Internet-Draft
draft-mallery-urn-pdi-00.txt
Expires in six months

J. C. Mallery
M.I.T.

November 10, 1997

Persistent Document Identifiers
Filename: draft-mallery-urn-pdi-00.txt

Status of This Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

This document specifies the syntax and semantics of the Persistent Document Identifier (PDI) namespace within the URN framework defined by RFC 2141 [17]. PDIs provide a means to refer to digital objects and fragments that does not depend their storage location or the protocol used to access them. Since 1994, several large-scale applications with these requirements have used PDIs [12] [21].

PDIs are intended primarily as permanent identifiers for archival reference to long-lived documents. PDIs have a fragment syntax to allow permanent references to parts of documents (within specific formats) as well as a citation syntax to allow references to appearances of such fragments in composite documents.

PDIs are most useful for any document series that is distributed via multiple protocols, is available from multiple sources, migrates to new locations, needs fragment references, or participates in distributed assertion semantics related to collaboration or access control.

1. Namespace Syntax

1.2 Design Goals

Persistent Document Identifiers provide a means to refer to digital objects and fragments that does not depend their storage location or the protocol used to access them. PDIs offer the following capabilities:

* Multisourcing: The same resource can be stored in different locations yet retrieved by a virtue of a shared identifier.

* Multiple Protocols: Identifiers are not tied to specific transport protocols.

* Persistence: PDIs persist across relocation of a digital object to different storage sites. The longevity of a PDI is not limited by lifetime of a directory, domain name, or even, a transport protocol.

* Organizational Delegation: PDIs define a hierarchical encoding of the issuing authority that allows delegation in a manner analogous to names in the Domain Name System names but more akin to X.400.

* Chronological Delegation: PDIs incorporate a time hierarchy that allows delegation of identifiers with different time ranges to different authorities or to different resolution regimes.

* Fragment Syntax: PDIs offer an extensible syntax for referring to part of a resource. This evolutionary approach allows different schemes according to media type as well as multiple schemes per media type. Longevity of reference is sought by defining fragment schemes that are independent of machine representation. Referential consistency is guaranteed by monotonic commitment of versioned PDIs to immutable resource representations.

* Citation Syntax: PDIs include a syntax for referring to appearances of document fragments as quoted in other composite documents. This makes fragment quotations first-class objects, about which assertions can be made.

* User Friendly: PDIs carry a relatively simple syntax with some mnemonics so that, if need be, people can type them to access a resource.

A guiding design principle for PDIs is to minimize the document semantics carried within the identifier. Most semantics is better encoded by assertions about PDIs. Not only is overloading of the identifier avoided, but assertions can also be modified without recourse to changing the identifier.

1. Namespace Syntax

Consistent with the URN syntax specification in RFC 2141 [17], each namespace must specify syntax related information that is specific to that namespace. This section provides these specifications for the PDI namespace. The PDI grammar below uses the ABNF [6]. A URN using the Persistent Document Identifier namespace has the form:

```
<URN> = "urn:" pdi              ; Encoding in URN syntax
```

1.1. Namespace Identifier (NID)

The Namespace Identifier for this namespace is "pdi", which is case insensitive.

```
<PDI> = "pdi" ":" nss           ; Persistent Document Identifier
```

1.2. Namespace Specific String (NSS)

The Namespace Specific String for this namespace is:

```
<NSS> = resource-identifier [(citation-specifier / fragment-specifier)]
```

1.2.1 Resource Identifier

```
<RESOURCE-IDENTIFIER> = "//" document-series "/" iso-date "/" specifier

<DOCUMENT-SERIES> = component *["." component] "." iso-country

<COMPONENT> = alpha-hyphen-digits

<ISO-COUNTRY> = 2*alpha                 ; See ISO Standard 3166 [10]

<ISO-DATE> = year "/" month "/" day

<YEAR> = 4*digit / wildcard

<MONTH> = 2*digit / wildcard

<DAY> = 2*digit / wildcard

<SPECIFIER> = unique-id ["." format ["." version]] ;versions require
formats

<UNIQUE-ID>  = daily-serial-number / encapulated-unique-id / digits /
                  wildcard

<DAILY-SERIAL-NUMBER> = digits

<ENCAPULATED-UNIQUE-ID> = unique-id-chars

<UNIQUE-ID-CHARS> = alpha / digit / other / "%" hex hex
```

```
        <FORMAT> =  media-type-token / wildcard

        <MEDIA-TYPE-TOKEN> = "text" / "html" / extension-token

        <EXTENSION-TOKEN> = alpha-hyphen

        <VERSION> = digits / wildcard

        <WILDCARD> = "*"
```

1.2.2 Citation Specifier

```
        <CITATION-SPECIFIER> = "@" origin-position "=" pdi

        <ORIGIN-POSITION> = position
```

1.2.3 Fragment Specifier

```
        <FRAGMENT-SPECIFIER> = "#" [fragment-scheme "="] position [*("," position)]

        <FRAGMENT-SCHEME> = "char" / "elt" / "name" / "rect" / "msec" / "sec" /
                            "crop" / "byte" / ext-fragment-scheme

        <EXT-FRAGMENT-SCHEME> = alpha-hyphen

        <POSITION> = char-position / element-position / element-name /
                     2-dim-coordinate / frame-number / time / byte-position /
                     ext-position

        <EXT-POSITION> = position-specifier /
                         "(" position-specifier  *["," position-specifier] ")"

        <POSITION-SPECIFIER> = alphadigits
```

