INTERNET-DRAFT J. Boynton
<draft-boynton-uol-04.txt> Produx House, Corp.
Expires six months from --> 5 August, 2001

Uniform Object Locator -- UOL

Status of This Memo

Abstract

A Uniform Object Locator (UOL) provides an architecture neutral mechanism for dynamic creation of unique, human readable identifiers for object oriented and relational data. UOL is designed to meet the recommendations for URI queries laid out in "Uniform Resource Identifiers (URI) Generic Syntax" [RFC2396].

This document defines syntax and semantics of UOL, including both absolute and relative forms, and guidelines for their use; it revises features, definitions, and examples, given in "draft-boynton-uol-03" and updates the scheme to provide additional functionality.

Table of Contents

1 Introduction

The UOL concept uses the logical location and relationship of data as a base for generating unique identifiers. Once mapped to a UOL id, data can be manipulated in memory by conventional programming tools and algorithms such as: hash tables, stacks, dynamic arrays etc..

The practical benefit of this approach is that an application can deconstruct the UOL, as needed, to derive additional information about the data source; even if such logic was not anticipated during the initial design process. Thus allowing programs to achieve many of the benefits of Object Oriented design without regard to application environment.

1.1 Purpose

The Uniform Object Locator (UOL) scheme is designed to encapsulate the location and relational context of data into a single unique identifier. This draft defines the hierarchical structure UOL and proposes relative forms of UOL for simplified concatenation and

efficient serialization of UOL within a data stream.

1.3 General Description

The UOL is constructed from a hierarchical list of elements in similar fashion to a URL path component; with the '/' forward slash as the delimiter for directory elements. The lower-most directory element may contain a single attribute delimited by the commercial "at" sign "@". A double slash "//" is used to define an authority element.

UOL maps a data structure by providing two separate components that define strings of related hierarchical elements. The two components are separated by a '#' (crosshatch), and divide the UOL into re-usable segments; an object constructor and object name. When used together, the two segments form a unique all-purpose identifier.

1.2 Terminology

For clarity, the specific role of elements within the UOL are defined below.

element
: A single, un-delimited, string of characters.

directory
: A typical hierarchical string of elements delimited by a '/' (forward slash).

attribute
: An element descending from a directory that terminates a constructor path.

schema
: For this draft, schema is a subset of nodes, directories, or attributes, defined by a schema, DTD(Document Type Definition), or class definition.

object
: A collection of directories and/or attributes defined by a schema. An object can be visualized as a re-usable tree of nodes descending from a single directory.

container
: A branching node that extends an object and servers as the parent component to one or more named child objects.

constructor

A directory path descending from an authority component and terminating with a single attribute; where no attribute is given, the path itself.

object name
A hierarchical collection of dot "." delimited elements that, when used together, provide the full name for a UOL constructor.

name element
An individual element within an object name. Its value is mapped to an object element within a constructor.

top level
The highest node, of a given type, within a hierarchy.

2. UOL Syntactic Components

UOL characteristics are derived from widely implemented URL syntax and semantics. A UOL consists of a scheme, authority, constructor and name fragment. UOL may also be appended to a URI as a query component; [RFC2396]. The UOL, itself, does not define a query component.

<scheme><authority><constructor>#<name>

Where used as a query to a URL, the authority and path component of the URL comprise the authority for the UOL

<scheme><authority-path-component>?<constructor>#<name>

2.1 Path Components

2.1.1 Authority

The authority component defines a root directory for resolving a UOL path to an absolute form. It may be a file, URL, or any name that identifies the resource being accessed. All elements within the UOL are bound by the context of the authority and may only resolve to elements that descend from the authority.

2.1.2 Constructor

The constructor component maps the relationship of objects, directories, and attributes through the use of a path metaphor. In most cases, the constructor does not map the actual location of data. Instead, the constructor is used as a pattern for identifying the relationships between an object and its various elements. The constructor is composed of directory elements and is terminated by a single attribute.

2.1.3 Name

The name component is comprised of name elements that are mapped to object nodes within the constructor. Name elements are appended,

from left-to-right, corresponding to the object element each identifies.

The constructor and name component are separated by a crosshatch

"#". Name elements are delimited by a dot ".".

2.2 Absolute and Relative

A UOL can be expressed in absolute and relative forms. An absolute UOL will resolve to, or contain, a top level element that is preceded by two forward slashes "//" (the authority).

