            The Architecture of the Common Indexing Protocol (CIP)

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   The Common Indexing Protocol (CIP) is used to pass indexing
   information from server to server in order to facilitate query
   routing. Query routing is the process of redirecting and replicating
   queries through a distributed database system towards servers holding
   the desired results. This document describes the CIP framework,
   including its architecture and the protocol specifics of exchanging
   indices.

1. Introduction

1.1. History and Motivation

   The Common Indexing Protocol (CIP) is an evolution and refinement of
   distributed indexing concepts first introduced in the Whois++
   Directory Service [RFC1913, RFC1914]. While indexing proved useful in
   that system to promote query routing, the centroid index object which
   is passed among Whois++ servers is specifically designed for
   template-based databases searchable by token-based matching. With
   alternative index objects, the index-passing technology will prove
   useful to many more application domains, not simply Directory
   Services and those applications which can be cast into the form of
   template collections.

The indexing part of Whois++ is integrated with the data access
protocol. The goal in designing CIP is to extract the indexing
portion of Whois++, while abstracting the index objects to apply more
broadly to information retrieval. In addition, another kind of
technology reuse has been undertaken by converting the ad-hoc data
representations used by Whois++ into structures based on the MIME
specification for structured Internet mail.

Whois++ used a version number field in centroid objects to facilitate
future growth. The initial version was "1". Version 1 of CIP (then
embedded in Whois++, and not referred to separately as CIP) had
support for only ISO-8895-1 characters, and for only the centroid
index object type.

Version 2 of the Whois++ centroid was used in the Digger software by
Bunyip Information Systems to notify recipients that the centroid
carried extra character set information. Digger's centroids can carry
UTF-8 encoded 16-bit Unicode characters, or ISO-8859-1 characters,
determined by a field in the headers.

This specification is for CIP version 3. Version 3 is a major
overhaul to the protocol. However, by using of a short negotiation
sequence, CIP version 3 servers can interoperate with earlier servers
in an index-passing mesh.

For unclear terms the reader is referred to the glossary in Appendix
A.

1.2 CIP's place in the Information Retrieval world

CIP facilitates query routing. CIP is a protocol used between servers
in a network to pass hints which make data access by clients at a
later date more efficient. Query routing is the act of redirecting
and replicating queries through a distributed database system towards
the servers holding the actual results via reference to indexing
information.

CIP is a "backend" protocol -- it is implemented in and "spoken" only
among network servers. These same servers must also speak some kind
of data access protocol to communicate with clients. During query
resolution in the native protocol implementation, the server will
refer to the indexing information collected by the CIP implementation
for guidance on how to route the query.

Data access protocols used with CIP must have some provision for
control information in the form of a referral. The syntax and
semantics of these referrals are outside the scope of this
specification.

2. Related Documents

   This document is one of three documents. This document describes the
   fundamental concepts and framework of CIP.

   The document "MIME Object Definitions for the Common Indexing
   Protocol" [CIP-MIME] describes the MIME objects that make up the
   items that are passed by the transport system.

   Requirements and examples of several transport systems are specified
   in the "CIP Transport Protocols" [CIP-TRANSPORT] document.

   A second set of document describe the various specifications for
   specific index types.

3. Architecture

3.1 CIP in the Information Retrieval World

3.1.1 Information Retrieval in the Abstract

   In order to better understand how CIP fits into the information
   retrieval world, we need to first understand the unifying abstract
   features of existing information retrieval technology. Next, we
   discuss why adding indexing technology to this model results in a
   system capable of query routing, and why query routing is useful.

   An abstract view of the client/server data retrieval process includes
   data sets and data access protocols. An individual server is
   responsible for handling queries over a fixed domain of data. For the
   purposes of CIP, we call this domain of data the dataset. Clients
   make searches in the dataset and retrieve parts of it via a data
   access protocol. There are many data access protocols, each optimized
   for the data in question. For instance, LDAP and Whois++ are access
   protocols that reflect the needs of the directory services
   application domain. Other data access protocols include HTTP and
   Z39.50.