1.2.4 Supporting Definitions

```
        <ALPHA> = %x41-5A / %x61-7A                    ; A-Z / a-z

        <ALPHA-DIGITS> = alphas / digits

        <ALPHA-HYPHEN-DIGITS> = alpha-hyphen / digits

        <ALPHA-HYPHEN> = alpha / "-"

        <ALPHA-HYPHENS> = *alpha-hyphen

        <ALPHAS> = *ALPHA

        <DIGIT> = %x30-39                              ; 0-9
```

```
<DIGITS> = *DIGIT

<URN-CHARS> = trans / "%" hex hex                ;RFC 2141

<TRANS> =  alpha / digit / other / reserved

<HEX> = digit / "A" / "B" / "C" / "D" / "E" / "F" /
                "a" / "b" / "c" / "d" / "e" / "f"

<OTHER> = "(" / ")" / "-" / ":" / ";" / "$" / "_" / "!" / "'"

<RESERVED> = "%" / "." / "," / "/" / "#" / "*" / "@" /
             "=" / "?" / "+"
```

1.2.5 Reserved Characters

<RESERVED> are used as special characters in the PDI grammar. They MUST be encoded according to the character escaping method described in RFC 2141 [17].

2 Discussion

2.1 Minting PDIs

PDIs are issued by the authority named in <DOCUMENT-SERIES>. <DOCUMENT-SERIES> is intended to look like a domain name for easy parsing but there is no requirement to serve the name via the Domain Name System (DNS) nor to assure that the name is not assigned for other purposes by DNS.

The encoded date in <ISO-DATE> is the date when the identifier is minted. This date is based on Greenwich meantime. The encoded date bears no relationship to dates associated with the resource that the PDI denotes, even if there may be proximity between the time when the resource issues and the time when the PDI is minted.

The PDI namespace is monotonic; PDIs cannot be retracted. If a new version of the same document issues, it MUST increment the version number for the previously issued PDI. This requirement assures that any machine representation (byte sequence) associated with formats of a versioned PDI never changes.

Byte equivalence for all resource formats denoted by a specific PDI version ensures that digital signatures associated with a PDI check for any uncorrupted resource. More significantly, byte equivalence enables reliable, efficient fragment references for many media types. It eliminates the potentially difficult problem of rolling fragment references forward as a target resource is modified.

2.2 Issuing Authority

The issuing authority controls the name in a document series. These names are hierarchical so that administration can be delegated within authority domains. Unlike domain names, the right most component of a <DOCUMENT-SERIES> MUST be a two digit ISO 3166 country code [10], indicating the country in which the issuing organization resides. In most cases, a <DOCUMENT-SERIES> SHOULD add a term to the issuing authority in order to differentiate the series from other document sets that the authority might issue. By specializing the document series below the issuing authority, identifiers reflect the chain of delegation. Additionally, it becomes easier to obsolesce an entire document series, if that becomes necessary.

For wide use of PDIs, an issuing authority will need to issue toplevel authority names to organizations wishing to mint PDIs in their own document series. Once a toplevel document series name has been obtained, an organization may issue PDIs itself or delegate subseries.

A subseries is delegated by adding a name component to the left of <DOCUMENT-SERIES>. The accretion of components on a document series MAY utilize existing organizational names or acronyms whenever feasible in order to preserve mnemonics in the document series name. Additionally, dropping components from the left SHOULD lead to ever more general issuing authorities in terms of organizational scope.

Delegation SHOULD follow de jure organizational structure. Issuing authority SHOULD NEVER be delegated outside the organization unless the external agent is acting directly on behalf of the document series owner. When organizational boundaries are crossed, a new document series toplevel SHOULD be acquired. Within an organization, issuing authority SHOULD be delegated to the level where responsibility for content resides. This facilitates contact with document originators. More importantly, it reduces administrative scope, and thus, encourages more uniform document management policies for a particular document series.

2.3 Hierarchical Date

<ISO-DATE> of a PDI MUST be assigned when the identifier is minted. The calendar date MUST correspond to Greenwich Mean time.

Inclusion of the ISO date conveys the time when the identifier was minted. Beyond making it easier to guarantee identifier uniqueness, hierarchicalization by date enables reference to ranges of identifiers issued within specific time intervals.

Use of ISO dates also ensures that lexical sorts of identifiers produce a chronological ordering of PDIs, making various listings (e.g., directory lists) automatically appear in a meaningful

order.

Moreover, different administrative policies MAY be applied to any particular time interval. For example, when responsibility for resolving PDIs shifts to a different administrative authority, intervals covered by the new policy are readily specifiable and conveyed. For example, different intervals may be delegated to different URN resolvers and these delegations recorded with relevant URN discovery systems.

Operations may be applied to identifiers within an interval. For example, a browser can provide a directory list of all the documents in a year, a month, or on a day.

More generally, assertions can be made about identifiers within an interval, such as where to find a resolver.

2.4 Daily Unique ID

An application may use a mnemonic name or a serial number as the <UNIQUE-ID>. The only requirement is that <UNIQUE-ID> MUST be a unique sequence of <UNIQUE-ID-CHARS> for <ISO-DATE> and <DOCUMENT-SERIES>.

If the unique ID is a <DAILY-SERIAL-NUMBER>, serial numbers SHOULD start from 1 and SHOULD be incremented by 1 as each new PDI is minted. When the calendar day is incremented at midnight GMT, the unique ID of the day SHOULD be reset to start at 1 on the new day. This prevents daily unique IDs from growing very large as it enforces date semantics on the identifier.

2.4.1 Encapsulation of Foreign Identifiers

The specification of this field has been left open so that foreign document identifiers MAY be incorporated within a PDI as the daily unique ID. For our purposes, a foreign identifier is any identifier used by other naming or reference regimes. Examples of foreign identifiers include, serial numbers, invoice numbers, URIs, URLs or other application-specific identifiers.