For directory and attribute elements descending from the authority, UOLs beginning with "..", ".", or "/", are resolved in a manner consistent with the behavior for URI path components in Section 5.2 [RFC2396]. The prefix, itself, is not considered an element of the UOL path.

UOL provides two switches that can be appended to the above components; dollar sign "$" and tilde "~". The switches limit parsing to object and container elements respectively. When used, the switches have the effect of masking trivial elements within UOL constructor.

The following components are defined for resolving a partial UOL to:

"/" = AUTHORITY

"." = the current DIRECTORY

".." = the next higher DIRECTORY.
Combinations of this component will resolve to higher directory nodes but will not resolve above the authority.

"$" = the top level OBJECT

".$" = the current OBJECT.

"..$" = the parent OBJECT.
Combinations of this component will resolve to higher object elements but will not resolve above the top level object element (i.e. it will not resolve to the authority).

"~" = the top level CONTAINER

".~" = the current CONTAINER.

"..~" = the parent CONTAINER.
Combinations of this component will resolve to higher container elements but will not resolve above the top level container element.

In each case, the elements above should be followed by a "path delimiter"; the forward slash "/" for directories and the commercial

at sign "@" for attributes.

Rules for parsing relative components:

Given that the "." dot character is used to identify object elements as well as the relative components. The following tests are used to disambiguate the two uses.

1) if "." is followed by a path delimiter, the element is relative.

2) if "." is followed by a switch (dollar sign "$" or tilde "~") and the switch is followed by a path delimiter, the element is relative.

3) if the "." dot is followed by any character other than a path delimiter and the result of test two is false, the element is an object.

2.3 Comment Marker

The UOL scheme reserves the exclamation point character "!" for marking the UOL as a references to a "comment" value. The comment provides user information regarding the data. It is not intended to provide machine level implementation instructions.

The marker is reserved for use as a UOL prefix. Its meaning modifies the entire relative or absolute UOL to which it is applied. A Marked UOL must not have white space between the marker and the UOL itself.

example
"!//authority/.object@att" = comment for att

3. Path Elements

3.1 Directory

A directory element is a generic element used to organize a UOL constructor into a logical hierarchical path. Its usage is similar to its counter-part in an ordinary file system. Object and container elements, discussed later, are both directory elements.

In relative forms of UOL, directory references must not be ambiguous. If a partial UOL begins with a directory and the directory is not an object or container element, it must be preceded by a relative component; section 2.2.

example
abs = "/directory"
rel = "./directory"

3.2 Attribute

An attribute element is the UOL equivalent to a file name in an ordinary file system. The element always descends from a directory and terminates a constructors path.

Attributes do not ordinarily carry an extension (e.g. .txt, .exe, etc...). To prevent an attribute reference from being ambiguous, the "@" (commercial at sign) character is used as a delimiter between directory and attribute elements.

example
ref = "/directory@attribute"
ref = "/@attribute"

Relative forms of UOL may omit the "@" sign if the reference is to an attribute in the current directory.

example
cur dir = "/directory"
ref = "attribute"
resolved = /directory@attribute

3.3 Object

In the UOL scheme, an object element is the root node for a tree of elements defined by a schema; section 1.2. It is also a directory element. An object element is identified by the prefix "." (dot).

object reference
"/.object/dir@attr"

It is important to note that UOL defines relative components that also begin with a dot ".". Rules for differentiating between the two uses are discussed in Section 2.2.

3.4 Container

An element with a prefix of two or more dots (e.g. ".." or "...") is interpreted as a "container". All containers extend object elements. The container element shares an objects name space as well as its object name (if any). It does not, however, reference the objects attributes. Instead, the container diverts the UOL path to named object nodes within its scope. The UOLs object name is then extended to identify the branching elements.

example
"/..container/dir@field#name"

Container elements should be followed by a path delimiter ("/" or "@"). If the container is not followed by a path delimiter, the UOL is a reference to the container itself; the "name element" is not parsed.

Containers can be nested. The name element assigned to the nested

container is appended to the object name inherited from its parent. The nested name element is delimited from the object name by a dot "." character.

example
"/..parent/..nested/dir@field#parent_name.nested_name"

In the UOL scheme, object and container elements are mutually exclusive (they cannot be visible at the same time). Because a container must extend an object, its presence implies the object and, therefore, eliminates the need to express both. This approach increases the efficiency of UOL by reducing the number of elements in the constructor.

