Internet Engineering Task Force
INTERNET DRAFT
13 **March 1997**

John Veizades
@Home Network
Erik Guttman
Charles Perkins
Sun Microsystems
Scott Kaplan

Service Location Protocol
draft-ietf-svrloc-protocol-16.txt

Status of This Memo

Abstract

The Service Location Protocol provides a scalable framework for the discovery and selection of network services. Using this protocol, computers using the Internet no longer need so much static configuration of network services for network based applications. This is especially important as computers become more portable, and users less tolerant or able to fulfill the demands of network system administration.

Contents

Status of This Memo i

Abstract i

1. Introduction 1

2. Terminology 1
   2.1. Notation Conventions 3
   2.2. Service Information and Predicate Representation 3
   2.3. Specification Language 4

3. Protocol Overview 4
   3.1. Protocol Transactions 5
   3.2. Schemes 7
      3.2.1. The ``service:'' URL scheme 7
   3.3. Standard Attribute Definitions 7
   3.4. Naming Authority 8
   3.5. Interpretation of Service Location Replies 8
   3.6. Use of TCP, UDP and Multicast in Service Location 9
      3.6.1. Multicast vs. Broadcast 9
      3.6.2. Service-Specific Multicast Address 10
   3.7. Service Location Scaling, and Multicast Operating Modes . 11

4. Service Location General Message Format 12
   4.1. Use of Transaction IDs (XIDs) 14
   4.2. URL Entries 14
   4.3. Authentication Blocks 15
   4.4. URL Entry Lifetime 18

5. Service Request Message Format 18
   5.1. Service Request Usage 20
   5.2. Directory Agent Discovery Request 21
   5.3. Explanation of Terms of Predicate Grammar 22
   5.4. Service Request Predicate Grammar 24
   5.5. String Matching for Requests 26

6. Service Reply Message Format 27

7. Service Type Request Message Format 28

8. Service Type Reply Message Format 30

9. Service Registration Message Format 31

10. Service Acknowledgement Message Format 34

11. Service Deregister Message Format 36

12. Attribute Request Message Format 37

13. Attribute Reply Message Format 39

14. Directory Agent Advertisement Message Format 41

15. Directory Agents 42
- 15.1. Introduction 42
- 15.2. Finding Directory Agents 42

16. Scope Discovery and Use 44
- 16.1. Protected Scopes 45

17. Language and Character Encoding Issues 46
- 17.1. Character Encoding and String Issues 47
  - 17.1.1. Substitution of Character Escape Sequences . . . 48
- 17.2. Language-Independent Strings 48

18. Service Location Transactions 49
- 18.1. Service Location Connections 49
- 18.2. No Synchronous Assumption 50
- 18.3. Idempotency 50

19. Security Considerations 50

20. String Formats used with Service Location Messages 52
- 20.1. Previous Responders' Address Specification 52
- 20.2. Formal Definition of the ``service:'' Scheme 52
  - 20.2.1. Service Type String 53
- 20.3. Attribute Information 53
- 20.4. Address Specification in Service Location 54
- 20.5. Attribute Value encoding rules 55

21. Protocol Requirements 55
- 21.1. User Agent Requirements 56
- 21.2. Service Agent Requirements 57
- 21.3. Directory Agent Requirements 59

22. Configurable Parameters and Default Values 60
- 22.1. Service Agent: Use Predefined Directory Agent(s) 62
- 22.2. Time Out Intervals 62

23. Non-configurable Parameters 63

24. **Acknowledgments** 63

A. Appendix: Technical contents of ISO 639:1988 (E/F): "Code for the representation of names of languages" 65

B. SLP Certificates 66

C. Example of deploying SLP security using MD5 and RSA 68

D. Example of use of SLP Certificates by mobile nodes 68

E. Appendix: For Further Reading 69

1. Introduction

Traditionally, users find services by using the name of a network host (a human readable text string) which is an alias for a network address. The Service Location Protocol eliminates the need for a user to know the name of a network host supporting a service. Rather, the user names the service and supplies a set of attributes which describe the service. The Service Location Protocol allows the user to bind this description to the network address of the service.

Service Location provides a dynamic configuration mechanism for applications in local area networks. It is not a global resolution system for the entire Internet; rather it is intended to serve enterprise networks with shared services. Applications are modeled as clients that need to find servers attached to the enterprise network at a possibly distant location. For cases where there are many different clients and/or services available, the protocol is adapted to make use of nearby Directory Agents that offer a centralized repository for advertised services.

2. Terminology

User Agent (UA)
: A process working on the user's behalf to acquire service attributes and configuration. The User Agent retrieves service information from the Service Agents or Directory Agents.

Service Agent (SA)
: A process working on the behalf of one or more services to advertise service attributes and configuration.