When encapsulating a foreign identifier, <FORMAT> is required and MUST use a <MEDIA-TYPE-TOKEN> that identifies the media type of the resource and format of the encapsulated identifier. The media type token is required in order to allow unambiguous interpretation by applications aware of the identifier semantics. All other applications, MUST treat the unique id as opaque.

2.5 Format

Format should use standard, controlled terms that indicate the media type [3] of the resource to which the identifier refers or, in the case of encapsulated identifiers, indicate the type of the

encapsulated identifier. <FORMAT> is case insensitive.

The standards for MIME content types [10] do not as yet provide a single controlled term per media type that can be used as a file extension or here as a PDI format. Below we provide a rule for constructing the <MEDIA-TYPE-TOKEN>. These tokens are created from the registered media types [10] by using the <MINOR-TYPE> if it is unique, or otherwise, concatenating the <MAJOR-TYPE> and <MINOR-TYPE>. These tokens are case insensitive and MUST encode any reserved characters (<RESERVED>) for PDIs.

```
<CONTENT-TYPE> = major-type "/" minor-type
                    [* (";" parameter ["=" value])]

<MEDIA-TYPE-TOKEN> = minor-type / (major-type "+" minor-type)

<MAJOR-TYPE> = alpha-hyphen-digits

<MINOR-TYPE> = alpha-hyphen-digits
```

There are two media types for which <MEDIA-TYPE-TOKEN> is not <MINOR-TYPE>:

Token	Content Type	
text	text/plain	
header	message/header	;RFC 822 message headers

<FORMAT> is always required when:

* A PDI is minted and assigned to a specific resource.
* A foreign document ID is encapsulated in <UNIQUE-ID>.
* References to resource fragments are made.
* A client requests a resource in a specific format.

The format indicates how to interpret encapsulated identifiers and MUST be supplied whenever foreign document identifiers are encapsulated. For example, if an HTTP URL was encapsulated, the PDI might look like:

pdi://oma.eop.gov.us/1994/10/20/http%3a%2f%2fwww%2ewhitehouse%2egov%2f.html.

This PDI encapsulates the URL http://www.whitehouse.gov/ and denotes its content on October 20, 1994, when the site was unveiled.

When a PDI contains fragment syntax, a format MUST be provided in order to convey the media type of the resource to which the fragment reference applies.

A server may store any subset of formats for a resource. It may compute unstored formats on demand. A client can specify the desired

format by using a PDI with the appropriate format field.

If format is omitted, the identifier refers to the generic resource denoted by the PDI. Assertions about the generic resource apply to all the instantiations in the various media types indicated by the universe of format in which the resource is available.

2.6 Version

The PDI <VERSION> is an optional component indicating a specific version of a resource. <VERSION> is a positive integer greater than 0. When <VERSION> is omitted, it defaults to version 1.

Version numbers refer to the generic resource and not the specific format, but a resource cannot have a version without having at least one format. When a resource is changed in any format, version numbers for all formats MUST be updated. In general, when a resource changes significantly, applications SHOULD generate new PDIs. When changes are small or incremental, applications SHOULD increment the version. Any change in the byte count of a resource for a specific <FORMAT> is a change and the version SHOULD be incremented. Addition of a new <FORMAT> with the same semantics as an existing <FORMAT> for the PDI is not a change and does not require the version to be incremented.

Consequently, if an HTML document issues under

pdi://oma.eop.gov/1997/09/01.html.1

,and later, the HTML is converted to text, the PDI for the text version is

pdi://oma.eop.gov/1997/09/01.text.1

However, if a spelling mistake is corrected later, whether or not it changes the byte count in any format, the version number is incremented.

pdi://oma.eop.gov/1997/09/01.text.2

An editing application MAY write internal versions of a document in progress and only commit to the versioned PDI at a point when the editing completed and the document is ready for release.

Version numbers MUST be included when:

* PDIs are minted and associated with specific resources.
* PDIs contain a fragment references.
* PDIs contain a fragment citation.

Inclusion of a <VERSION> in a fragment references ensures that the fragment reference is resolved against a consistent machine

representation of the resource.

3 Fragment Syntax

3.1 Motivation

The PDI namespace provides an extensible syntax for referring to parts of resources. Fragment syntax must be extensible because:

* There are too many existing media types.

* Some media types require highly technical fragment syntax, (e.g., multidimensional points, multiresolution channels).

* New media types are coming into existence all the time.

The approach adopted here is to allow additional RFCs to extend fragment syntax by adding fragment specifiers as they are needed.

The availability of a syntax for referring to resource fragments raises the problem of referring to citations of fragments by composite resources. The PDI namespace provides a fragment citation syntax to address this issue.

3.2 Philosophy

3.2.1 Media Representations

A fragment syntax SHOULD differentiate the media representation from the machine representation. If fragment schemes for a particular media type use a media representation, they can be retargeted at new or different machine representations. Otherwise, fragment schemes may become unresolvable in the future when machine representations change. Consequently, although a byte fragment specifier is provided below, it SHOULD be used only for short-term purposes when alternatives are unavailable.

3.2.2 Immediate Fragments

URNs require a fragment syntax because the alternative of interning every fragment PDI in a URN namespace does not scale. It requires the resolver to store potentially all possible permutations of the fragment specifier for every resource. Immediate fragments require the fragment syntax to be part of the identifier. With immediate fragments, resolvers need only store those fragment PDIs for which there are assertions beyond the binding to the resource subset. Additionally, immediate fragments enhance privacy by not storing all references to resource subsets. They also conserve storage and reduce computation on resolvers.

3.2.3 Fragment Conjunctions

The fragment syntax does not support conjunctions of fragments because this introduces a source of ambiguity when assertions are made about PDIs. Conjunctive fragments SHOULD be handled by creating a new PDI and asserting that it is the conjunction of some fragments. In this way, the set is explicitly represented and ambiguous references are excluded from the syntax.

