Network Working Group
Internet Draft
December 1993
Expiration Date: June 1994

C. Huitema/INRIA
P-A. Pays/INRIA
A. Zahm/TS-E3X
A. Woermann/TS-E3X

Simple Object Look-up protocol (SOLO)
<draft-huitema-solo-00.txt>

Status of this Memo

Abstract

We present here a simple access protocol to a "white page" service. This protocol is based on the experience acquired in several X.500 based "white page" pilots, and builds upon such developments as the "user friendly naming" notation or the "centroids" concept developped in the "whois++" project.

Although largely inspired by the X.500 protocol, SOLO does not try follow the X.500 model. Designing a SOLO front-end to an X.500 server is very easy, but the SOLO internetworking is not based on X.500 -- the navigation part is based on the WHOIS++ centroids. In fact, one can as well design a SOLO front-end to a WHOIS server, and only minor efforts are needed for redesigning a "finger" daemon to provide the SOLO service. Conversely, the SOLO service is powerful enough to make gatewaying between X.500 access protocols and SOLO easy.

The first section of this memo present the basic SOLO protocol. The second section presents the methods which are used for "networking" SOLO servers. The third section is a formal definition of the SOLO protocol. Section 4 list the various SOLO information messages. Section 5 addresses the security implications of providing a SOLO service.

1. **Presentation of the SOLO protocol:**

The SOLO protocol consists of two main operations: a simple look-up operation that enables clients to request information on a name object, and a "zone transfer" operation used by index handlers. These operations are carried upon TCP-IP connections; the requests and results use a simple text format. Three additional commands are defined, i.e. a "signal" to notify the need to poll a server, a "reset" to abort an ongoing operation, and a "quit" to close the connection.

1.1. A basic example:

In order to request information on an object, a client will set up a TCP-IP connection towards the server. Servers will be awaiting requests on port <solo>. After establishing the connection, the client will send a request of the form:

```
SOLO <Huitema, Sophia, INRIA, FR> ? CN, O, OU, Phone, Email;
```

In this line, SOLO is the command name (other commands are for example QUIT or POLL). It is followed by the "name" of the object, which is expressed using the "user friendly name" conventions, by a question mark, and by a list of "attribute names", using the acronyms defined for the white page pilot.

If the server was able to find the information, it will send a "matched" reply, followed by the values of the attributes, ending by a dot on a line by itself, e.g.:

```
500 Matches: <CN=Christian Huitema,OU=Sophia,O=INRIA,C=FR>
CN: Christian Huitema
OU: Sophia, Sophia-Antipolis,
    "Unite de recherche de Sophia Antipolis"
O: INRIA, "INSTITUT NATIONAL DE RECHERCHE EN
   INFORMATIQUE ET AUTOMATIQUE"
Phone: +33 93 65 77 77
Email: Christian.Huitema@sophia.inria.fr
.
```

Indeed, several errors could have occured, for example if the name was ambiguous or non existent, or if the server did not hold the information locally.

After receiving the response, the client has the choice to send another command, e.g. to get information on another object, or to close the connection.

1.2. Name errors

SOLO servers distinguish three kinds of name errors, ambiguous specifications, incorrect specifications and over specifications. They reply to these errors by an appropriate error message, optionally followed by one or several "hints", i.e. suggested references. The following dialogue is an example of ambiguous reference:

```
SOLO <Martin, Sophia, INRIA, FR> ? Phone, Email;
201-Ambiguous name: <Martin, Sophia, INRIA, FR>
301-Partial Match: <Sophia, INRIA, FR> <OU=Sophia,O=INRIA,C=FR>
400-Suggestion: <CN= Laure Martin,OU=Sophia,O=INRIA,C=FR>
400 Suggestion: <CN= Michel Martin,OU=Sophia,O=INRIA,C=FR>
```