Service Information
A collection of attributes and configuration information associated with a single service. The Service Agents advertise service information for a collection of service instances.

Service
The service is a process or system providing a facility to the network. The service itself is accessed using a communication mechanism external to the the Service Location Protocol.

Directory Agent (DA)
A process which collects information from Service Agents to provide a single repository of service information in order to centralize it for efficient access by User Agents. There can only be one DA present per given host.

Service Type
Each type of service has a unique Service Type string. The Service Type defines a template, called a "service scheme", including expected attributes, values and protocol behavior.

Naming Authority
The agency or group which catalogues given Service Types and Attributes. The default Naming Authority is IANA, the Internet Assigned Numbers Authority.

Keyword
A string describing a characteristic of a service.

Attribute
A (class, value-list) pair of strings describing a characteristic of a service. The value string may be interpreted as a boolean, integer or opaque value if it takes specific forms (see section 20.5).

Predicate
A boolean expression of attributes, relations and logical operators. The predicate is used to find services which satisfy particular requirements. See section 5.3.

Alphanumeric
A character within the range 'a' to 'z', 'A' to 'Z', or '0' to '9'.

```
      Scope     A collection of services that make up a logical group.
                See sections 3.7 and 16.

      Site Network
                All the hosts accessible within the Agent's multicast
                radius, which defaults to a value appropriate for
                reaching all hosts within a site (see section 22).  If
                the site does not support multicast, the agent's site
                network is restricted to a single subnet.

      URL       A Universal Resource Locator - see [6].

      Address Specification
                This is the network layer protocol dependent mechanism
                for specifying an Agent.  For Internet systems this is
                part of a URL.
```

2.1. Notation Conventions

```
      CAPS   Strings which appear in all capital letters are protocol
             literal.  All string comparison is case insensitive,
             however, (see section 5.5).  Some strings are quoted in
             this document to indicate they should be used literally.
             Single characters inside apostrophes are included
             literally.

      <>     Values set off in this manner are fully described in
             section 20.  In general, all definitions of items in
             messages are described in section 20 or immediately
             following their first use.

      |  |
      \  \   Message layouts with this notation indicate a variable
      |  |   length field.
```

2.2. Service Information and Predicate Representation

Service information is represented in a text format. The goal is that the format be human readable and transmissible via email. The location of network services is encoded as a Universal Resource Locator (URL) which is human readable. Only the datagram headers are encoded in a form which is not human readable. Strings used in the Service Location Protocol are NOT null-terminated.

Predicates are expressed in a simple boolean notation using keywords, attributes, and logical connectives, as described in Section 5.4.

The logical connectives and subexpressions are presented in prefix-order, so that the connective comes first and the expressions it operates on follow afterwards.

2.3. Specification Language

In this document, several words are used to signify the requirements of the specification [8]. These words are often capitalized.

MUST	This word, or the adjective "required", means that the definition is an absolute requirement of the specification.
MUST NOT	This phrase means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "recommended", means that, in some circumstances, valid reasons may exist to ignore this item, but the full implications must be understood and carefully weighed before choosing a different course. Unexpected results may result otherwise.
MAY	This word, or the adjective "optional", means that this item is one of an allowed set of alternatives. An implementation which does not include this option MUST be prepared to interoperate with another implementation which does include the option.
silently discard	The implementation discards the datagram without further processing, and without indicating an error to the sender. The implementation SHOULD provide the capability of logging the error, including the contents of the discarded datagram, and SHOULD record the event in a statistics counter.