3.2.4 Decoupling from Reference Mechanics

Fragment reference could be accomplished by providing a program that given a resource return the specified part. This is not the approach advocated here. The fragment scheme MUST be a minimal set of parameters required for a program to extract the relevant part. Additionally, these parameters SHOULD be specified in the order of importance for extracting the referent. This increases the probability of finding a referent if an identifier is accidently truncated. In general, new fragment specifiers SHOULD minimize the syntax the of invariants and parameters they require.

3.3 Fragment Scheme

The <FRAGMENT-SCHEME> indicates the position syntax used in <POSITION>. A default position scheme should be defined for each Content Type token used in PDIs. For example, text/plain uses character positions as the default. The <FRAGMENT-SCHEME> MAY be omitted when it is the default position scheme for the content type indicated by <FORMAT>. In all other circumstances, <FRAGMENT-SCHEME> MUST be supplied in order to ensure unambiguous interpretation of position specifiers. Position schemes are case insensitive.

3.4 Fragment Specifiers

The following position reference schemes have been defined:

3.4.1 Text Fragment Specifier

Text fragments are defined for the MIME Content Type text/*. Each text fragment is an interval bounded by two character positions in the resource. The fragment is the set of characters from <START> upto but excluding <END>. The first character position starts with 0. Character positions are relative to the canonical,CRLF encoded text for the resource. Therefore, all text/* resources MUST be CRLF encoded to ensure correct fragment references. The PDI <FORMAT> for text/plain is "text" and <CHAR-FRAGMENT-SPECIFIER> is the default position specifier for the media type.

```
<CHAR-FRAGMENT-SPECIFIER> = "#" ["char" "="] start-char
                                "," end-char

<START-CHAR> = digits
```

```
<END-CHAR> = digits
```

Although wide-spread encodings for many alphabets use a single 8 bit byte (e.g., ISO-8859 [15]), other encodings (e.g., unicode) employ multi-byte encodings. Consequently, a server MUST be aware of the character set used to encode a text resource. For 8 bit character sets, char fragment resolution reduces to byte position. However, multi-byte character sets require the server to perform appropriate translation from the stored data representation.

The following PDI refers to the text starting at character 37 and continuing upto but excluding character 51.

```
pdi://oma.eop.gov.us/1997/09/01/1.text.1#char=37,51
```

Since the default fragment specifier for text is <CHAR-FRAGMENT-SCHEME>, the following PDI is equivalent:

```
pdi://oma.eop.gov.us/1997/09/01/1.text.1#37,51
```

When a text/plain content type uses a multi-byte character set, <FORMAT> MUST be the character set token as defined by the IANA Character Set Registry [18].

3.4.2 HTML Fragment Specifier

Fragments may be specified for the MIME Content Type text/html using character fragment specifiers. The PDI <FORMAT> for text/html is "html". The default position specifier for text/html is "char" because it simplifies serving fragments.

Although character references are simple and effective for HTML document fragments, it is often more convenient to use HTML elements to delimit an interval within a document. Specific HTML elements can be identified using the name parameter value or the position of the tag in the document. In either case, the fragment consists of all text and HTML tags from <START-ELEMENT> to and including <END-ELEMENT>. References to HTML containers is facilitated by use of a closed interval, but it can be awkward for tags that are not explicitly closed, especially if they are implicitly closed (e.g., <p>). Tag positions are counted from the start of the resource, with the first being assigned 0. An <ELEMENT-NAME> refers to the first element whose name parameter value is equal to <ELEMENT-NAME>, which must be encoded according to URN syntax [17], but decoded for case-sensitive equality testing.

```
<HTML-FRAGMENT-SPECIFIER> = "#" start-element "," end-element

<HTML-FRAGMENT-SCHEME> = char-fragment-scheme /
                         element-fragment-scheme /
                         named-fragment-scheme
```

```
<ELEMENT-FRAGMENT-SCHEME> = "elt"

<START-ELEMENT> = element-position / element-name

<END-ELEMENT> = element-position / element-name

<ELEMENT-POSITION> = digits

<NAMED-FRAGMENT-SCHEME> = "name"

<ELEMENT-NAME> = urn-chars
```

Char, elt, and name position references MUST use the same position scheme for <START-ELEMENT> and <END-ELEMENT> an HTML fragment reference.

HTML fragments may depend on surrounding context that is not part of the fragment. HTML rendition without this containing context may produce different effects or incorrect HTML. Responsibility for assuring legal and felicitous HTML must reside with the user or application creating the fragment reference because document authors cannot be expected to anticipate all possible citations. Therefore, the user or application creating the fragment citation MUST NOT create illegal HTML fragments.

When fragments require context, the user or application MAY create an intermediate document that uses fragment references to extract both the relevant context and the target fragment. This intermediate document SHOULD be legal HTML capable of standing alone.

3.4.2 SGML & XML Fragment Specifier

The element and char fragment schemes can be applied to the more general Standard Generalized Markup Language (SGML) [14] and Extensible Markup Language XML [4] mark up languages, of which it is a subset. The BNF below give the fragment specification for SGML and any subsets, such as XML.

```
<SGML-FRAGMENT-SPECIFIER> = "#" sgml-start-element ","
                                sgml-end-element

<SGML-FRAGMENT-SCHEME> = element-fragment-scheme /
                         char-fragment-scheme

<SGML-START-ELEMENT> = element-position

<SGML-END-ELEMENT> = element-position
```

The default fragment specifier for SGML and SGML subsets is "char". The following content tokens are defined:

```
text/sgml               sgml
text/xml                xml
```

The context caveats for HTML fragments should be extended pari pasu to SGML and XML fragments.