There were several "Martin" at INRIA, and the server just provide the names of all entries that matched the request. The next dialogue is an example of incorrect reference:

```
SOLO <Huttema, Sophia, INRIA, FR> ? Email;
202-No such name: <Huttema, Sophia, INRIA, FR>
301-Partial Match : <Sophia, INRIA, FR> <OU=Sophia,O=INRIA,C=FR>
400-Suggestion: <CN= Christian Huitema,OU=Sophia,O=INRIA,C=FR>
400 Suggestion: <CN= Bernard Hettena,OU=Sophia,O=INRIA,C=FR>
```

Note that the server applied here a local strategy to derive possible matches -- it tried a "phonetic" match on the name token "Huttema". Also note the presence in both examples of a "partial match" information line: it signals that the upper components of the name assertion, in this case <Sophia, INRIA, FR>, were exactly matched by the server and correspond to the distinguished name <OU=Sophia,O=INRIA,C=FR>.

Note that there could be examples where intermediate components are ambiguous, for example:

```
SOLO <Martin, IN*A, FR>? Email;
201-Ambiguous name: <Martin, IN*A, FR>
400-Suggestion: <Martin,O=INA,C=FR>
400-Suggestion: <Martin,O=INRA,C=FR>
400 Suggestion: <Martin,O=INRIA,C=FR>
```

In that case, the suggestion contains potential candidates for "continuation" queries. The server can use two methods to construct the "suggested name", i.e.:

(1) replace the "upper part" components of the initial query by a "distinguished" value,

(2) provide the "distinguished name" of candidate entries.

In the first case, the suggested name is not necessarily a "distinguished" name, but merely a hint that can be used in a further query.

Over specification occurs when a requestor present a set of naming attributes, some of which cannot be matched:

```
SOLO <Huitema, Sophia, Region=PACA, INRIA, FR> ? Email;
203-Over specified name: <Huitema,Sophia,Region=PACA,INRIA,FR>
301-Partial Match: <INRIA, FR> <O=INRIA,C=FR>
400 Suggestion: <CN= Christian Huitema,OU=Sophia,O=INRIA,C=FR>
```

1.3. Partial name matches and redirections

A SOLO server may not always be able to fully match a query. There will be cases however where the local server can provide some "hint" on the resolution of some of the "name parts". For example, a server that knows that the company E3X has recently changed names may be able to respond to a query by a partial match indication:

```
SOLO <Woermann, OSI, E3X, FR> ? Email;
301-Partial Match: <E3X, FR> <O=TS-E3X,C=FR>
400 Suggestion: <Woermann, OSI,O=TS-E3X,C=FR>
```

In the previous exchange, the SOLO server did not provide an indication of where to submit the next query. It could indeed have provided such information through a "suggested server" message:

```
SOLO <Woermann, OSI, E3X, FR> ? Email;
301-Partial-Match: <E3X, FR> <O=TS-E3X,C=FR>
302-Suggested-server: <O=TS-E3X,C=FR> mitsou.inria.fr
302-Suggested-server: <O=TS-E3X,C=FR> osi.e3x.fr
400 Suggestion: <Woermann, OSI,O=TS-E3X,C=FR>
```

Note that we have suggested here two different servers. The client will presumably chose one based on local information.

1.4. Absolute matches:

There are some cases where the SOLO user wants to completely specify the name of the requested object, without leaving any interpretation possibility to the server. This is specified by a particular format of the SOLO request:

```
SOLO <CN= Christian Huitema,OU=Sophia,O=INRIA,C=FR> ! Email;
```

The differences between this format and a standard request are the following:

(1) The name should be fully specified. The separator between the name and the list of attributes is an exclamation mark (!) instead of a question mark (?).

The SOLO server can only respond positively to this query if it finds an entry that exactly matches the purported name.