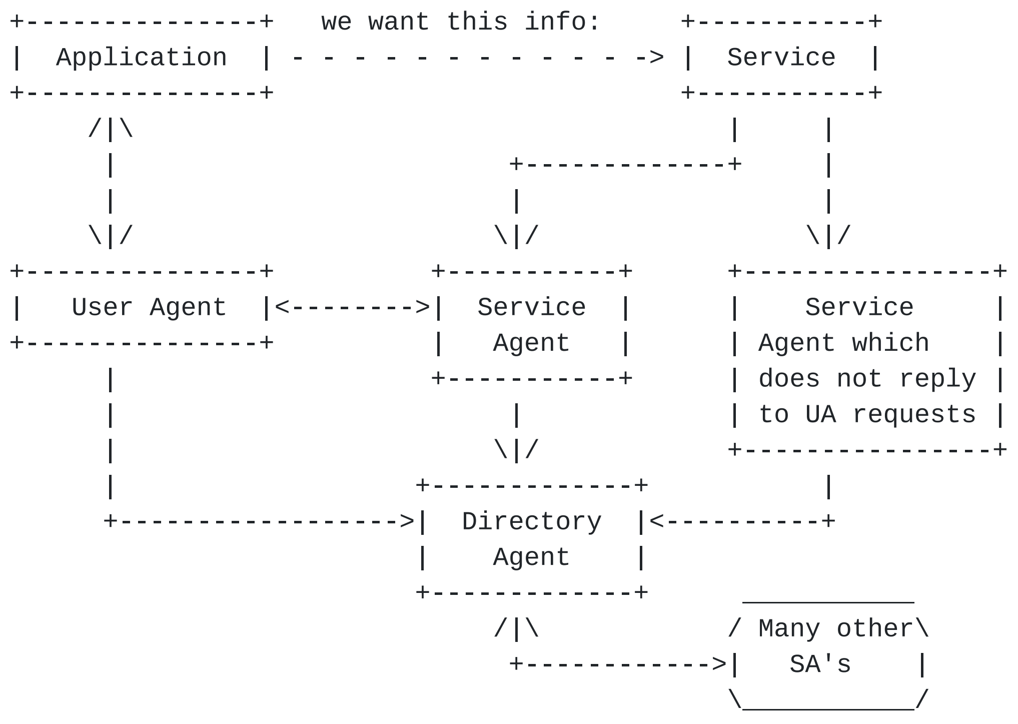
3. Protocol Overview

The basic operation in Service Location is that a client attempts to discover the location of a Service. In smaller installations, each service will be configured to respond individually to each client. In larger installations, services will register their services with one or more Directory Agents, and clients will contact the Directory Agent to fulfill requests for Service Location information. Clients may discover the whereabouts of a Directory

Agent by preconfiguration, DHCP [2, 12], or by issuing queries to the Directory Agent Discovery multicast address.

3.1. Protocol Transactions

The diagram below illustrates the relationships described below:



```
      +---------------+   we want this info:     +-----------+
      |  Application  | - - - - - - - - - - - -> |  Service  |
      +---------------+                          +-----------+
           /|\                                      |     |
            |                         +-------------+     |
            |                         |                   |
           \|/                       \|/                 \|/
      +---------------+          +-----------+      +----------------+
      |   User Agent  |<-------->|  Service  |      |    Service     |
      +---------------+          |   Agent   |      | Agent which    |
            |                    +-----------+      | does not reply |
            |                         |             | to UA requests |
            |                        \|/            +----------------+
            |                   +-------------+           |
            +------------------>|  Directory  |<----------+
                                |    Agent    |
                                +-------------+      ___________
                                     /|\            / Many other\
                                      +------------>|   SA's    |
                                                    \___________/
```

The following describes the operations a User Agent would employ to find services on the site's network. The User Agent needs no configuration to begin network interaction. The User Agent can acquire information to construct predicates which describe the services that match the user's needs. The User Agent may build on the information received in earlier network requests to find the Service Agents advertising service information.

A User Agent will operate two ways: If the User Agent has already obtained the location of a Directory Agent, the User Agent will unicast a request to it in order to resolve a particular request. The Directory Agent will unicast a reply to the User Agent. The User Agent will retry a request to a Directory Agent until it gets a reply, so if the Directory Agent cannot service the request (say it has no information) it must return an response with zero values, possibly with an error code set.

If the User Agent does not have knowledge of a Directory Agent or if there are no Directory Agents available on the site network, a second

mode of discovery may be used. The User Agent multicasts a request to the service-specific multicast address, to which the service it wishes to locate will respond. All the Service Agents which are listening to this multicast address will respond, provided they can satisfy the User Agent's request. A similar mechanism is used for Directory Agent discovery; see section 5.2. Service Agents which have no information for the User Agent MUST NOT respond.

When a User Agent wishes to obtain an enumeration of ALL services which satisfy the query, a retransmission/convergence algorithm is used. The User Agent resends the request, together with a list of previous responders. Only those Service Agents which are not on the list respond. Once there are no new responses to the request the accumulation of responses is deemed complete. Depending on the length of the request, around 60 previous responders may be listed in a single datagram. If there are more responders than this, the scaling mechanisms described in section 3.7 should be used.

While the multicast/convergence model may be important for discovering services (such as Directory Agents) it is the exception rather than the rule. Once a User Agent knows of the location of a Directory Agent, it will use a unicast request/response transaction.

The Service Agent SHOULD listen for multicast requests on the service-specific multicast address, and MUST register with an available Directory Agent. This Directory Agent will resolve requests from User Agents which are unicasted using TCP or UDP. This means that a Directory Agent must first be discovered, using DHCP, the DA Discovery Multicast address, the multicast mechanism described above, or manual configuration. See section 5.2.

A Service Agent which does not respond to multicast requests will not be useful in the absence of Directory Agents. Some Service Agents may not include this functionality, if an especially lightweight implementation is required.