3.4.5 Image Fragment Specifier

Image media types use a variety of encoding schemes and some include multiple frames. Fragment reference for image/* uses a two dimensional cartesian coordinate system with the origin (0, 0) being in the upper left hand corner. The scale of the coordinate system is the pixel level scale of the containing image. References to subrectangles are made by specifying for the image fragment the <START-COORDINATE> as the upper left most point and <END-COORDINATE> as the lower right most point. These x and y coordinates are in coordinate system of the containing image. When multiple frames are present in an image, the reference frame is specified by providing <FRAME>, which is 0 based and defaults to 0 when omitted. <RECTANGLE-FRAGMENT-SPECIFIER> is the default fragment specifier for the media types image/*.

```
<RECTANGLE-FRAGMENT-SPECIFIER> = # ["rect" "="] start-coordinate
                                   "," end-coordinate ["," frame]

<FRAME> = digits

<START-COORDINATE> = 2-dim-coordinate

<END-COORDINATE> = 2-dim-coordinate

<2-DIM-COORDINATE> = "(" x-coordinate "," y-coordinate ")"

<X-COORDINATE> = digits

<Y-COORDINATE> = digits
```

The example below refers to an image fragment whose origin is x=5, y=10 and extends to x=25, y=30. This yields the maximal rectangle including the coordinates (5,10), (24,10), (24,29), (5,29). Note that the zero-based coordinate system does not include the point denoted by the <END-COORDINATE>.

```
pdi://images.satellite.nasa.gov.us/1997/09/30/1234.gif#(5,10),(25,30)
```

Since frame is unspecified, it defaults to zero and this PDI is equivalent to

```
pdi://images.satellite.nasa.gov.us/1997/09/30/1234.gif#(5,10),(25,30),0
```

The next PDI refers to the third frame in an animated GIF. As it simplifies array references, the zero-based index shifts references

to the left by 1.

pdi://images.satellite.nasa.gov.us/1997/09/30/1234.gif#(5,10),(25,30),2

3.4.4 Audio Fragment Specifier

Audio media types use various encoding schemes (including variable quality) that make byte ranges problematic for fragment references. Start and end times provide a coordinate scheme that can be resolved for any audio media type. A fragment reference includes data from and including <START-TIME> upto and excluding <END-TIME>. The position scheme for the temporal reference gives the time units. Two time position schemes are defined. "msec" is millesconds and "sec" is seconds. Temporal position schemes MUST NOT be intermixed. The default time position scheme for audio/* is "sec".
<TIME-FRAGMENT-SPECIFIER> is the default fragment specifier for audio/*.

```
<TIME-FRAGMENT-SPECIFIER> = # time-position-scheme "="
                             start-time "," end-time

<TIME-POSITION-SCHEME> = "msec" / "sec" /
                             ext-time-position-scheme

<START-TIME> = time

<END-TIME> = time

<TIME> = digits

<EXT-TIME-POSITION-SCHEME> = alpha-hyphen-digits
```

The example below denotes the audio clip extending from second 23 upto but not including second 57.

pdi://audio.npr.org.us/1997/09/30/1234.au#sec=23,57

3.4.5 Video Fragment Specifier

Video media types combine difficulties similar to those presented by audio and image media types. A simple syntax for video should allow fragment references to video by start and end times. Because some applications may wish to crop the image, an optional x-y coordinate framework is also supported.

The video fragment specifier uses a required time component and an optional pair of coordinates to denote cropping. A video fragment reference includes data from and including <START-TIME> upto and excluding <END-TIME> in time units given by <TIME-POSITION-SCHEME>.

When cropping is desired, a fragment may include the optional

<START-COORDINATE> and <END-COORDINATE>.

<VIDEO-FRAGMENT-SPECIFIER> = # "crop" "=" time-position-scheme
"," start-time "," end-time
["," start-coordinate "," end-
coordinate]

The following example refers to seconds 23 upto 51 of the video clip 1234.mpeg.

pdi://video.cnn.co.us/1997/09/30/1234.mpeg.1#sec,23,51

The next PDI uses the crop fragment scheme and is equivalent to the preceding one because no cropping is specified.

pdi://video.cnn.co.us/1997/09/30/1234.mpeg.1#crop=sec,23,51

However, the crop scheme is easily able to specify a cropping, such as the one below.

pdi://video.cnn.co.us/1997/09/30/1234.mpeg.1#crop=sec,23,51,(10,10),
(20,20)

Here, only the rectangle from origin (10,10) to (20,20), the right lowest point, is included in the fragment.

The video fragment scheme presupposes a pixel-based imaging model. For other models, such as line-oriented analog video, specialized fragment schemes may be appropriate when cropping is desired. Note that the time fragment scheme works for analog models. For this reason, the time position scheme <TIME-FRAGMENT-SPECIFIER> is the default fragment specifier for video/*.

3.4.6 Octet Fragment Specifier

In general, long-lived fragment specifiers seek to avoid schemes that depend on the underlying data representation because of low generality and high probably of future failure when the data representation is obsolesced by future developments. The byte fragment scheme is provided as short-term solution that SHOULD be superseded by defining a new fragment specifier. In any event, byte fragments provide a fallback scheme for use until a new extension can be introduced.

The content token for <BYTE-FRAGMENT-SPECIFIER> is "byte". The fragment includes <START-BYTE> and all intervening bytes upto <END-BYTE> which is excluded from the fragment reference.

<BYTE-FRAGMENT-SPECIFIER> = "#" "byte" "=" start-byte "," end-byte

<START-BYTE> = digits

<END-BYTE> = digits

The fragment PDI below refers to the byte sequence starting with byte 23 and continuing upto but excluding byte 57.

pdi://documentation.adobe.co.us/1997/09/30/1234.pdf#byte=23,57

These byte indices are compatible with byte ranges used in HTTP 1.1 [9].