1.5. Attributes and Universal Resource Locators (URL)

It is often impractical for a server to provide a complete attribute value: some objects like photographies or voice segments are too bulky to be transmitted easily. Others may not be available on the SOLO server itself. In these cases, the SOLO server may elect to return a pointer to the value, in the form of an "Universal Resource Locator", instead of the value itself. The presence of an URL in the value field is signalled by a placing an hyphen in front of the attribute type, as in:

```
SOLO <Huitema, Sophia, INRIA, FR> ? Photo, Email;
500 Matches: <CN=Christian Huitema,OU=Sophia,O=INRIA,C=FR>
-Photo: "http://zenon.inria.fr:2220/peoples/huitema/photo"
Email: Christian.Huitema@sophia.inria.fr
.
```

In fact, some clients may want to force this facility, in order to be able to locate an attribute without having to actually receive the data. They can achieve this by prepending an hyphen in front of the attribute type in their request, as in:

```
SOLO <Huitema, INRIA, FR> ? -Email, -Photo;
500 Matches: <CN=Christian Huitema,O=INRIA,C=FR>
-Photo: "httl://zenon.inria.fr:2220/peoples/huitema/photo"
-Email: "solo://mitsou.inria.fr:8888/<Huitema,INRIA,FR>!Email"
.
```

Note the special form of universal resource locator used for the "Email" attribute: this is the normal way to designate a resource reachable through within a SOLO server.

Clients cannot force SOLO servers to return a value instead of the URL: this could be outside of the server's capabilities. However, servers should always return the full attribute value when this is available.

1.6. Aliases and alternate values:

Some SOLO servers will be implemented as access points to an X.500 service. It is important that the SOLO clients recognize some oddities which can result from the X.500 support of aliases and alternate values, e.g.:

```
SOLO <S=Bolot+First=Jean,O=INRIA,C=FR>?Email;
500 Name matches: <CN="Jean-Chrysostome Bolot",O=INRIA,C=FR>
Email: "bolot@mitsou.inria.fr"
SOLO <CN="Suzan Mendes",OU=Sophia,O=INRIA,C=FR>?Email;
500 Name matches: <CN="Suzan Mendes",OU=OSI,O=TS-E3X,C=FR>
Email: "s.mendes@osi.e3x.fr"
.
```

In the first case, the first name specification "Jean" was matched by one of the multiple values declared for the first name attribute, although the common name refers to the dis-tinguished value "Jean-Chrysostome". In the second case, the entry for "Suzan Mendes" was an alias, pointing to her real entry in the X.500 server of TS-E3X.

In fact, X.500 is probably not the only name service that provides aliasing or alternate names. We are likely to observe the same "oddities" with many data bases.

1.7. Filters and substrings:

There are cases where the clients will not be able to completely specify the name of the object they are searching. The SOLO protocol allow these clients to enter "loose" specifications by means of substrings or filter.

Any name part can be replaced by a substring specification, using the special character "*", as in for example:

```
SOLO <chr*Hu*ma*, INR*, FR> ? Email;
500 Matches: <CN=Christian Huitema,O=INRIA,C=FR>
Email: huitema@sophia.inria.fr
.
```

As implied by well established conventions, the "*" can be matched by any sequence of characters.

The "User Friendly Name" syntax allows users to combine several "attribute assertions" in a name part, using the "+" character as a delimiter, e.g.:

```
SOLO <First=Christian + S=Huitema, INRIA, FR> ? Email;
```

The "+" character is in fact equivalent to an "AND" operator. The SOLO syntax extends this possibility by also allowing the character "|" as an OR operator, e.g.:

```
SOLO <CN=John Smith | CN=Jack Smith, INRIA, FR> ? Email;
```

Which can be matched by either value. Indeed, the "+" and "|" can be combined, in which case "natural" precedence rules apply, i.e. the AND operation takes precedence over the OR. For example, the previous specification is equivalent to:

```
SOLO <First=John+S=Smith | First=Jack+S=Smith,INRIA,FR>?Email;
```

Experts will note that this notation is more restrictive than the full set of boolean combinations allowed by X.500 filters. This restriction is based on the desire to keep SOLO simple, and also supported by the substantial experience acquired by the authors when deploying X.500 systems. We believe that the proposed compromize is just right.