When resolving partial UOL with relative components, the "object/container" name space is treated as a single element. Switch characters within the relative component are used to differentiate between overlapping object and container elements; section 2.2.

3.4.1 Table Container

As the name implies, table containers are used to encapsulate tabular data structures. Each named sub-set of elements below a table container use the same schema. The schema used by a table container is inherited from the object element it extends. For example, if an object defines the field "price", so too will the container.

example UOL
object instance = "/.invoice@price"
container instance = "/..invoice@price#name"

In addition to suppling a schema, the containers object instance may also be used to hold default values for populating each named instance within the container.

3.4.2 List Container

A List container is used to encapsulate an array of unrelated objects. Because the list is an array, the name for objects enclosed by the list must be in the form of an integer equal to, or greater than, zero (0).

List containers can be used to describe SGML documents and popular sub-sets of SGML, such as HTML and XML. The following UOLs illustrate this concept by describing attributes within the HTML table below (An extended example is shown in section 6.1).

```
<table cellspacing=0 cellpadding=0 border=10>
     <tr>
      <td align="Left" valign="Top">Text one</td>
      <td align="Center" valign="Top">Text two</td>
   </tr>
```

```
   </table>

   ref to object
      .table@border="10"
```

ref to object in a container
...table/...tr/.td@align#0.1="Center"

A list container can also use undefined objects to bind a UOL to its value. The undefined object is implied when a list container is followed by a forward slash "/" and terminates a UOL constructor.

ref to default object
...table/...tr/...td/#0.1.0="Text two"

3.5 Name

Object elements enclosed by a container are assigned a unique name id. The name is appended to the object name (if any) inherited from the container itself; section 3.4. The name elements are delimited by a "." (dot).

example
"/..container/.object/#obj_name"
"/..container1/..container2/.object/#con2_name.obj_name"

4 Name Substitution

This section proposes a minimum set of parsing capabilities that should be supported for name substitution. All substitute name elements must be fully interpreted before the UOL is considered to be valid.

4.1 Wild Cards

The asterisk "*" is reserved as a wild card for name elements. Where present, an application should return a list of matching UOLs delimited by a line feed character (U000A).

4.2 Index Names

UOL should allow name substitution with index values. The index name is an integer >= -1 and inclosed by "[" and "]" (open and closed square brackets). For any name element, the integer index may be used in place of the actual name.

A name elements index is application dependant and does not have to conform to a specific algorithm. However, it should be the case, for any un-modified collection of UOL, the index will consistently return the same element.

parsing rules:

1) If the name contains square brackets and the contents cannot be

parsed to an integer >= -1, the UOL is invalid.

2) If a name cannot be located from the supplied index, the UOL is invalid.

3) If the integer is equal to -1, the index is the largest valid index plus 1.

4.3 Default Name

The "" (empty string) is reserved for naming default object elements. Applications can use the default object for storing schemas or default values.

ref to default object
"//authority/..object1/..object2@attr#"

Where a named object appears with a
default object, the "." (dot) delimiter implies the default name.

default name (leading)
"//authority/..object1/..object2@attr#.name2"

default name (trailing)
"//authority/..object1/..object2@attr#name1"

5 Reserved Attributes and Parameters

UOL does not define a language for interacting with its components. It does, however, reserve the semicolon ";" character for delimiting parameters as described in section 3.3 of [RFC2396]. Although supporting such functionality is optional, at minimum, a UOL parser should remove the parameter where identified.

In addition to parameter support, UOL reserves a set of case insensitive, directory specific, attributes. The attributes are used to return and set information regarding directory and object structure.

Directory level attributes operate on the directory node itself. In other words, they do not extend a schema. Directory attributes are identified by doubling the attribute delimiter "@@".

example
"/directory@@attlist"

The following are a minimum set of directory level attributes proposed for UOL.

names - The names parameter operates exclusively on container elements. If the container is a table, its value is a comma "," delimited string of name elements. If the container is a list, the parameters value is "[n]" where 'n' is the number of elements in the list.

schema - Used to obtain a line delimited list of UOL strings
representing a complete schema for sub-elements of the current
directory. The list provides partial UOL constructors that will

resolve correctly when combined with the current path. Each element path is enumerated in turn. The object name fragment (if any) is removed. This attribute is "read-only".

attlist - a non-recursive, comma "," delimited list of elements contained in the current directory.

code - the class description or file for an object that will use data from this directory.

codebase - the path description or URL to a class described by "code".

aka - "Also Known As" holds a single, alternate name for describing an object element. Where present, "aka" should provide the most common generic substitute for the actual name used.

functions - Used to retrieve or update a line delimited list of functions that will interact with values associated with a UOL directory. functions should be stated in a form consistent with a "classid" URI; Section 13.3 [REC html 4.01].