3.5 Fragment Citation

3.5.1 Motivation

Once fragments can be specified, documents can move from a cut-and-paste model to an inline reference model, where the fragment is served from an origin site. Composite documents are those that combine fragments using inline references. One reason for preferring inline fragment references to cut-and-paste is that the etiology of the fragment is preserved. Thus, meta-data such as digital signatures can be carried forward and a document consumer can easily follow references back to sources to examine original contexts.

An example might be citing some sentences from Presidential policy speech in a decision memorandum. Authenticity can be checked because the original document and its digital signature are available. But, what if the citation was out of context? Someone else could use the citation syntax to refer to the citation of a fragment by the composite document and assert that the fragment citation was out of context as it built an argument against the logic of the decision memorandum.

3.5.2 Discussion

Reference of a fragment as cited in a composite document is accomplished by using a position specifier to give the <ORIGIN-POSITION> in the composite document. The cited fragment begins at the origin position and continues for the full extent of the fragment. Thus, <ORIGIN-POSITION> provides the alignment in the citing document while the dimensions or extent is carried by the fragment reference. By keeping knowledge of the dimensions of the fragment out of the coordinate framework of the composite document, any position scheme can use the same generic citation syntax.

The origin position is defined as the point closest to the start of the document. In a media type structured as a single sequence of characters, the origin position is character 0. In a three dimensional cartesian coordinate systems, the origin position is

0,0,0. When each content type token is made available for fragment citation, the origin position MUST be defined.

The following identifier

pdi://oma.eop.gov.us/1997/11/03/4.text.1@103=pdi://oma.eop.gov.us/1997/09/01/1.text.1#37,51

refers to the citation of the fragment

pdi://oma.eop.gov.us/1997/09/01/1.text.1#37,51

by the document,

pdi://oma.eop.gov.us/1997/11/03/4.text.1

The citing document PDI uses the fragment text from character 37 to 51 starting from character 103. If the fragment PDI had no fragment specifier, the entire document would appear starting at position 103.

3.6 Operations on PDIs

The three classes of operations on PDIs have slightly different characteristics.

3.6.1 Minting

When PDIs are minted, they MUST carry a format and a version number and they MUST NOT contain any wildcards. This ensures that the identifiers associated with digital resources are fully specified and convey both the media type and the version number. The presence of a version number makes it possible to check for a higher version of the resource. Together, the format and version number enable fragment citation.

3.6.2 Binding

Once minted, a PDI can be bound to directly to a resource or indirectly via a Uniform Resource Indicator (URI) [2], which may often be a URL. Binding to a resource commits the PDI to a specific sequence of bytes. Therefore, a URI that is indirectly bound to a PDI MUST NOT change. If the URI changes, the indirect binding MUST be broken, and the original machine representation associated directly with the PDI.

3.6.3 Resolution

PDIs may be resolved to URLs, or other locators using recent URN resolution standards, such as THTTP described by RFC 2169 [10] and these resolvers may be discovered using the DNS extensions for

Naming Authority PoinTeR (NAPTR) described by RFC 2168 [8]. As these experimental resolution standards evolve, or new ones are introduced, PDI resolution can track any new standards precisely because a URN namespace is independent of the method used to resolve them.

When requesting a PDI from a URN resolver, the omission of a version number is a request for the resolver to return to the highest version available for the PDI. When defaulting a PDI to the highest version, the resolver MUST indicate to the client the fully qualified PDI associated with the media object. For THTTP, when a server returns an entity, the server MUST include a content location header [9] containing the fully-qualified PDI. This allows the client to associate the entity body with the versioned PDI that specifically identifies it.

When submitting a PDI to a URN resolver, wildcards may be used to obtain information for sets of PDIs. However, the set of returned PDIs MAY be complete only with regard to the specific knowledge about a document series available to a resolver at the time.

3.6.4 Lexical Equivalence

PDIs can be lexically compared for equivalence after they are converted to canonical form using the following procedure:

1.Unescape all escaped characters that are within <URN-CHARS> but which are not <RESERVED-CHARACTERS>.

2. Downcase the two <HEX> characters following the escape character "%".

3. Downcase all PDI components except <UNIQUE-ID> and <POSITION>.

4. If it is an encapsulated identifier, canonicalize <UNIQUE-ID> according the rules for the foreign identifier.

5. If <POSITION> is case-insensitive, as indicated by <FRAGMENT-SCHEME>, downcase position.

Wildcards carry a semantic interpretation that is not relevant for lexical equivalence. Two PDIs are lexically equal if and only if any wildcards appear in exactly the same positions in both.

3.6.5 Assertions

With the advent of URN standards, networked assertion infrastructures can now associate assertions (meta-data) with the unique identifier of a resource rather than replicating that information with every instance of a resource and creating a variety of synchronization problems. On this view, each URN namespace may also become an

assertion domain with specific semantics.

A document series thus becomes an address space in which each PDI serves as a pointer. Assertions about PDIs are assertions about the digital objects or fragments to which they refer. The ability to make and retrieve assertions about PDIs provides a means to associate meta-data with digital objects. For example, collaboration systems may use a typed link semantics to structure resources in meaningful ways. Alternatively, security systems may assert digital signatures or other trust information about PDIs. For example, a digital signature may be attached to some mobile code to check its integrity regardless of its proximate source. Access control systems might even assert differential access to various components of a single digital object.