Experts will also remember that the "User Friendly Name" specification imposes a restriction on the use of the "+" delimiter: when multiple assertions are stated in a name part, then they should be explicitely typed. This restriction does not apply in SOLO. The following command:

```
SOLO <John + Smith | Jack + Smith, INRIA, FR> ? Email;
```

is explicitely allowed.

1.8. Limiting the size of the output:

The filters, the substring notation, or even the use of partial matches, may enable the user to specify extremely fuzzy names. Such fuzzy names can potentially be matched by a very large number of entries.

Sending very long lists of "suggested" names is inadequate for several reasons:

- it may be a very inefficient use of network resources,
- it may impose unexpected constraints to the user interface which is not necessarily rigged to handle long lists,
- it may also be a privacy violation, as it would allow intruders to easily obtain complete listings of an

organization's personal.

SOLO servers are allowed to limit the number of names which can be returned by a single command. A limited set of result will be signaled by a specific error report:

```
SOLO <Martin*, INRIA, FR> ? Email;
201-Ambiguous name: <Martin*, INRIA, FR>
400-Suggestion: <CN= Laure Martin,OU=Sophia,O=INRIA,C=FR>
400-Suggestion: <CN= Michel Martin,OU=Sophia,O=INRIA,C=FR>
204 Too many names to list them all.
```

In fact, the privacy argument is so compelling that SOLO servers are encouraged to impose a relatively low limit, e.g. not larger than 8. The precise value of this limit should be set by the local administrator, as well as a policy choice to display a random set of suggested names first and then the error message, or to not display any name at all if the command is too fuzzy.

2. Networking white page informations:

The SOLO server may want to maintain "indexing" information. This is done according to the "centroid" model. The informations held by a given server are summarized in a centroid, i.e. a gigantic entry that contain for all indexed attribute types the list of attribute values found in any of the data base entries.

2.1. Using the index request:

In order to obtain the indexing information belonging to a server, a client or another server may set up a TCP connection to the server using the <solo> port, then send the index command:

```
POLL S, OU, O, L, C;
```

Where POLL is the command, <> an empty name specification, and where the rest of the command just lists the attributes for which indexing information is required. A typical response will be:

```
501 Sending indexes.
S: Name1, Name2, Name3, Name4,
   name5
OU: Unit1, Unit2
O: The local org name
C: XX
.
```

Errors can indeed occur, e.g. if the server is unwilling to send the requested information.

A special form of the command allows the clients to obtain the list of attributes managed by the server, i.e. by specifying an empty list of attributes.

```
POLL ;
502 Providing attribute list
C, L, O, OU, S, G
.
```

Indexing servers will keep this information and organize it in tables in order to manage navigation.

In some cases, the server may be unwilling to provide a full listing: for example, providing the names of all the employees of a company may be perceived as a serious security violation. In this case, the server may elect to replace the attribute values by a simple "star":

```
POLL S, OU, O, L, C;
501 Sending indexes.
S: *
OU: Unit1, Unit2
O: The local org name
C: XX
.
```

This procedure may also be used when listing all values would be unpractical.

2.2. Signalling secondary servers

In order to set up or maintain the network of SOLO servers, a POLL command is not sufficient: the index servers have no way to guess that a new server has becomed operational, or that a remote data base has been updated.

The SGNL command allows remote servers to "signal" their existence. More precisely, it allows administrators to notify the presence of remote servers, as in:

```
SGNL mitsou.inria.fr
403 Will poll: mitsou.inria.fr
```

Where an administrator signalled the need to poll the server "mitsou.inria.fr", and where the index server responded that he scheduled the emission of a POLL command. There can indeed be cases where the server will not schedule this emission, such as:

```
SGNL matso.inrja.fr
111 No such host: matso.inrja.fr

SGNL felix.inria.fr
112 No SOLO server on host: matso.inrja.fr

SGNL tom.inria.fr
113 Unwilling to poll: matso.inrja.fr
```

Note that this simple "announce message" was prefered to a more radical transaction where the remote server will take the initiative to download its indexing data in the local server for the following reasons:

- o we want to allow administrators to use a client interface to signal the need to poll.

