Network Working Group
Internet Draft
<draft-ietf-find-cip-hierarchy-01.txt>

Chris Weider
Paul Leach
Microsoft Corp.
June 1997

Index Object for Hierarchical and Nonhierarchical Attribute-Value Data

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'.

WARNING: The specification in this document is subject to change, and will certainly change. It is inappropriate AND STUPID to implement to the proposed specification in this document. In particular, anyone who implements to this specification and then complains when it changes will be properly viewed as an idiot, and any such complaints shall be ignored. YOU HAVE BEEN WARNED.

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).

Distribution of this document is unlimited. Please send comments to the FIND working group at find@bunyip.com. Discussions of the working group are archived at <URL: ftp://ftp.bunyip.com/pub/mailing-lists/find>.

This Internet Draft expires on November 30, 1997.

1. Introduction

This document defines an index type for use with the Common Indexing Protocol [1]. This index type merges indexing of attributes which have hierarchical values and attributes which do not have hierarchical values. This document follows the conventions of the MIME definitions for CIP, detailed in [2].

2: Data model of the indexed data

This index type is designed to transmit data from information that is encoded in attribute-value pairs. This data may have attributes whose values are non-hierarchical, such as names; and values which are hierarchical, such as domain names or email addresses. The servers which

hold and transmit this data may have schema that define their holdings; this index type allows transmission of schema information as well.

3: Technical specification of the index type

3.1 Transmission Syntax

This index type contains nested multiply occurring blocks. These blocks are delimited by lines which start with the < character, and have comments indicating that they may be used multiple times. Keywords in angle brackets indicate the opening of a block, the corresponding keyword with a backslash in front of it and surrounded by angle brackets indicate the closure of the block.

3.2 Contents

Comments on the various syntactical components appear after the double slashes. They are not part of the contents.

The contents of the index type are as follows:

```
<INDEX>
Version: // version number of this format. Current value is 1.0
Start-time: // change list starting time, GMT
End-time: // change list ending time, GMT
Options: // which options the polled server was able to satisfy. Current
        // options are WEIGHT
Operation: // one of 3 keywords: ADD, DELETE, FULL
        // ADD - add these entries to the index for this server
        // DELETE - delete these entries from the index for this server
        // FULL - this is a full listing of the index between the times
        // indicated
Tokenization: // The tokenization used to generate the data. Can be
        // overridden by the  tokenization attribute on the individual
field
        // listings. Is one of TRUE or FALSE
Delimiter: // Character used in the tokenization algorithm. This field
may
        // be repeated. Non-printing characters must be represented as
hex
        // encodings.
<SCHEMA> // may occur multiple times
Template:
Field:
</SCHEMA>
<DATA>
<TEMPLATE> // may occur multiple times
Template:
Any-field: // See explanation below
<FIELD>
Field:
```

```
Hierarchy: // LEFT, RIGHT, or NONE
Tokenization: // TRUE or FALSE
Delimiter: // as above
Data: // Either the token *, or the value list itself, one per line,
        // cr/lf terminated. See below.
</FIELD>
</TEMPLATE>
</DATA>
</INDEX>
```

The token * as the only item of a Data: list means that any value for this field should be treated as a hit by the indexing server.

The field Any-field: can take two values, TRUE or FALSE. If the value is TRUE, the polled server is indicating that there are fields in this template which are not being exported to the polling server, but wishes to treat as a hit. Thus, when the polling server gets a query which has a term requesting a field not in this list for this template, the polling server will treat that term as a hit. If the value is FALSE, the pollee is indicating that there are no other fields for this template which should be treated as a hit. This field is required because the basic model for the CIP query syntax requires that the results of each search term be 'and'ed together. This field allows polled servers to export data for non-sensitive fields, yet still get referrals of queries, which contain sensitive terms.

3.3 Use with the generic CIP wrapper protocol

This is designed to be carried in the CIP wrapper protocol [1]. The 'type' argument that must be used by the CIP wrapper protocol when carrying this index type around is 'av-hierarchy'.

3.4 Base URI generation

Many protocols which use attribute-value stores have entry names which are hierarchical, for example, X.500 and LDAP. Others have entry names which are not hierarchical except when the handle of the server is prepended to the entry name, for example, WHOIS++. A base URI for this index type consists of the DNS name of the server which is generating the index. If the primary store of the index generator has hierarchical entry names, some sub-hierarchy of the entry names may be included in the base URI. For example, if a WHOIS++ server foo.bar.com generates an index, the base URI for the server must be whoispp://foo.bar.com. If an LDAP server foo2.bar.com generates an index, and contains the ou=Foo, o=Bar, c=France naming context, the base URI might be ldap://foo2.bar.com/o=Bar/c=France.