Several groups are developing standards for associating meta-data with resources [11] [16]. These approaches are currently evolving and as yet lack a suitable persistent identifier model. URNs [17] [19] [20] and specifically PDIs offer a suitable persistent identifier for use with assertion schemes. The ability to refer to tokens and relations as first-class objects will dramatically simplify equality testing and significantly enhance the power and flexibility of the emerging standards for wide-area assertion infrastructures.

3.7 Registering New Fragment Specification Schemes

[TBD: This section will provide rules for registering new fragment specification schemes with IANA.]

3.8 Interpreting PDIs as Uniform Resource Locators

Current URN resolver discovery [8] and URN resolution [7] standards are experimental and lack wide deployment. As these standards evolve and become more wide spread, an interim resolution strategy using existing servers and clients for HTTP may prove useful. This section defines use of PDIs as URLs [1] with standard HTTP servers.

PDIs MAY be interpreted as Uniform Resource Locators (URL) and MAY be used in contexts where URLs are appropriate, such as with HTTP servers. When interpreting PDIs as URLs, PDIs do not carry URN prefix.

```
<URL> = pdi             ; Encoding in URL syntax
```

In all other ways, the syntax and semantics of PDIs remains exactly the same as under the URN namespace interpretation. However, resolver discovery and resolution under the URL interpretation are somewhat different.

3.8.1 Resolver Discovery

While an appropriate resolver for a PDI document series SHOULD be

found using current standards [8], an HTTP resolver MAY also be discovered heuristically by interpreting the document series as a domain name and looking up an IP address associated with the name. If a PDI-aware HTTP server is operating at the IP address and it successfully answers HTTP requests on the document series, then assume that a resolver for the document series has been located.

A client MAY also use new resolver discovery standards as they emerge or out of band methods to locate a resolver for a document series.

Since the heuristic discovery method provides no indication concerning the completeness or authority of the resolver, server operators SHOULD ensure that any server providing PDI resolution has complete knowledge of the document series, whether served locally or proxied from remote servers. Thus, if a server can resolve one PDI in a document series, the server operator guarantees the assumption that the server has sufficient knowledge to resolve any PDI in the series. An HTTP 1.1 or higher client MAY check whether an HTTP server resolves a PDI document series by issuing a HTTP request using the OPTIONS method on the PDI.

3.8.2 Resolution Methods

Since the HTTP standard [9] provides for use of any URI [2] in HTTP requests, no immediate extensions to the HTTP standard are required for PDI resolution. Requests for PDIs are just like requests for URLs, except that there are no relative PDIs; the scheme and document series MUST always be provided. Fully-specified PDIs allow the server to distinguish PDIs from ordinary URLs using the HTTP scheme. Servers SHOULD invoke augmented parsing for PDIs, for example, as necessary for fragment resolution.

```
<HTTP-REQUEST-LINE> = <method> " " <pdi> " " <http-version>
                      <crlf>
```

HTTP provides a series of methods on URIs. Below we define the operations for each method in HTTP 1.1 with respect to PDIs.

GET Resolves the PDI and returns the resource.

HEAD Returns the meta-data of the PDI as headers.

OPTIONS Returns the HTTP operations supported for the PDI.

TRACE returns information on the path to the origin server through proxies.

Non-idempotent HTTP methods interact with PDI semantics and require special handling.

PUT associates a resource with a PDI. If the PDI already

exists, the server MUST increment the PDI version number and store the resource under this new version. If the PDI does not exist, the server SHOULD return 404 "Not Found". If the client wishes to assign a new PDI, the PUT request MUST indicate the document series by providing a partial PDI:

<DOCUMENT-SERIES-IDENTIFIER> = "pdi://" document-series "/"

When a server receives a <document-series-identifier> as the URI in a PUT method, it MUST assign a unique PDI within the document series using the current <ISO-DATE>, a daily <UNIQUE-ID>, an appropriate <FORMAT>, and a <VERSION> equal to 1.

DELETE is not defined for PDIs as they are long-lived, persistent identifiers. An HTTP server MUST return 405 "Method Not Allowed".

When an HTTP method is applied to an unknown PDI, the server MUST return 404 "Not Found."

For all HTTP methods, whenever the server is unable to apply the method to an existing resource or to assign a new PDI, it MUST return 405 "Method Not Allowed."

3.8.3 Proxying HTTP Resolution

When a server is proxying PDI operations to an upstream HTTP server, it MUST pass through all requests and responses. The server contributes its knowledge of an origin server that can resolve the PDI. A server unable or not wishing to proxy PDI operations but aware of a server capable of handling the operations SHOULD redirect the client to the PDI-capable server.

3.8.4 Proxying THTTP Resolution

When the upstream server implements URN resolution, the proxy SHOULD perform protocol translation. For THTTP [7], the server SHOULD perform the following request translations.

Request	Proxy Request
GET <PDI>	GET "/uri-res/N2R?urn:" <PDI>
HEAD <PDI>	GET "/uri-res/N2C?urn:" <PDI>

Responses from the THTTP resolver SHOULD be passed through. For HTTP methods other than GET and HEAD, the server MUST return a 405 "Method Not Allowed"

4. History

Persistent Document Identifiers (PDI) were developed in early 1994 in order to provide persistent, location-independent identifiers for electronic publications. PDIs were deployed in Fall, 1994 when the second White House Electronic Publications System [21] was brought online. Every document published by the system since January 20, 1993 now carries a PDI. An identifier that was independent of protocol and transport proved extremely useful in managing this document set and resolving delivery failures. The publications server currently issues PDIs and resolves them using THTTP URN resolution [7].