- o polling the servers and fetching the data is simpler to organize, as the data are naturally available.

- o pushing the data to an unprepared index server would be distateful, as the recipient is not necessarily ready to receive.

A similar problem was analyzed in the WHOIS++ effort.

2.3. Relaying of queries

SOLO servers which cannot fully solve a query have two choices. They may simply send back the "partial matches" and/or "suggested servers" informations described in section 1. But they may also decide to immediately relay the query towards one or several of the suggested servers.

When doing so, the server will forward the query using the SOLO protocol, then send the result back to the user.

The server which takes the decision to relay a query may chose one of the possible servers, or try several of them and construct a response by merging the various results. Whether one or several server are tested, whether they are tested in parallel or in sequence, and how the answer is constructed is fully a local decision: we do not believe that we have acquired enough experience to write down a mandatory recommendation. However, it is obvious that some indication of the decision, and in particular some indication of the source of the data, should be given back to the user. This is done through a specific information message, indicating the decision to relay, as in the following example:

```
SOLO <Woermann, OSI, E3X, FR> ? Email;
301-Partial-Match: <E3X, FR> <O=TS-E3X,C=FR>
302-Suggested-server: <O=TS-E3X,C=FR> mitsou.inria.fr
302-Suggested-server: <O=TS-E3X,C=FR> osi.e3x.fr
402-Relaying: <Woermann, OSI,O=TS-E3X,C=FR> mitsou.inria.fr
500 Matches: <CN=Ascan Woermann,OU=OSI,O=TS-E3X,C=FR>
Email: woermann@osi.e3x.fr
.
```

The "relaying" information shows the request that was deduced from the original query, as well as the name of the server to which this request was relayed. It is followed by the messages received from the this server.

As relayed commands can be further relayed, it is important to include some protection against loops in the protocol. This is achieved by inserting an optional "count of relays" between the SOLO request code and the request itself, as in:

```
SOLO 1 <Woermann, OSI,O=TS-E3X,C=FR> ? Email;
```

SOLO servers should increment the count of relay when relaying a request. They should never relay a request where the count as already reached a value equal to 8, or larger. An absent count is equivalent to a count of zero.

2.4. Use of caches:

The experience has showed that name queries tend to be repetitive, and that a "working set" effect can be efficiently exploited. It is thus highly desirable that servers exploit this working set effect by keeping a cache of the "most frequent" requests.

There are three types of information that can be cached by the SOLO: the full or partial name matches, the binding between names and servers, and the attribute values.

Full matches are obtained through the "Matches" information lines (code 500). Partial matches are obtained by the similarly named information lines (code 301). In both cases, the cache entry will associate a name request to the matched value.

Server suggestions are obtained through the "Suggested-server" lines (code 302). The cache entry will associate a name to one or several suggested servers.

Attribute values are obtained as a result of successfull queries.

Servers should be careful to time out cache entries after a reasonable duration, so as to avoid keeping aged information alive. It is suggested to avoid keeping any cache entry more than a day.

2.5. Networking with other technologies:

In some cases, SOLO server will be able to determine that the requested object cannot be accessed through the SOLO service, but could be obtained using another technology, e.g. whois, finger or X.500. They can pass this information through the "suggested service" command, e.g.:

```
SOLO <Baker, CS, UCL, GB> ? Phone;
301-Partial match: <CS, UCL, GB> <OU=CS;O=UCL;C=GB>
401-S-service: x500://cs.ucl.ac.uk:107/<Baker,OU=CS;O=UCL;C=GB>
401 S-service: finger://cs.ucl.ac.uk:79/Baker
```

The end of the code 401 line carries an "universal resource locator". How this information is acquired is outside the scope of this memo.

2.6. Stacking multiple requests:

The SOLO protocol follows a very simple client/server model. Clients ask questions, and servers reply by a list of informations. In some cases, clients may want to send several requests in one session. They normally do so by sending one question at a time, waiting for an answer, then proceeding to the next question.