If the service is to become unavailable, it should be deregistered with the Directory Agent. The Directory Agent responds with an acknowledgment to either a registration or deregistration. Service Registrations include a lifetime, and will eventually expire. Service Registrations need to be refreshed by the Service Agent before their Lifetime runs out. If need be, Service Agents can advertise signed URLs to prove that they are authorized to provide the service.

3.2. Schemes

The Service Location Protocol, designed as a way for clients to access resources on the network, is a natural application for Universal Resource Locators (URLs). It is intended that by re-using URL specification and technology from the World Wide Web, clients and servers will be more flexible and able to be written using already existing code. Moreover, it is hoped that browsers will be written to take advantage of the similarity in locator format, so that a client can dynamically formulate requests for services that are resolved differently depending upon the circumstances.

3.2.1. The ``service:'' URL scheme

The service URL scheme is used by Service Location. It is used to specify a Service Location. Many Service Types will be named by including a scheme name after the ``service:''  scheme name.  Service Types are used by SAs to register and deregister Services with DAs. It is also used by SAs and DAs to return Service Replies to UAs.  The formal definition of the ``service:'' URL scheme is in section 20.2. The format of the information which follows the ``service:'' scheme should as closely as possible follow the URL structure and semantics as formalized by the IETF standardization process.

Well known Service Types are registered with the IANA and templates are available as RFCs. Private Service Types may also be supported.

3.3. Standard Attribute Definitions

Service Types used with the Service Location Protocol must describe the following:

      Service Type string of the service
      Attributes and Keywords
      Attribute Descriptions and interpretations

Service Types not registered with IANA will use their own Naming Authority string. The registration process for new Service Types is defined in [14].

Services which advertise a particular Service Type must support the complete set of standardized attributes. They may support additional attributes, beyond the standardized set. Unrecognized attributes MUST be ignored by User Agents.

Service Type names which begin with "x-" are guaranteed not to conflict with any officially registered Service Type names. It is suggested that this prefix be used for experimental or private Service Type names. Similarly, attribute names which begin with "x-" are guaranteed not to be used for any officially registered attribute names.

A service of a given Service Type should accept the networking protocol which is implied in its definition. If a Service Type can accept multiple protocols, configuration information SHOULD be included in the Service Type attribute information. This configuration information will enable an application to use the results of a Service Request and Attribute Request to directly connect to a service.

See section 20.2.1 for the format of a Service Type String as used in the Service Location Protocol.

3.4. Naming Authority

The Naming Authority of a service defines the meaning of the Service Types and attributes registered with and provided by Service Location. The Naming Authority itself is a string which uniquely identifies an organization. If no string is provided IANA is the default. IANA stands for the Internet Assigned Numbers Authority.

Naming Authorities may define Service Types which are experimental, proprietary or for private use. The procedure to use is to create a 'unique' Naming Authority string and then specify the Standard Attribute Definitions as described above. This Naming Authority will accompany registration and queries, as described in sections 5 and 9.

3.5. Interpretation of Service Location Replies

Replies should be considered to be valid at the time of delivery. The service may, however, fail or change between the time of the reply and the moment an application seeks to make use of the service. The application making use of Service Location MUST be prepared for the possibility that the service information provided is either stale or incomplete. In the case where the service information provided does not allow a User Agent to connect to a service as desired, the Service Request and/or Attribute Request may be resubmitted.

Service specific configuration information (such as which protocol to use) should be included as attribute information in Service

Registrations. These configuration attributes will be used by applications which interpret the Service Location Reply.

3.6. Use of TCP, UDP and Multicast in Service Location

The Service Location Protocol requires the implementation of UDP (connectionless) and TCP (connection oriented) transport protocols. The latter is used for bulk transfer, only when necessary. Connections are always initiated by an agent request or registration, not by a replying Directory Agent. Service Agents and User Agents use ephemeral ports for transmitting information to the service location port, which is 427.

The Service Location discovery mechanisms typically multicast messages to as many enterprise networks as needed to establish service availability. The protocol will operate in a broadcast environment with limitations detailed in section 3.6.1.

3.6.1. **Multicast vs.** Broadcast

The Service Location Protocol was designed for use in networks where DHCP is available, or multicast is supported at the network layer. To support this protocol when only network layer broadcast is supported, the following procedures may be followed.

3.6.1.1. Single Subnet

If a network is not connected to any other networks simple network layer broadcasts will work in place of multicast.

Service Agents SHOULD and Directory Agents MUST listen for broadcast Service Location request messages to the Service Location port. This allows UAs which lack multicast capabilities to still make use of Service Location on a single subnet.

3.6.1.2. Multiple Subnets

The Directory Agent provides a central clearing house of information for User Agents. If the network is designed so that a Directory Agent address is statically configured with each User Agent and Service Agent, the Directory Agent will act as a bridge for information that resides on different subnets. The Directory Agent address can be dynamically configured with Agents using DHCP. The address can also be determined by static configuration.