3.1.2 Indexing Information Facilitates Query Routing

   The above description reflects a world without indexing, where no
   server knows about any other server. In some cases (as with X.500
   referrals, and HTTP redirects) a server will, as part of its reply,
   implicate another server in the process of resolving the query.
   However, those servers generate replies based solely on their local
   knowledge. When indexing information is introduced into a server's
   local database, the server now knows not only answers based on the

local dataset, but also answers based on external indices. These
indices come from peer servers, via an indexing protocol. CIP is one
such indexing protocol.

Replies based on index information may not be the complete answer.
After all, an index is not a replicated version of the remote
dataset, but a possibly reduced version of it. Thus, in addition to
giving complete replies from the local dataset, the server may give
referrals to other datasets. These referrals are the core feature
necessary for effective query routing. When servers use CIP to pass
indices from server to server, they make a kind of investment. At the
cost of some resources to create, transmit and store the indices,
query routing becomes possible.

Query Routing is the process of replicating and moving a query closer
to datasets which can satisfy the query. In some distributed systems,
widely distributed searches must be accomplished by replicating the
query to all sub-datasets. This approach can be wasteful of resources
both in the network, and on the servers, and is thus sometimes
explicitly disabled. Using indexing in such a system opens the door
to more efficient distributed searching.

While CIP-equipped servers provide the referrals necessary to make
query routing work, it is always the client's responsibility to
collate, filter, and chase the referrals it receives. This gives the
end-user (or agent, in the case that there's no human user involved
in the search) greatest control over the query resolution process.
The cost of the added client complexity is weighed against the
benefits of total control over query resolution. In some cases, it
may also be possible to decouple the referral chasing from the client
by introducing a proxy, allowing existing simple clients to make use
of query routing. Such a proxy would transparently resolve referrals
into concrete results before returning them to the simple-minded
client.

3.1.3 Abstracting the CIP index object

As useful as indices seem, the fact remains that not all queries can
benefit from the same type of index. For example, say the index
consists of a simple list of keywords. With such an index, it is
impossible to answer queries about whether two keywords were near one
another, or if a keyword was present in a certain context (for
instance, in the title).

Because of the need for application domain specific indices, CIP
index objects are abstract; they must be defined by a separate
specification. The basic protocols for moving index objects are
widely applicable, but the specific design of the index, and the

   structure of the mesh of servers which pass a particular type of
   index is dependent on the application domain. This document describes
   only the protocols for moving indices among servers. Companion
   documents describe initial index objects.

   The requirements that index type specifications must address are
   specified in the [CIP-MIME] document.

3.2 Architectural Details

   CIP implements index passing, providing the forward knowledge
   necessary to generate the referrals used for query routing. The core
   of the protocol is the index object. In the following sections, the
   structure of the index objects themselves is presented. Next, how and
   why indices are passed from server to server is discussed. Finally,
   the circumstances under which a server may synthesize an index object
   based on incoming ones are discussed.

3.2.1 The CIP Index Object

   A CIP index object is composed of two parts, the header and the
   payload. The header contains metadata necessary to process and make
   use of the index object being transmitted. The actual index resides
   in the payload.

   Three particular headers warrant specific mention at this point. The
   "type" of the index object selects one of many distinct CIP index
   object specifications which define exactly how the index blocks are
   to be created, parsed and used to facilitate query routing. Another
   header of note is the "DSI", or Dataset Identifier, which uniquely
   identifies the dataset from which the index was created. Another
   header that is crucial for generating referrals is the "Base-URI".
   The URI (or URI's) contained in this header form the basis of any
   referrals generated based on this index block. The URI is also used
   as input during the index aggregation process to constrain the kinds
   of aggregation possible, due to multiprotocol constraints. How that
   URI is used is defined by the aggregation algorithm. The exact
   syntax of these headers is specified in the CIP MIME specification
   document [CIP-MIME].