Clients may however send further commands on the TCP connection without waiting for the answer to the previous one.

3. Protocol specification:

3.1. The format of a request:

All requests of the SOLO protocol start by a four letter request code, e.g. SOLO, POLL, SGNL, RSET, QUIT. The codes can be expressed in upper or lower cases, or a combination of both.

In the case of the SOLO request, the command includes:

(1) a variable number of spacing characters (spaces or tabs).

(2) an optional "count of relays" indicator, encoded using one decimal digit, followed if present by a variable number of spacing characters.

(3) the name of the requested object, enclosed within angle brackets.

(4) a variable number of spacing characters (spaces or tabs).

(5) a precision specifier, which can be either a question mark (?) or an exclamation mark (!).

(6) a list of attributes, terminated by a semicolon (;).

In the case of the POLL request, the command includes:

(1) a variable number of spacing characters (spaces or tabs).

(2) a list of attributes, terminated by a semicolon (;). lp In the case of the SGNL request, the command includes:

(3) a variable number of spacing characters (spaces or tabs).

(4) the domain name of the server that should be polled.

The RSET and QUIT request don't contain any information after the command code.

All commands are terminated by an "end of line" mark, composed of the two characters "carriage return" and "line feed".

3.2. The format of the replies:

The replies sent by the SOLO servers are always composed of one of several information lines, followed in some cases by a data line.

The information lines are always composed of:

(1) A three digit information code,

(2) A continuation character, i.e. a space character if this is the last information line or an hyphen otherwise,

(3) A free form text composed of letters, white spaces and digits.

If the line does not contain additional information, the free form text is terminated by a dot(.). If it contains additional information, the free text is followed by a semi-colon (:) and the information. This information is composed of a name specification enclosed in angle brackets, optionally followed by another name enclosed in angle brackets, for partial match reports, or the domain name of a suggested server. In one case (code 401), the information is composed of an "universal resource locator"; in a few other (111, 112, 113, 403), the information consists solely of a "domain name".

All information lines are terminated by an "end of line" mark, composed of the two characters "carriage return" and "line feed".

When the first digit of the last information line has the value "5", this line is followed by a list of attribute values.

3.3. Format of name specifications:

Name specifications occur in the SOLO request and in several error messages. Names are formatted according to the rules specified in RFC-1485 (string representation of distinguished names)...

3.4. Format of attribute lists:

Attribute lists are present in the SOLO and POLL requests. Attribute lists are composed of attribute names separated by commas (,). An arbitrary number of spacing characters (space, tab) can be inserted around the attribute names. The attribute list is always terminated by a semi-colon. An attribute list may be empty.

The attribute type can be preceded by an hyphen to indicate, within the POLL command, that pointers (URL) are requested instead of values.

3.5. List of attribute values:

A list of attribute values is composed of several attribute values. Each attribute value is encoded of one or several lines of texts; all lines of text are terminated by a carriage return and a line feed. The list is terminated by a line containing a single dot character. The list may be empty, i.e. it may be the case that the dot line is the only line in it.

The first line of an attribute value starts by the encoding of the attribute type, followed by a colon. All other lines should start by one or several spacing characters (space or tab). Each line contains one or several values, encoded according to the rules specified in RFC-1488 (String representation of standard attribute syntaxes), and separated from the following value by a comma and an arbitrary number of linear spaces.

Within the values returned by the POLL command, the attribute type can be optionally preceded by an hyphen. In this case, the values returned are universal resource locators, instead of actual values.

3.6. Attribute names:

Attribute names are encountered in name specifications, in attribute lists and in attribute values.

The attribute names can be:

- o simple keywords, as defined in RFC-1488 (String representation of standard attribute syntaxes) or in the Whois++ lookup service.

- o text renditions of object identifiers.