The intent here is to not allow base URIs to express query restrictions other than those that appear in the namespace for the entries on a given

server.

3.5 DSI policy

If the generating server is a base-level server, i.e. does not index any other server, it may assign a DSI to either the entire server or to a given naming context. It may not arbitrarily assign DSIs to random subsets of the data.

If the generating server is an index server, it may issue a DSI for its entire contents or it may issue a DSI for the entire set of data for which it contains actual entries. In addition, if it is not aggregating its contents into a single data set, it may also transmit DSIs for the indexes it holds.

3.6 Aggregation characteristics

This index type is designed to aggregate well. If a given attribute is hierarchical, the aggregating server may publish the values it holds for the attribute, or any hierarchical prefix for the value. I.e., if an aggregating server has the values foo.bar.com, foo2.bar.com, foo3.bar.com, it can publish bar.com or .com for its aggregated value. If a given attribute is non-hierarchical, and the aggregating server can detect a hierarchical pattern among the values of that attribute, the server can aggregate the attribute as a hierarchical attribute and publish prefixes for its values. A server may not aggregate a hierarchical
attribute as a non-hierarchical one, as too much information is likely to have been lost in the hierarchical aggregation.

3.7 Referral Generation Semantics and Matching Semantics

A server holding this index type generates a referral when an incoming query is satisfied. Terms in a query can be either typed or typeless: a typeless term gives just a value; a typed term gives both an attribute and a value (e.g. name=Chris). A value in the query matches a value in the index under the following conditions:

A: The value is compared against an attribute which has 'any-value' specified in the index
B: The attribute is non-hierarchical and some value in the index meets the search criteria given
C: The attribute is hierarchical and the index contains a value which is a substring of the query value.

If a query term is typeless, it is matched against all attributes which appear in the template specified in the query. If it is typed, it is matched against the attribute which matches the attribute name in the term.

3.8 Example

This example illustrates index data for a simple dataset.

```
<INDEX>
Version: 1.0
Start-time: 19970615061500Z
End-time: 19970701061500Z
Operation: FULL
Tokenization: TRUE
Delimiter: \b
<SCHEMA>
Template: SimplePerson
Field: Name
Field: TelephoneNumber
</SCHEMA>
<DATA>
<TEMPLATE>
Template: SimplePerson
Any-field: FALSE
<FIELD>
Field: Name
Hierarchy: NONE
Data: *
</FIELD>
<FIELD>
Field: TelephoneNumber
Hierarchy: LEFT
Tokenization: TRUE
Delimiter: (
Delimiter: )
Delimiter: -
Data: 425
</FIELD>
</TEMPLATE>
</DATA>
</INDEX>
```

4: Other operations

4.1 POLL

To request this data type in a poll request, the body must contain

```
Version: // version number of poller's index software, used to insure
compatibility. Current is 2.3
 Charset: // specifies character set in which the index changes are to
        // be transmitted. Must be one of ISO-8859-1 or
UNICODE-1-1-UTF-8
 Start-time: // send all the index changes starting at this time, GMT
 End-time: // ending at this time, GMT
<REQUEST> // This block may occur multiple times
```

```
 Template: // a standard template or object class name, or the keyword
        // ALL, for a full update.
 Field:    // used to limit index update information to specific fields,
      // is either a specific field name, a list of field names
        // separated by spaces, or the keyword ALL. May occur multiple
times
        // per template.
</REQUEST>
 Host-Name: // Host name of the polling server.
 Host-Port: // Port number of the polling server.
  Description: // This field contains a brief text description of the
                // polling server
Options: // Can be used to request the WEIGHT of the returned values
```

4.1.1 Example

```
Version: 2.3
Charset: UNICODE-1-1-UTF-8
Start-time: 19960101000000Z
End-time: 19970615120000Z
<REQUEST>
Template: ALL
Field: ALL
</REQUEST>
Host-Name: foo.bar.com
Host-Port: 65535
Description: North America Master Server
```

4.2 DATA-CHANGED

To use this index type in a data-changed message, the body must contain:

```
Version-number: // version number of index service software, used to
        // insure compatibility. Current value is 2.3
 Time-of-latest-index-change: // time stamp of latest forward
        // information change, GMT
 Time-of-message-generation: // time when this message was generated,
        // GMT
 Host-Name: // Host name of this server (current name)
 Host-Port: // Port number of this server (current port)
 Protocol: // Access protocol to use when speaking to this server
 Best-time-to-poll: // For heavily used servers, this will identify when
                    // the server is likely to be lightly loaded
                    // so that response to the poll will be speedy, GMT
```