   The payload is opaque to CIP itself. It is defined exclusively by the
   index object specification associated with the object's MIME type.
   Specifications on how to parse and use the payload are published
   separately as "CIP index object specifications". This abstract
   definition of the index object forms the basis of CIP's applicability
   to indexing needs across multiple application domains.

A precise definition of the content and form of a CIP index block can
be found in the Protocol document [CIP-MIME]

3.2.2 Moving Index Objects: How to Build a Mesh

Indices are transmitted among servers participating in a CIP mesh. By
distributing this information in anticipation of a query, efficient,
accurate query routing is possible at the time a query arrives.

A CIP mesh is a set of CIP servers which pass indices of the same
type among themselves. Typically, a mesh is arranged in a
hierarchical tree fashion, with servers nearer the root of the tree
having larger and more comprehensive indices. See Figure 1. However,
a CIP mesh is explicitly allowed to have lateral links in it, and
there may be more than one part of the mesh that has the properties
of a "root". Mesh administrators are encouraged to avoid loops in the
system, but they are not obliged to maintain a strict tree structure.
Clients wishing to completely resolve all referrals they receive
should protect against referral loops while attempting to traverse
the mesh to avoid wasting time and network resources. See the
section on "Navigating the Mesh" for a discussion of this.

```
      base level              index                   index
      directory               servers                 servers
       servers                  for                     for
                            base level              lower-level
                             servers               index servers

       _____
      |        |
      |   A    |__
      |_____|  \
                   \---CIP----|        |
       _____               |   D    |__
      |        |   /---CIP----|_____|  \
      |   B    |__/                        \--CIP------|      |
      |_____|                                       |  F   |
                                            /--CIP------|_____|
                                           /
       _____      _____             /
      |        |    |        |-
      |   C    |-------CIP----|   E    |
      |_____|    |_____|-
                        |     \
                        r      \
       _____         e       \
      |        |        f        \--CIP-----|      |
      |   G    |-------CIP---------e-------------------|  H   |
      |_____|                  r                    |_____|
        \--referral---|           r      --referral-/
                      |           a      |

                      |           l      |

                      \ 3    | 2    | 1

                        \--------/

                        |         |

                        | client  |

                        |         |

                         --------
```
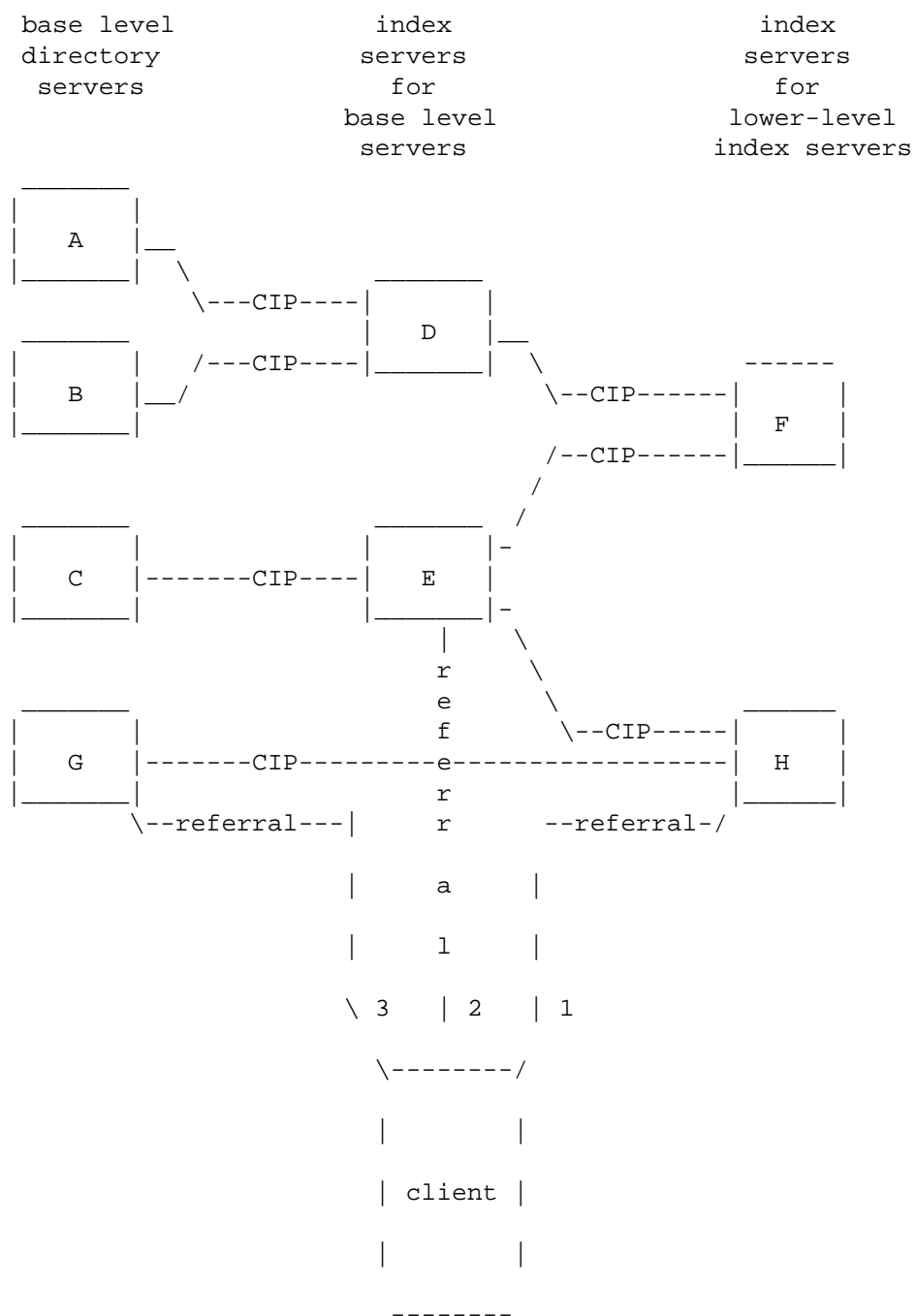
                Figure 1: Sample layout of the Index Service mesh