Official keywords should be prefered whenever possible. The following table list a set of key words which are expected to be recognized by SOLO servers, as well as their WHOIS and "User Friendly Name" equivalents:

Whois (RFC-1274)	RFC-1485	SOLO
Name (CommonName)	CN	CN
(Surname)	S	S
(First name)		First
(Country)	C	C
(StateOrProvinceName)	ST	ST
(LocalityName)	L	L
Organization	O	O
Department	OU	O
Title		Title
Work-telephone		Phone
Fax-telephone		Fax
Work-address		Address
Email-address (RFC822)		Email

The "First" keyword is derived from the "first" command line argument to "whois"; the "Phone", "Fax" and "Email" simplications come from the wide usage of these arguments and the desire to focus on "business" attribute by default (see the "security" section for a rationale). The corresponding "object identifiers" can be found in RFC-1274 (X.500 directory schema).

3.7. Values:

Values are encoutered in name specifications and in value lists.

There are three possible encodings for a value:

- o a set of printable characters, excluding any occurence of the special characters comma(,), colon(:), equal(=), semi-colon(;) question mark (?) and of the angle brackets (< and >). In this syntax, multiple occurence of white spaces can be replaced by a single white space.

- o a quoted string, optionally preceded by an alphabet notation, and delimited by double quotes. In this case, the MIME "quoted text" format is used.

- o a binary string, encoded as defined in the "base64" notation, and delimited by simple quotes.

Only the first two formats can be used within the command line. When the value is an universal resource locator, use of the quoted text notation is encouraged.

4. A list of information codes:

Information codes fall within 5 categories:

(1) System errors are numbered from 100 to 199.

(2) Name errors are numbered from 200 to 299.

(3) Intermediate informations are numbered from 300 to 399.

(4) User informations are numbered from 400 to 499.

(5) Completion reports are numbered from 500 to 599.

4.1. System errors

System errors indicate that the query could not be further processed. The following errors have been defined:

100	Unrecognized command	
101	Incorrect name specification	
102	Incorrect attribute list	
103	Incorrect command parameters	
104	Momentary congestion, try later	
105	Server problem	
106	Data base problem	
107	Timing out after relaying	
108	Abandoning the request	
110	Too many relays, probably looping	
111	No such host	domain-name
112	No SOLO server on host	domain-name
113	Unwilling to poll	domain-name

4.2. Name errors

Name errors occurs when the name specification is invalid. The following errors have been defined:

201	Ambiguous name	<name>
202	No such name	<name>
203	Over specified name	<name>
204	Too many names to list them all	

4.3. Intermediate informations

Intermediate informations are informations that can be "cached" by the server. The following codes are defined:

301	Partial Match	<name> <name>
302	Suggested-server	<name> domain-name

4.4. User informations

User informations are suggestions that can be passed to the user. The following codes have been defined:

400	Suggestion	<name>
401	S-service	URL
402	Relaying	<name> domain-name
403	Will poll	domain-name

4.5. Completion reports

Completion reports precede a list of attribute values. Two reports are defined:

500	Matches	<name>
501	Sending indexes	

5. **Security considerations:**

The SOLO protocol is designed to widely publicize information about people, e.g. telephone numbers or electronic mail addresses. As such, the most important considerations relate to the protection of personal privacy: notification and filtering, listing and browsing. Then, one finds the classical problem of "name server" attacks. The attention of the administrators is also drawn onto the particular problem of embedding login information into mail addresses.

5.1. Notification and removal

In several countries, the privacy protection laws require that users be notified of their listing in any data base containing personal information; in some countries, one has to sollicit their explicit agreement before listing their informations. The matter is only made worse by the international nature of the Internet: information placed in a connected data base should be considered as publicized on a world wide basis.

Even when the local laws do not require it, it is good practice to observe the following guidelines:

o Notify people listed in the SOLO server that you are publishing their personal information,

o Be ready to remove some people from their data bases. There are a number of people who may have good reasons not to be listed at all.

- o Don't publish more data than necessary. We clearly need to list identification data such as first names or surnames, and there are very good reasons for listing an email address, a fax number or an office phone number. But one should clearly think twice before listing a home address or the list of ongoing projects...