4.2.1 Example

```
Version-number: 2.3
Time-of-latest-index-change: 19970615120500Z
Time-of-message-generation: 19970615140000Z
Host-Name: bar.foobar.com
```

Host-Port: 23456
Protocol: LDAP
Best-time-to-poll: 000000Z

5: Security Considerations

This index type does not have any security beyond that discussed in [2]. The primary potential security hole is that this index type may contain enough data for a BadGuy to recreate the underlying database. If you consider this a potential threat, you should use a secure transmission stream and authenticate the server at the other end.

6: References

[1] J. Allen, Michael Mealling, The Common Indexing Protocol, Internet Draft, June, 1997. Available as draft-ietf-find-cip-arch-00.txt
[2] J. Allen, Michael Mealling, MIME Object Definitions for the Common Indexing Protocol, Internet
Draft, June, 1997. Available as draft-ietf-cip-mime-00.txt

7: Author's addresses

Chris Weider, cweider@microsoft.com
Paul Leach, paulle@microsoft.com
1 Microsoft Way
Redmond, WA 98052
+1-425-882-8080

Appendix A: BNF

This BNF follows the Extended BNF defined in RFC 2068.

A.1 Contents of the index type

```
index = "<INDEX>" CRLF body CRLF "</INDEX>
body = ( header schema data )
header = ( version start end [ options ] operation tokens )
version = "Version:" timestring CRLF
start = "Start-time:" timestring CRLF
end= "End-time:" timestring CRLF
options = "Options:" optionstring CRLF
optionstring= "WEIGHT"
operation= "Operation:" ( "ADD" | "DELETE" | "FULL" ) CRLF
tokens= "Tokenization:"  ( ("TRUE" CRLF delimiter) | ("FALSE" CRLF))
delimiter= "Delimiter:" delimiterchar
schema= "<SCHEMA>" CRLF schemabody "</SCHEMA>" CRLF
schemabody= 1* (template 1*field )
template= "Template:" charstring CRLF
field= "Field:" charstring CRLF
```

```
data= "<DATA>" CRLF ( 1*templatedef ) "</DATA>" CRLF
templatedef = "<TEMPLATE>" CRLF templatebody "</TEMPLATE>" CRLF
templatebody= (template anyfield 1*fieldblock )
anyfield= "Any-field:" boolean CRLF
fieldblock= "<FIELD>" CRLF fielddef  "</FIELD>" CRLF
fielddef= ( field hierarchy tokens  dataval )
hierarchy= "Hierarchy:" ( "LEFT" | "RIGHT" | "NONE") CRLF
dataval= "Data:" ( ("*" CRLF) | 1*(charstring CRLF) )
timestring= (14*DIGIT "Z")      ; Time string in YYYYMMDDHHMMSS format,
GMT
delimiterchar= ( "." | "," | "/" | "\b" | "\t" | "\n" )   ; The escaped
characters mean what they do in C
charstring= 1*CHAR ; CHAR is any legal 10646 character
boolean= ( "TRUE" | "FALSE" )
```

A.2 Contents of the poll request

This uses the strings defined above.

```
pollrequest = ( version charset start end requestblock hostname hostport
description [  options ] )
charset= ( "ISO-8859-1" | "UNICODE-1-1-UTF-8" ) CRLF
requestblock = ( alltemplates | 1*sometemplate)
alltemplates= ( "<REQUEST>" CRLF "Template: ALL" CRLF "</REQUEST>" )
sometemplate= ("<REQUEST> CRLF templateblock </REQUEST>" CRLF )
templateblock= ( template ( allvalues | 1*field) )
allvalues= "Field:" "ALL" CRLF
hostname= "Hostname:" charstring CRLF ; This is a standard Internet host
name
hostport= "Hostport:" 1*DIGIT CRLF ; This is a standard port number,
between 1 and 65535
description= "Description:" charstring CRLF
```

A.3 Contents of the data changed notification

```
datachanged= (version lastchange messagetime hostname hostport
1*protocol besttime )
lastchange= "Time-of-last-index-change:" timestring CRLF
messagetime= "Time-of-message-generation:" timestring CRLF
protocol= "Protocol:" ("LDAP" | "RWHOIS" | "WHOIS++" ) CRLF
bestting= "Best-time-to-poll:" timestring CRLF
```