All indices passed in a given mesh are assumed, as of this writing,
to be of the same type (i.e. governed by the same CIP index object
specification). It may be possible to create gateways between meshes
carrying different index objects, but at this time that process is
undefined and declared to be outside the scope of this specification.

In the case where a CIP server receives an index of a type that it
does not understand it _can_ pass that index forward untouched. In
the case where a server implementation decides not to accept unknown
indices it should return an appropriate error message to the server
sending the index. This behavior is to allow mesh implementations to
attempt heterogeneous meshes. As stated above heterogeneous meshes
are considered to be ill defined and as such should be considered
dangerous.

Experience suggests that this index passing activity should take
place among CIP servers as a parallel (and possibly lower-priority)
job to their primary job of answering queries. Index objects travel
among CIP servers by protocol exchanges explicitly defined in this
document, not via the server's native protocol. This distinction is
important, and bears repeating:

    Queries are answered (and referrals are sent) via the native data
    access protocol.

    Index objects are transferred via alternative means, as defined by
    this document.

When two servers cooperate to move indexing information, the pair are
said to be in a "polling relationship". The server that holds the
data of interest, and generates the index is called the "polled
server". The other server, which is the one that collects the
generated index, is the "polling server".

In a polling relationship, the polled server is responsible for
notifying the polling server when it has a new index that the polling
server might be interested in. In response, the polling server may
immediately pick up the index object, or it may schedule a job to
pick up a copy of the new index at a more convenient time. But, a
polling server is not required to wait on the polled server to notify
it of changes. The polling server can request a new index at any
time.