Local regulations may impose stronger protections. These regulations should indeed be followed.

5.2. Listing and browsing

Practicians of the "privacy laws" are adamant against the risks posed by "remote listing" procedures. When available without restriction, services such as the X.500 "LIST" operation allow complete foreigners to copy a data base and use it for "innovative" purposes, e.g. correlate it with another base, or use it as the start of a "mailing" operation.

We have seen in the section on "limiting the size of the output" the need to limit the number of names returned by the "SOLO" command. A similar risk is posed by the "POLL" command, as:
POLL CN;

would allow to list the names of all the person present in the data base. In the absence of secure communications, if the remote party cannot be precisely identified, administrators of the SOLO service would be well inspired to limit the set of attributes returned by the POLL command to only those that identify the organization, e.g. Organization name, Unit name, Country, Region and City. The "CN=*" facility should be used instead of returning a full listing of names.

5.3. The name server attacks

The SOLO service will be used to store communication data, e.g. phone numbers or electronic mail addresses. There is a classic attack against this service, in which an intruder "emulates" a server and return forged addresses, so that he or an accumplice receives the information that was meant to be sent to a third party.

X.500 protects clients against this attack by "strongly authenticating" request and responses, so that there is no way to emulate a server or to alter the information. SOLO does not include strong authentication. Two alternate methods could be used:

(1) Run SOLO on top of a secure network, e.g. using IP security.

(2) Protect the data themselves, e.g. have them signed and scealed, as e.g. PEM certificates.

One should note that the second alternative is the only one that maintain the protection even when caches are used.

5.4. Logins and addresses

Email addresses are often build as a combination of a host name and a login name or account number. It is well known that exposing these combinations facilitates the task of intruders -- they only have to "guess a password" instead of having to guess both an account number and a password. This problem is however not specific to SOLO. Any other white page service or any message sent to a wide distribution list would also publicize the information.

6. **Acknowledgments:**

We would like to thank the members of the IETF "directory service" working groups and the participant to OPAX, PARADISE and various other white page pilots for their encouragements. The use of URLs was suggested by Marshall Rose in a discussion on the "future of X.500", and was also discussed in the RARE "WG-NAP" working group. The members of the RARE WG-NAP initiated the analysis of the privacy requirements which is reported in the "security" section. Several members of the INRIA and TS-E3X teams reviewed the draft. Also, we would like to pay tribute to the member of the WNILS working group, as we shamelessly copied the "centroid" concept which was developped for the WHOIS++ service.

7. References:

(1) Gargano, Joan, Weiss, Ken. Whois and Network Information Lookup Service, Whois++. RFC in preparation.

(2) Weider, Chris, Fulton, Jim, Spero, Simon. Architecture of the Whois++ Index Service. RFC in preparation.

(3) Howes, T. , Kille, S., Yeong, W., Robbins, C. The X.500 String Representation of Standard Attribute Syntaxes. RFC-1488.

(4) Kille, Steve, Using the OSI Directory to achieve User Friendly Naming. RFC-1484. July 1993.

(5) Kille, Steve, A String Representation of Distinguished Names. RFC-1485. July 1993.

(6) CCITT. X.500 series of recommendations. 1988.

(7) Barker, Paul, Kille, Steve, The COSINE and Internet X.500 Schema. RFC-1274. November 1991.

8. Authors' Addresses

Comments should be sent to:

Christian Huitema
INRIA
2004 Route des Lucioles
BP 93
06902 Sophia-Antipolis Cedex
France

Email: Christian.Huitema@sophia.inria.fr

Paul-Andre Pays
INRIA
BP105
78153 Le Chesnay Cedex
France

Email: Paul-Andre.Pays@faugeres.inria.fr

Alain Zahm, Ascan Woermann
TS-E3X Groupe France Telecom,
Les Algorithmes,
Route des Lucioles,
Bat. Pythagore A
06560 Valbonne France

Email: zahm@osi.e3x.fr, Woermann@osi.e3x.fr

Table of Contents