As dynamic discovery is not feasible in a broadcast environment with multiple subnets and manual configuration is difficult, deploying DAs to serve enterprises with multiple subnets will require use of multicast discovery with multiple hops (i.e., TTL > 1 in the IP header).

3.6.2. Service-Specific Multicast Address

This mechanism is used so that the number of datagrams any one service agent receives is minimized. The Service Location General Multicast Address MAY be used to query for any service, though one SHOULD use the service-specific multicast address if it exists.

If the site network does not support multicast then the query SHOULD be broadcast to the Service Location port. If, on the other hand, the underlying hardware will not support the number of needed multicast addresses the Service Location General Multicast Address MAY be used. Service Agents MUST listen on this multicast address as well as the service-specific multicast addresses for the service types they advertise.

Service-Specific Multicast Addresses are computed by calculating a string hash on the Service Type string. The Service Type string MUST first be converted to an ASCII string from whatever character set it is represented in, so the hash will have well-defined results.

The string hash function is modified from a code fragment attributed to Chris Torek:

```
     /*
      *  SLPhash returns a hash value in the range 0-1023 for a
      *  string of single-byte characters, of specified length.
      */
     unsigned long SLPhash (const char *pc, unsigned int length) {
         unsigned long h = 0;
 while (length-- != 0) {
             h *= 33;
             h += *pc++;
         }
         return (0x3FF & h);  /* round to a range of 0-1023 */
     }
```

This value is added to the base range of Service Specific Discovery Addresses, to be assigned by IANA. These will be 1024 contiguous multicast addresses.

3.7. Service Location Scaling, and Multicast Operating Modes

In a very small network, with few nodes, no DA is required. A user agent can detect services by multicasting requests. Service Agents will then reply to them. Further, Service Agents which respond to user requests must be used to make service information available. This does not scale to environments with many hosts and services.

When scaling Service Location systems to intermediate sized networks, a central repository (Directory Agent) may be added to reduce the number of Service Location messages transmitted in the network infrastructure. Since the central repository can respond to all Service and Attribute Requests, fewer Service and Attribute Replies will be needed; for the same reason, there is no need to differentiate between Directory Agents.

A site may also grow to such a size that it is not feasible to maintain only one central repository of service information. In this case more Directory Agents are needed. The services (and service agents) advertised by the several Directory Agents are collected together into logical groupings called "Scopes".

All Service Registrations that have a scope must be registered with all DAs (within the appropriate multicast radius) of that scope which have been or are subsequently discovered. Service Registrations which have no scope are only registered with unscoped DAs. User Agents make requests of DAs whose scope they are configured to use.

Service Agents MUST register with unscoped DAs even if they are configured to specifically register with DAs which have a specific scope or set of scopes. User Agents MAY query DAs without scopes, even if they are configured to use DAs with a certain scope. This is because any DA with no scope will have all the available service information.

Scoped user agents SHOULD always use a DA which supports their configured scope when possible instead of an unscoped DA. This will prevent the unscoped DAs from becoming overused and thus a scaling problem.

It is possible to specially configure Service Agents to register only with a specific set of DAs (see Section 22.1). In that case, services may not be available to User Agents via all Directory Agents, but some network administrators may deem this appropriate.

There are thus 3 distinct operating modes. The first requires no administrative intervention. The second requires only that a DA be run. The last requires that all DAs be configured to have scope and

that a coherent strategy of assigning scopes to services be followed. Users must be instructed which scopes are appropriate for them to use. This administrative effort will allow users and applications to subsequently dynamically discover services without assistance.

The first mode (no DAs) is intended for a LAN. The second mode (using a DA or DAs, but not using scopes) scales well to a group of interconnected LANs with a limited number of hosts. The third mode (with DAs and scopes) allows the SLP protocol to be used in an internetworked campus environment.

If scoped DAs are used, they will not accept unscoped registrations or requests. UAs which issue unscoped requests will discover only unscoped services. They SHOULD use a scope in their requests if possible and SHOULD use a DA with their scope in preference to an unscoped DA. In a large campus environment it would be a bad idea to have ANY unscoped DAs: They attract ALL registrations and will thus present a scaling problem eventually.

A subsequent protocol document will describe mechanisms for supporting a service discovery protocol for the global Internet.

4. Service Location General Message Format

The following header is used in all of the message descriptions below and is abbreviated by using "Service Location header =" followed by the function being used.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Version    |    Function   |            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |O|M|U|A|F| rsvd|    Dialect    |        Language Code          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        Char Encoding          |              XID              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

    Version This protocol document defines version 1 of the Service
             Location protocol.