Independent of the symmetric polling relationship, there's another
way that servers can pass indices using CIP. In an "index pushing"
relationship, a CIP server simply sends the index to a peer whenever
necessary, and allows the receiver to handle the index object as it

chooses. The receiving server may refuse it, may accept it, then
silently discard it, may accept only portions of it (by accepting it
as is, then filtering it), or may accept it without question.

The index pushing relationship is intended for use by dumb leaf nodes
which simply want to make their index available to the global mesh of
servers, but have no interest in implementing the complete CIP
transaction protocol. It lowers the barriers to entry for CIP leaf
nodes. For more information on participating in a CIP mesh in this
restricted manner, see the section below on "Protocol Conformance".
CIP index passing operations take place across a reliable transport
mechanisms, including both TCP connections, and Internet mail
messages. The precise mechanisms are described in the Transport
document [CIP-Transport].

3.2.3 Index Object Synthesis

From the preceding discussion, it should be clear that indexing
servers read and write index objects as they pass them around the
mesh. However, a CIP server need not simply pass the in-bound indices
through as the out-bound ones. While it is always permissible to pass
an index object through to other servers, a server may choose to
aggregate two or more of them, thereby reducing redundancy in the
index, at the cost of longer referral chains.

A basic premise of index passing is that even while collapsing a body
of data into an index by lossy compression methods, hints useful to
routing queries will survive in the resulting index. Since the index
is not a complete copy of the original dataset, it contains less
information. Index objects can be passed along unchanged, but as more
and more information collects in the resulting index object,
redundancy will creep in again, and it may prove useful to apply the
compression again, by aggregating two or more index objects into one.

This kind of aggregation should be performed without compromising the
ability to correctly route queries while avoiding excessive numbers
of missed results. The acceptable likelihood of false negatives must
be established on a per-application-domain basis, and is controlled
by the granularity of the index and the aggregation rules defined for
it by the particular specification.

However, when CIP is used in a multi-protocol application domain,
such as a Directory Service (with contenders including Whois++, LDAP,
and Ph), things get significantly trickier. The fundamental problem
is to avoid forcing a referral chain to pass through part of the mesh
which does not support the protocol by which that client made the
query. If this ever happens, the client loses access to any hits

beyond that point in the referral chain, since it cannot resolve the
referral in its native data access protocol. This is a failure of
query routing, which should be avoided.

In addition to multi-protocol considerations, server managers may
choose not to allow index object aggregation for performance reasons.
As referral chains lengthen, a client needs to perform more
transactions to resolve a query. As the number of transactions
increases, so do the user-perceived delays, the system loads, and the
global bandwidth demands. In general, there's a tradeoff between
aggressive aggregation (which leads to reductions in the indexing
overhead) and aggressive referral chain optimization. This tradeoff,
which is also sensitive to the particular application domain, needs
to be explored more in actual operational situations.

Conceptually, a CIP index server has several index objects on hand at
any given time. If it holds data in addition to indexing information,
the server has an index object formed from its own data, called the
"local index". It may have one or more indices from remote servers
which it has collected via the index passing mechanisms. These are
called "in-bound indices".

   Implementor's Note: It may not be necessary to keep all of these
   structures intact and distinct in the local database. It is also
   not required to keep the out-bound index (or indices) built and
   ready to distribute at all times. The previous paragraph merely
   introduces a useful model for expressing the aggregation rules.
   Implementors are free to model index objects internally however
   they see fit.

The following two rules control how a CIP server formulates its
outgoing indices:

1. An index server may pass any of the index objects in its local
   index and its in-bound indices through unchanged to polling
   servers.

2. If and only if the following three conditions are true, an index
   server can aggregate two or more index objects into a single new
   index object, to be added to the set of out-bound indices.

   a. Each index object to be aggregated covers exactly the same set
      of protocols, as defined by the scheme component of the Base-
      URI's in each index object.

   b. The index server supports every one of the data access
      protocols represented by the Base-URI's in the index objects to
      be aggregated.

      c. The specification for the index object type specified by the
         type header of the index objects explicitly defines the
         aggregation operation.