6 Example UOL

The following section provides examples of common data structures expressed in the form of UOL.

6.1 HTML Table

```
<table cellspacing=0 cellpadding=0 border=10>
     <tr>
      <td align="Left" valign="Top">Text one</td>
      <td align="Center" valign="Top">Text two</td>
   </tr>
     <tr>
      <td><img src="myimage.gif"></td>
   </tr>
</table>
```

The following shows the above html source expressed in UOL. Note that with UOL, changing the logical order of tags will not effect the meaning of the table.

```
.table@cellspacing                  = 0
.table@cellpadding                  = 0
.table@border                       = 0
...table/...tr/.td@align#0.0        = Left
...table/...tr/.td@valign#0.0       = Top
...table/...tr/...td/#0.0.0         = Text one
...table/...tr/.td@align#0.1        = Center
```

```
   ...table/...tr/.td@valign#0.1       = Top
   ...table/...tr/...td/#0.1.0         = Text two
   ...table/...tr/...td/.img@src#1.0.0 = myimage.gif
```

The following shows the above list in a serial form.

```
.table
cellspacing = 0
cellpadding = 0
border      = 0
...table
...tr#0
.td#0
align       = Left
valign      = Top
...td
.#0         = Text one
..
.td#1
align       = Center
valign      = Top
...td
.#0         = Text two
..
..
...tr#1
...td#0
.img#0
src         = myimage.gif
```

6.2 Relational Database

```
----------------------
|PK_Entity |  Name  |
----------------------
|   101    | Smith  |
|   102    | Jones  |
|   103    | Carter |
----------------------

--------------------------------------
|PK_Meeting | Start_Time | Stop_Time |
--------------------------------------
|    201    |   8:00 am  |   8:30 am |
|    202    |   9:00 am  |   9:30 am |
--------------------------------------
```

Attendees Table:

```
--------------------------
| FK_Entity | FK_Meeting |
--------------------------
|    101    |     201    |
|    102    |     201    |
```


| 102 | 202 |
| 103 | 202 |

```
   UOL for "Meeting" attendees

      constructor = "/..Meeting/..Entity@Name#"

      "/..Meeting/..Entity@Name#201.101" = Smith
      "/..Meeting/..Entity@Name#201.102" = Jones

   relative UOL for "Meeting" time
      "..@Start_Time#201" = 8:00 am

   A UOL can also expresses this data in the context of an appointment
   for a specific entity.

      constructor = "/..Entity/..Meeting@Start_Time#"

      "/..Entity/..Meeting@Start_Time#103.202" = 9:00 am
      "/..Entity/..Meeting@Start_Time#102.201" = 8:00 am

   relative UOL for attendee name
      "..@Name#103" = Carter
```

7 XML, XPath, and UOL

Interaction with XML data is one possible use for the UOL scheme. However, UOL is NOT proposed for use as an embedded identifier within XML schemas, DTDs, or payload. This functionality is provided by XML Path Language which was specifically written for this task. At the time this draft was updated, the W3C recommendation for XML Path Language was [XPath10];

8 References

[RFC2396] T. Berners-Lee, R. Fielding, and L. Masinter.
"Uniform Resource Identifiers (URI) Generic Syntax".
IETF RFC 2396 August 1998.

[HTML401] D. Raggett, A. Le Hors, and I. Jacobs, Editors,
"HTML 4.01 Specification".
W3C Recommendation HTML401, 24 December 1999.

[XPath10] J. Clark, and S. DeRose, Editors,
"XML Path Language (XPath) Version 1.0".
W3C Recommendation XPath10, 16 November 1999.

9 Author's Address

Jon L. Boynton
Produx House, Corp.
19300 Nalle Rd.
North Ft.Myers, FL 33917

Phone 941 543 4491
Email Comments to jon@datamessenger.com