Function Service Location datagrams can be identified as to their operation by the function field. The following are the defined operations:

Message Type	Abbreviation	Function Value
Service Request	SrvReq	1
Service Reply	SrvRply	2
Service Registration	SrvReg	3
Service Deregister	SrvDereg	4
Service Acknowledge	SrvAck	5
Attribute Request	AttrRqst	6
Attribute Reply	AttrRply	7
DA Advertisement	DAAdvert	8
Service Type Request	SrvTypeRqst	9
Service Type Reply	SrvTypeRply	10

Length The number of bytes in the message, including the Service Location Header.

O The 'Overflow' bit. See Section 18 for the use of this field.

M The 'Monolingual' bit. Requests with this bit set indicate the User Agent will only accept responses in the language (see section 17) that is indicated by the Service or Attribute Request.

U The 'URL Authentication Present' bit. See sections 4.2, 4.3, 9, and 11 for the use of this field.

A The 'Attribute Authentication Present' bit. See sections 4.2, 4.3, and 13 for the use of this field.

F If the 'F' bit is set in a Service Acknowledgement, the directory agent has registered the service as a new entry, not as an updated entry.

rsvd MUST be zero.

Dialect Dialect tags will be used by future versions of the Service Location Protocol to indicate a variant of vocabulary used. This field is reserved and MUST be set to 0 for compatibility with future versions of the Service Location Protocol.

Language Code
Strings within the remainder of the message which follows are to be interpreted in the language encoded (see section 17 and appendix A) in this field.

Character Encoding
The characters making up strings within the remainder of the message may be encoded in any standardized encoding (see section 17.1).

Transaction Identifier (XID)
The XID (transaction ID) field allows the requester to match replies to individual requests (see section 4.1).

Note that, whenever there is an Attribute Authentication block, there will also be a URL Authentication block. Thus, it is an error to have the 'A' bit set without also having the 'U' bit set.

4.1. Use of Transaction IDs (XIDs)

Retransmission is used to ensure reliable transactions in the Service Location Protocol. If a User Agent or Service Agent sends a message and fails to receive an expected response, the message will be sent again. Retransmission of the same Service Location datagram should not contain an updated XID. It is quite possible the original request reached the DA or SA, but reply failed to reach the requester. Using the same XID allows the DA or SA to cache its reply to the original request and then send it again, should a duplicate request arrive. This cached information should only be held very briefly (CONFIG_INTERVAL_0.) Any registration or deregistration at a Directory Agent, or change of service information at a SA should flush this cache so that the information returned to the client is always valid.

The requester creates the XID from an initial random seed and increments it by one for each request it makes. The XIDs will eventually wrap back to zero and continue incrementing from there.

Directory Agents use XID values in their DA Advertisements to indicate their state (see section 15.2).

4.2. URL Entries

When URLs are registered, they have lifetimes and lengths, and may be authenticated. These values are associated with the URL for

the duration of the registration. The association is known as a "URL-entry", and has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Lifetime            |        Length of URL          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
\                              URL                              \
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              (if present) URL Authentication Block .....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Lifetime	The length of time that the registration is valid, in the absence of later registrations or deregistration.
Length of URL	The length of the URL, measured in bytes and < 32768.
URL Authentication Block	(if present) A timestamped authenticator (section 4.3)

The URL conforms to RFC 1738 [6]. If the 'U' bit is set in the message header, the URL is followed by an URL Authentication Block. If the scheme used in the URL does not have a standardized representation, the minimal requirement is:

```
service:<srvtype>://<addr-spec>
```

"service" is the URL scheme of all Service Location Information included in service registrations and service replies. Each URL entry contains the service:<srvtype> scheme name. It may also include an <addr-spec> except in the case of a reply to a Service Type request (see section 7).

4.3. Authentication Blocks

Authentication blocks are used to authenticate service registrations and deregistrations. URLs are registered along with an URL Authentication block to retain the authentication information in the URL entry for subsequent use by User Agents who receive a Service Reply containing the URL entry. Service attributes are registered along with an Attribute Authentication block. Both authentication blocks have the format illustrated below.

If a service registration is accompanied by authentication which can be validated by the DA, the DA MUST validate any subsequent service deregistrations, so that unauthorized entities cannot invalidate such registered services. Likewise, if a service registration is accompanied by an Attribute Authentication block which can be validated by the DA, the DA MUST validate any subsequent attribute registrations, so that unauthorized entities cannot invalidate such registered attributes.

To avoid replay attacks which use previously validated deregistrations, the deregistration or attribute registration message must contain a timestamp for use by the DA. To avoid replay attacks which use previously validated registrations to nullify a valid deregistration, registrations must also contain a timestamp.