      The resulting index object must have Base-URI's characteristic of
      the local server for each protocol it supports. The outgoing
      objects should have the DSI of the local server.

4. Navigating the mesh

   With the CIP infrastructure in place to manage index objects, the
   only problem remaining is how to successfully use the indexing
   information to do efficient searches. CIP facilitates query routing,
   which is essentially a client activity. A client connects to one
   server, which redirects the query to servers "closer to" the answer.
   This redirection message is called a referral.

4.1 The Referral

   The concept of a referral and the mechanism for deciding when they
   should be issued is described by CIP. However, the referral itself
   must be transferred to the client in the native protocol, so its
   syntax is not directly a CIP issue. The mechanism for deciding that a
   referral needs to be made and generating that referral resides in the
   CIP implementation in the server. The mechanism for sending the
   referral to the client resides in the server's native protocol
   implementation.

   A referral is made when a search against the index objects held by
   the server shows that there may be hits available in one of the
   datasets represented by those index objects. If more that one index
   object indicates that a referral must be generated to a given
   dataset, the server should generate only one referral to the given
   dataset, as the client may not be able to detect duplicates.

   Though the format of the referral is dependent on the native
   protocol(s) of the CIP server, the baseline contents of the referral
   are constant across all protocols. At the least, a DSI and a URI must
   be returned. The DSI is the DSI associated with the dataset which
   caused the hit. This must be presented to the client so that it can
   avoid referral loops. The Base-URI parameter which travels along with
   index objects is used to provide the other required part of a
   referral.

   The additional information in the Base-URI may be necessary for the
   server receiving the referred query to correctly handle it. A good
   example of this is an LDAP server, which needs a base X.500
   distinguished name from which to search. When an LDAP server sends a

centroid-format index object up to a CIP indexing server, it sends a
Base-URI along with the name of the X.500 subtree for which the index
was made. When a referral is made, the Base-URI is passed back to the
client so that it can pass it to the original LDAP server.

As usual, in addition to sending the DSI, a DSI-Description header
can be optionally sent. Because a client may attempt to check with
the user before chasing the referral, and because this string is the
friendliest representation of the DSI that CIP has to offer, it
should be included in referrals when available (i.e. when it was sent
along with the index object).

4.2 Cross-protocol Mappings

Each data access protocol which uses CIP will need a clearly defined
set of rules to map queries in the native protocol to searches
against an index object. These rules will vary according to the data
domain. In principle, this could create a bit of a scaling
difficulty; for N protocols and M data domains, there would be N x M
mappings required. In practice, this should not be the case, since
some access protocols will be wholly unsuited to some data domains.
Consider for example, a LDAP server trying to make a search in an
index object composed from unorganized text based pages. What would
the results be? How would the client make sense of the results?

However, as pre-existing protocols are connected to CIP, and as new
ones are developed to work with CIP, this issue must be examined. In
the case of Whois++ and the CENTROID index type, there is an
extremely close mapping, since the two were designed together. When
hooking LDAP to the CENTROID index type, it will be necessary to map
the attribute names used in the LDAP system to attribute names which
are already being used in the CENTROID mesh. It will also be
necessary to tokenize the LDAP queries under the same rules as the
CENTROID indexing policy, so that searches will take place correctly.
These application- and protocol-specific actions must be specified in
the index object specification, as discussed in the [CIP-MIME]
document.

4.3 Moving through the mesh

From a client's point of view, CIP simply pushes all the "hard work"
onto its shoulders. After all, it is the client which needs to track
down the real data. While this is true, it is very misleading.
Because the client has control over the query routing process, the
client has significant control over the size of the result set, the
speed with which the query progresses, and the depth of the search.

