULD support the Internet-wide CR+LF line ending convention, and MAY support additional conventions not related to the protocols or systems they work with.

Implementers should be aware of the fact that line endings in plain text entities can be represented by other characters or character sequences than CR+LF. Besides the abovementioned CR and LF, there are also NEL and CR+NEL. In general, the encoding of line endings can also depend on the character encoding of the MIME entity, and implementations have to take this into account where necessary.

## 4.2. Handling of Position Values

If any position value (as a position or as part of a range) is greater than the length of the actual MIME entity, then it identifies the last character position or line position of the MIME entity. If the first position value in a range is not present, then the range extends from the start of the MIME entity. If the second position value in a range is not present, then the range extends to the end of the MIME entity. If a range scheme's positions are not properly ordered (i.e., the first number is less than the second), then the fragment identifier MUST be ignored.

## 4.3. Handling of Integrity Checks

Clients are not required to implement the handling of integrity checks, so they MAY choose to ignore integrity check information altogether. However, if they do implement integrity checking, the following applies:

If a fragment identifier contains one or more integrity checks, and a client retrieves a MIME entity and, using some integrity check(s), detects that the entity has changed (observing the character encoding specification as described in Section 3.1, if present), then the client SHOULD NOT interpret the text/plain fragment identifier. A client MAY signal this situation to the user.

### 4.4. Syntax Errors in Fragment Identifiers

If a fragment identifier contains a syntax error (i.e., does not conform to the syntax specified in Section 3), then it MUST be ignored by clients. Clients MUST NOT make any attempt to correct or guess fragment identifiers. Syntax errors MAY be reported by clients.

## 5. Examples

The following examples show some usages for the fragment identifiers defined in this memo.

http://example.com/text.txt#char=100

This URI identifies the position after the 100th character of the text.txt MIME entity. It should be noted that it is not clear which octet(s) of the MIME entity this will be without retrieving the MIME entity and thus knowing which character encoding it is using (in case of HTTP, this information will be given in the Content-Type header of the response). If the MIME entity has fewer than 100 characters, the URI identifies the position after the MIME entity's last character.

http://example.com/text.txt#line=10,20

This URI identifies lines 11 to 20 of the text.txt MIME entity. If the MIME entity has fewer than 11 lines, it identifies the position after the last line. If the MIME entity has less than 20 but at least 11 lines, it identifies the range from line 11 to the last line of the MIME entity.

https://example.com/text.txt#line=,1

This URI identifies the first line. Please note that the URI scheme has been changed to https.

ftp://example.com/text.txt#line=10,20;length=9876,UTF-8

As in the second example, this URI identifies lines 11 to 20 of the text.txt MIME entity. The additional length integrity check specifies that the MIME entity has a length of 9876 characters when encoded in UTF-8. If the client supports the length scheme, it may test the retrieved MIME entity for its length, but only if the retrieved MIME entity uses the UTF-8 encoding or has been locally transcoded into this encoding.

Please note that the FTP protocol, as well as some other protocols underlying some other URI schemes, do not provide explicit information about the media type of the resource being retrieved. Using fragment identifiers with such URI schemes is therefore inherently unreliable. Current user agents use various heuristics to infer some media type for further processing. Processing of the fragment identifier according to this memo is only appropriate if the inferred media type is text/plain.

## 6. IANA Considerations

IANA has added a reference to this specification in the text/plain Media Type registration.

## 7. Security Considerations

The fact that software implementing fragment identifiers for plain text and software not implementing them differs in behavior, and the fact that different software may show documents or fragments to users in different ways, can lead to misunderstandings on the part of users. Such misunderstandings might be exploited in a way similar to spoofing or phishing.

In particular, care has to be taken if fragment identifiers are used together with a mechanism that allows showing only the part of a document identified by a fragment. One scenario may be the use of a fragment identifier to hide small-print legal text. Another scenario may be the inclusion of site-key-like material, which may give the user the impression of using the real site rather than a fake site; other scenarios may also be possible. Possible countermeasures may include but are not limited to displaying the included content within clearly visible boundaries and limiting inclusion to material from the same security realm or from realms that give explicit permission to be included in another realm.

Please note that the above issues all apply to the client side; fragment identifiers are not used when resolving a URI to retrieve the representation of a resource, but are only applied on the client side.

Implementers and users of fragment identifiers for plain text should also be aware of the security considerations in RFC 3986 [7] and RFC 3987 [8].

## 8. References

### 8.1. Normative References

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

[2] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[3] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

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

[5] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.

[6] Gellens, R., "The Text/Plain Format and DelSp Parameters", RFC 3676, February 2004.

[7] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[8] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRI)", RFC 3987, January 2005.

[9] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

### 8.2. Informative References

[10] Connolly, D. and L. Masinter, "The 'text/html' Media Type", RFC 2854, June 2000.

[11] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

[12] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[13] ANSI X3.4-1986, "Coded Character Set - 7-Bit American National Standard Code for Information Interchange", 1986.

[14] DeRose, S., Maler, E., and D. Orchard, "XML Linking Language (XLink) Version 1.0", World Wide Web Consortium Recommendation, June 2001, <http://www.w3.org/TR/xlink/>.

[15] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

## Appendix A. Acknowledgements

Thanks for comments and suggestions provided by Marcel Baschnagel, Stephane Bortzmeyer, Tim Bray, Iain Calder, John Cowan, Spencer Dawkins, Lisa Dusseault, Benja Fallenstein, Ted Hardie, Sam Hartman, Sandro Hawke, Jeffrey Hutzelman, Cullen Jennings, Graham Klyne, Dan Kohn, Henrik Levkowetz, Chris Newman, Mark Nottingham, Conrad Parker, and Tim Polk.

## Authors' Addresses

Erik Wilde
UC Berkeley
School of Information, 311 South Hall
Berkeley, CA 94720-4600
U.S.A.

Phone: +1-510-6432253
EMail: dret@berkeley.edu
URI: http://dret.net/netdret/

Martin Duerst (Note: Please write "Duerst" with u-umlaut wherever possible, for example as "Dürst" in XML and HTML.)
Aoyama Gakuin University
5-10-1 Fuchinobe
Sagamihara, Kanagawa 229-8558
Japan

Phone: +81 42 759 6329
Fax: +81 42 759 6495
EMail: duerst@it.aoyama.ac.jp
URI: http://www.sw.it.aoyama.ac.jp/D%C3%BCrst/

## Full Copyright Statement



## Intellectual Property