An authentication block has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                           Timestamp                           +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Block Structure Descriptor   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Structured Authenticator ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Timestamp A 64-bit value formatted as specified by the Network
         Time Protocol (NTP) [17].

Block Structure Descriptor (BSD)
         A value describing the structure of the Authenticator.
         The only value currently defined is 1, for
         Object-Identifier.

Length   The length of the Authenticator

Structured Authenticator
         An algorithm specification, and the authentication data
         produced by the algorithm.

The algorithms to use for encryption, decryption and message digest calculation are identified within the authenticator itself. When producing a URL Authentication block, the authentication data produced by the algorithm identified within the Structured

Authenticator is a message digest D calculated over the following ordered stream of data:

```
    CHARACTER ENCODING OF URL   (2 bytes in network byte order)
    LIFETIME                    (2 bytes in network byte order)
    LENGTH OF URL               (2 bytes in network byte order)
    URL                         (n bytes)
    TIMESTAMP                   (8 bytes in SNTP format [17])
```

When producing a URL Authentication block, the authentication data produced by the algorithm identified within the Structured Authenticator is a message digest D calculated over the following ordered stream of data:

```
    ATTRIBUTE CHARACTER ENCODING   (2 bytes in network byte order)
    LENGTH OF ATTRIBUTES           (2 bytes in network byte order)
    ATTRIBUTES                     (n bytes)
    TIMESTAMP                      (8 bytes in SNTP format [17])
```

In either case, D is then encrypted using the key of the protected scope to produce N bytes of authenticator info. The length field of the authenticator block is set to N. The N bytes of the encrypted message digest D follow.

To verify an authenticator, the UA or DA computes the message digest D from the same data as the SA did. The UA or DA then decrypts the N bytes of authenticator data to obtain D'. If D is the same as D' the URL has been authenticated.

There is currently only one value defined for the Block Structure Descriptor (BSD):

1	The Structured Authenticator uses AlgorithmIdentifiers [10] in ASN.1 [9] notation to identify the algorithm and encrypted data format.

BSD values 0 and 2-255 are reserved for future standardized structure descriptors. Other values may be defined in a site-dependent fashion, and are not intended be used to denote well-known structure descriptor formats.

Every Service Location Protocol entity (User Agent, Service Agent, or Directory Agent) which is to be configured for use with protected scopes MUST implement "md5WithRSAEncryption" [4] and be able to associate it with BSD value == 1 to verify the authentication.

4.4. URL Entry Lifetime

The Lifetime field is set to the number of seconds the reply can be cached by any agent. A value of 0 means the information must not be cached. User Agents MAY cache service information, but if they do, they must provide a way for applications to flush this cached information and issue the request directly onto the network.

Services should be registered with DAs with a Lifetime, the suggested value being CONFIG_INTERVAL_1. The service must be reregistered before this interval elapses, or the service advertisement will no longer be available. Thus, services which vanish and fail to deregister eventually become automatically deregistered.

5. Service Request Message Format

The Service Request is used to obtain URLs from a Directory Agent or Service Agents.

The format of the Service Request is as follows:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Service Location header (function = SrvReq)           |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |length of prev resp list string|<Previous Responders Addr Spec>|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                  <Previous Responders Addr Spec>              \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  length of predicate string   |  Service Request <predicate>  |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \               Service Request <predicate>, contd.             \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If a UA issues a request which will result in a reply which is too large, the SA or DA will return an abbreviated response (in a datagram the size of the site's MTU) which has the 'Overflow' bit flag set. The UA must then issue the request again using TCP.

The <Previous Responders Addr Spec> is described in sections 7 and 20.1.

After a User Agent restarts (say, after rebooting of a system, loading of the network kernel), Service Requests should be delayed for some random time uniformly distributed within a one second interval centered about a configured delay value (by default, CONFIG_INTERVAL_4).

The Service Request allows the User Agent to specify the Service Type of the service and a Predicate in a specific language. The general form of a Service Request is shown below:

```
   <srvtype>[.<na>]/[<scope>]/[<where>]/
```

The punctuation is necessary even where the fields are omitted.

- The <srvtype> refers to the Service Type. For each type of service available, there is a unique Service type name string. See section 20.2.1.

- The <na> is the Naming Authority. This string determines the semantic interpretation of the attribute information in the <where> part of the Service Request.

- The <scope> is a string used to restrict the range of the query. Scope is determined administratively, at a given site. It is not necessarily related to network topology (see Section 16). Leaving this field out means that the request can be satisfied only by unscoped service advertisements.

- The <where> string is the Where Clause of the request. It contains a query which allows the selection of those service instances which the User Agent is interested in. The query includes attributes, boolean operators and relations. (See section 5.3.)