The simplest client implementation provides referrals to the user in
a raw, ready-to-reuse form, without attempting to follow them. For
instance, one Whois++ client, which interacts with the user via a
Web-based form, simply makes referrals into HTML hypertext links.
Encoded in the link via the HTML forms interface GET encoding rules
is the data of the referral: the hostname, port, and query. If a user
chooses to follow the referral link, he executes a new search on the
new host. A more savvy client might present the referrals to the user
and ask which should be followed. And, assuming appropriate limits
were placed on search time and bandwidth usage, it might be
reasonable to program a client to follow all referrals automatically.

When following all referrals, a client must show a bit of
intelligence. Remember that the mesh is defined as an interconnected
graph of CIP servers. This graph may have cycles, which could cause
an infinite loop of referrals, wasting the servers' time and the
client's too. When faced with the job of tacking down all referrals,
a client must use some form of a mesh traversal algorithm. Such an
algorithm has been documented for use with Whois++ in RFC-1914. The
same algorithm can be easily used with this version of CIP. In
Whois++ the equivalent of a DSI is called a handle. With this
substitution, the Whois++ mesh traversal algorithm works unchanged
with CIP.

Finally, the mesh entry point (i.e. the first server queried) can
have an impact on the success of the query. To avoid scaling issues,
it is not acceptable to use a single "root" node, and force all
clients to connect to it. Instead, clients should connect to a
reasonably well connected (with respect to the CIP mesh, not the
Internet infrastructure) local server. If no match can be made from
this entry point, the client can expand the search by asking the
original server who polls it. In general, those servers will have a
better "vantage point" on the mesh, and will turn up answers that the
initial search didn't. The mechanism for dynamically determining the
mesh structure like this exists, but is not documented here for
brevity. See RFC-1913 for more information on the POLLED-BY and
POLLED-FOR commands.

It still should be noted that, while these mesh operations are
important to optimizing the searches that a client should make, the
client still speaks its native protocol. This information must be
communicated to the client without causing the client to have to
understand CIP.

5. Security Considerations

   In this section, we discuss the security considerations necessary
   when making use of this specification. There are at least three
   levels at which security considerations come into play. Indexing
   information can leak undesirable amounts of proprietary information,
   unless carefully controlled. At a more fundamental level, the CIP
   protocol itself requires external security services to operate in a
   safe manner. Lastly, CIP itself can be used to propogate false
   information.

5.1 Secure Indexing

   CIP is designed to index all kinds of data. Some of this data might
   be considered valuable, proprietary, or even highly sensitive by the
   data maintainer. Take, for example, a human resources database.
   Certain bits of data, in moderation, can be very helpful for a
   company to make public. However, the database in its entirety is a
   very valuable asset, which the company must protect. Much experience
   has been gained in the directory service community over the years as
   to how best to walk this fine line between completely revealing the
   database and making useful pieces of it available. There are also
   legal considerations regarding what data can be collected and shared.

   Another example where security becomes a problem is for a data
   publisher who'd like to participate in a CIP mesh. The data that
   publisher creates and manages is the prime asset of the company.
   There is a financial incentive to participate in a CIP mesh, since
   exporting indices of the data will make it more likely that people
   will search your database. (Making profit off of the search activity
   is left as an exercise to the entrepreneur.) Once again, the index
   must be designed carefully to protect the database while providing a
   useful synopsis of the data.

   One of the basic premises of CIP is that data providers will be
   willing to provide indices of their data to peer indexing servers.
   Unless they are carefully constructed, these indices could constitute
   a threat to the security of the database. Thus, security of the data
   must be a prime consideration when developing a new index object
   type. The risk of reverse engineering a database based only on the
   index exported from it must be kept to a level consistent with the
   value of the data and the need for fine-grained indexing.