The White House publications are an excellent example of documents that are monotonic (not subject to revision), widely mirrored, and subject to eventual relocation. Documents are never revised after they issue. Instead, they may be superseded by a corrected versions, but corrections are limited to transcription errors. Many sites around the world archive and redistribute the documents. These include major online services, libraries, advocacy groups, government entities. As of March 1996, it was estimated that about one million people around the world read at least some part of a document during the course of a week. At the end of an administration, the White House ceases to serve a former President's documents and they must be relocated to the National Archives, and normally, to his Presidential library. In sum, the White House documents are mirrored in many locations from which they may be obtained and the primary document repository must move after a period of time.

Persistent Document Identifiers were also used in an advanced experiment in large-scale, asynchronous collaboration during the Vice President's Open Meeting on Government Reinvention in December 1994. [12] In the Open Meeting, PDIs not only identified resources but they also associated a collaboration semantics with resources. A variety of meta-data and annotations were attached to resources via PDIs. More generally, PDIs were used to build the persistent semantic network around resources that allowed arbitrary assertions using first-class links, each with their own PDIs. Both the document database and the semantic network were mirrored at another site using PDIs to align all structures.

5. Acknowledgments

This specification was improved by comments from Andrew J. Blumberg, Mitchell N. Charity, Ron Daniel jr., Henrik Frystyk Nielsen, Jerry Saltzar, Karen R. Sollins, Christopher R. Vincent. This specification was revised and extended from an earlier draft dated December, 1994.

This specification describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the M.I.T. Artificial Intelligence Laboratory's artificial intelligence research is provided in part

by the Defense Advanced Research Projects Agency of the Department of Defense under contract numbers MDA972-93-1-003N7 and F30602-97-2-0239.

6. References

[1] T. Berners-Lee, L. Masinter, M. McCahill, ``Uniform Resource Locators (URL)'', RFC 1738, December, 1994. http://ds.internic.net/rfc/rfc1738.txt

[2] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax and Semantics", Internet Draft (work in progress), November 5, 1997.

[3] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, September 1993. http://ds.internic.net/rfc/rfc1521.txt

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, ``Extensible Markup Language (XML),'' World Wide Web Consortium, August, 1997, http://www.w3.org/TR/WD-xml

[5] T. Bray, S. DeRose, ``Extensible Markup Language (XML): Part 2. Linking,'' World Wide Web Consortium, July, 1997. http://www.w3.org/TR/WD-xml-link

[6] D. Crocker, P. Overell, "Augmented BNF for Syntax Specifications: ABNF," Internet Draft (work in progress), January 1997.

[7] R. Daniel, "A Trivial Convention for using HTTP in URN Resolution," RFC 2169, June, 1997. http://ds.internic.net/rfc/rfc2169.txt

[8] R. Daniel, M. Mealling, "Resolution of Uniform Resource Identifiers using the Domain Name System," RFC 2168, June, 1997. http://ds.internic.net/rfc/rfc2168.txt

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2068, January, 1997. http://ds.internic.net/rfc/rfc2068.txt

[10] N. Freed, J. Klensin, J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, November, 1996. http://ds.internic.net/rfc/rfc2048.txt Registered media types can be found at: ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types

[11] R. V. Guha, T. Bray, "Meta Content Framework Using XML," W3C Technical Note NOTE-MCF-XML, June, 1997.

[12] R. Hurwitz, J. C. Mallery, ``The Open Meeting: A Web-Based System for Conferencing and Collaboration,'' Proceedings of The Fourth International Conference on The World-Wide Web, December 12, 1995. Also in Web Journal, Winter, 1996, 1(1): 17-36.

[13] ISO 3166:1988 (E/F) - Codes for the representation of names of countries - The International Organization for Standardization, 3rd edition, 1988-08-15. ISO 3166 country codes can be found at: ftp://ftp.isi.edu/in-notes/iana/assignments/country-codes

[14] ISO 8879 Information Processing -- Text and Office Systems -- Standard Generalized Markup Language (SGML), ISO 8879:1986. For the list of SGML entities, consult ftp://ftp.ifi.uio.no/pub/SGML/ENTITIES/.

[15] ISO-8859. International Standard -- Information Processing -- 8-bit Single Byte Coded Graphic Character Sets --
Part 1: Latin alphabet No. 1, ISO 8859-1:1987.
Part 2: Latin alphabet No. 2, ISO 8859-2, 1987.
Part 3: Latin alphabet No. 3, ISO 8859-3, 1988.
Part 4: Latin alphabet No. 4, ISO 8859-4, 1988.
Part 5: Latin/Cyrillic alphabet, ISO 8859-5, 1988.
Part 6: Latin/Arabic alphabet, ISO 8859-6, 1987.
Part 7: Latin/Greek alphabet, ISO 8859-7, 1987.
Part 8: Latin/Hebrew alphabet, ISO 8859-8, 1988.
Part 9: Latin alphabet No. 5, ISO 8859-9, 1990.

[16] O. Lassila, R. R. Swick, "Resource Description Framework (RDF): Model and Syntax," World Wide Web Consortium, Technical Note RDF-Syntax (work in progress), August, 1997.

[17] R. Moats, "URN Syntax," RFC 2141, May 5, 1997. http://ds.internic.net/rfc/rfc2141.txt

[18] J. Reynolds & J. Postel, ``Assigned Numbers,'' STD 2, RFC 1700, October, 1994.

[19] K. Sollins, "Architectural Principles of Uniform Resource Name Resolution," Internet Draft (work in progress), July, 1997.

[20] K. Sollins, L. Masinter, "Functional Requirements for Uniform Resource Names," RFC 1737, December, 1994. http://ds.internic.net/rfc/rfc1737.txt

[21] White House Electronic Publications System, http://www.pub.whitehouse.gov

6. Author's Address

John C. Mallery
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

545 Technology Square, NE43-797
Cambridge, MA 02139 USA
Email: JCMa@ai.mit.edu
Phone: 617-253-5966