   Lastly, mesh organizers should be aware that the insertion of false
   data into a mesh can be used as part of an attack. Depending on the
   type of mesh and aggregation algorithms, an index can selectivly
   prune parts of a mesh. Also, since CIP is used to discover

   information, it will be the target for the advertisement of false
   information. CIP does not provide a method for trusting the data that
   it contains.

Acknowledgments

   Thanks to the many helpful members of the FIND working group for
   discussions leading to this specification.

   Specific acknowledgment is given to Jeff Allen formerly of Bunyip
   Information Systems. His original version of these documents helped
   enormously in crystallizing the debate and consensus. Most of the
   actual text in this document was originally authored by Jeff. Jeff
   is no longer involved with the FIND Working Group or with editing
   this document. His authorship is preserved by a specific decision of
   the current editor.

Authors' Addresses

   Jeff R. Allen
   246 Hawthorne St.
   Palo Alto, CA 94301

   EMail: jeff.allen@acm.org

   Michael Mealling
   Network Solutions, Inc.
   505 Huntmar Park Drive
   Herndon, VA 22070

   Phone: (703) 742-0400
   EMail: michael.mealling@RWhois.net

References

   [RFC1913]   Weider, C., Fullton, J. and S. Spero, "Architecture
                     of the Whois++Index Service", RFC 1913, February
                     1996.

   [RFC1914]   Faltstrom, P., Schoultz, R. and C. Weider, "How to
                     Interact with a Whois++ Mesh", RFC 1914, February
                     1996.

   [CIP-MIME]   Allen, J. and M. Mealling, "MIME Object Definitions
                     for the Common Indexing Protocol (CIP)", RFC 2652,
                     August 1999.

   [CIP-TRANSPORT] Allen, J. and P. Leach, "CIP Transport Protocols",
                     RFC 2653, August 1999.

Appendix A: Glossary

   application domain: A problem domain to which CIP is applied which
      has indexing requirements which are not subsumed by any existing
      problem domain. Separate application domains require separate
      index object specifications, and potentially separate CIP meshes.
      See index object specification.

   centroid: An index object type used with Whois++. In CIP versions
      before version 3, the index was not extensible, and could only
      take the form of a centroid. A centroid is a list of (template
      name, attribute name, token) tuples with duplicate removed.

   dataset: A collection of data (real or virtual) over which an index
      is created. When a CIP server aggregates two or more indices, the
      resultant index represents the index from a "virtual dataset",
      spanning the previous two datasets.

   Dataset Identifier: An identifier chosen from any part of the
      ISO/CCITT OID space which uniquely identifies a given dataset
      among all datasets indexed by CIP.

   DSI: See Dataset Identifier.

   DSI-description: A human readable string optionally carried along
      with DSI's to make them more user-friendly. See dataset
      Identifier.

   index: A summary or compressed form of a body of data. Examples
      include a unique list of words, a codified full text analysis, a
      set of keywords, etc.

   index object: The embodiment of the indices passed by CIP. An index
      object consists of some control attributes and an opaque payload.

   index object specification: A document describing an index object
      type for use with the CIP system described in this document. See
      index object and payload.

   index pushing: The act of presenting, unsolicited, an index to a
      peer CIP server.

   MIME: see Multipurpose Internet Mail Extensions

   Multipurpose Internet Mail Extensions: A set of rules for encoding
      Internet Mail messages that gives them richer structure. CIP uses
      MIME rules to simplify object encoding issues. MIME is specified
      in RFC-1521 and RFC-1522.

   payload: The application domain specific indexing information stored
      inside an index object. The format of the payload is specified
      externally to this document, and depends on the type of the
      containing index object.

   polled server: A CIP server which receives a request to generate and
      pass an index to a peer server.

   polling server: A CIP server which generates a request to a peer
      server for its index.

   referral chain: The set of referrals generated by the process of
      routing a query. See query routing.

   query routing: Based on reference to indexing information,
      redirecting and replicating queries through a distributed database
      system towards the servers holding the actual results.

6. Full Copyright Statement

Acknowledgement