In the case of a multicast service request, a list of previous responders is sent. This list will prevent those in the list from responding, to be sure that responses from other sources are not drowned out. The request is multicast repeatedly (with a recommended wait interval of CONFIG_INTERVAL_2) until there are no new responses, or a certain time (CONFIG_INTERVAL_3) has elapsed. Different timing values are applied to a Service Request used for Directory Agent Discovery, see Section 5.2.

In order for a request to succeed in matching registered information, the following conditions must be met:

1. The result must have the same Service Type as the request.

2. It must have the same Naming Authority.

3. It must have the same scope. (If the scope of the request was omitted, the request will only match services which were registered with no scope. Note that a scoped request WILL match all unscoped Services).

4. The conditions specified in the Where Clause must match the attributes and keywords registered for the service.

5.1. Service Request Usage

The User Agent may form Service Requests using preconfigured knowledge of a Service Type's attributes. It may also issue Attribute Requests to obtain the attribute values for a Service Type before issuing Service Requests (see Section 13). Having obtained the attributes which describe a particular kind of service from an Attribute Request, or using configured knowledge of a service's attributes, the User Agent can build a predicate that describes the service needs of the user.

Service Requests may be sent directly to a Directory Agent. Suppose a printer supporting the lpr protocol is needed on the 12th floor which has UNRESTRICTED_ACCESS and prints 12 pages per minute. Suppose further that a Attribute Request indicates that there is a printer on the 12th floor, a printer that prints 12 pages per minute, and a printer that offers UNRESTRICTED_ACCESS. To check whether they are same printer, issue the following request:

```
lpr//(& (PAGES PER MINUTE==12)
         (UNRESTRICTED_ACCESS)
         (LOCATION==12th FLOOR))/
```

Suppose there is no such printer. The Directory Agent responds with a Service Reply with 0 in the number of responses and no reply values.

The User Agent then tries a less restrictive query to find a printer, using the 12th floor as "where" criteria.

```
lpr//(LOCATION==12th FLOOR)/
```

In this case, there is now only one reply:

```
Returned URL:   service:lpr://igore.wco.ftp.com:515/draft
```

The Address Specification for the printer is: igore.wco.ftp.com:515, containing the name of the host managing the requested printer. Files would be printed by spooling to that port on that host. The word 'draft' refers to the name of the print queue the lpr server supports.

In the absence of a Directory Agent, the request above could be multicast. In this case it would be sent to the Service Specific Multicast Address for "service:printer" and not to the Directory Agent. Service Agents that can satisfy the predicate will reply. Service Agents which cannot support the character set of the request MUST return CHARSET_NOT_UNDERSTOOD in the SrvRply. In all other circumstances, Service Agents which cannot satisfy the reply do not send any reply at all.

The only way a User Agent can be sure there are no services which match the query is by retrying the request (CONFIG_INTERVAL_8). If no response comes, the User Agent gives up and assumes there are no such printers.

Another form of query is a simpler 'join' query. Its syntax has no parentheses or logical operators. Each term is conjoined (AND-ed together.) Rewriting the initial query provides an example:

```
   lpr//PAGES PER MINUTE==12,
        UNRESTRICTED_ACCESS,
        LOCATION==12th FLOOR/
```

5.2. Directory Agent Discovery Request

Normally a Service Request returns a Service Reply. The sole exception to this is a Service Request for the Service Type "directory-agent". This Service Request is answered with a DA Advertisement.

Without configured knowledge of a Directory Agent (DA), a User Agent or Service Agent uses a Service Request to discover a DA. (See section 15.1 for mechanisms by which a client may be configured to have knowledge of a DA.) Such a Service Request used for Directory Agent Discovery includes a predicate of the form:

```
      directory-agent///
```

This query is always sent to the Directory Agent Discovery multicast address. The Service Type of a Directory Agent is "directory-agent", hence it is the Service Type used in the request. No scope is included in the request, so all Directory Agents will reply. This is

the only request which omits a scope which all Directory Agents MUST respond to. Normally, a Directory Agent with a scope ONLY responds to requests with that scope. No Naming Authority is included, so "IANA" is assumed. We want to reach all the available directory agents. If the scope were supplied, only DAs supporting that scope would reply.

DA Advertisement Replies may arrive from different sources, similar in form to:

```
URL returned:   service:directory-agent://slp-resolver.catch22.com
Scope returned: ACCOUNTING

URL returned:   service:directory-agent://204.182.15.66
Scope returned: JANITORIAL SERVICES
```

The DA Advertisement format is defined in Section 14.

If the goal is merely to discover any Directory Agent, the first reply will do. If the goal, however, is to discover all reachable DAs, the request must be retransmitted after an interval (the recommended time is CONFIG_INTERVAL_5). This retransmitted request will include a list of DAs which have already responded. See sections 7 and 20.1. Directory Agents which receive the request will only respond if they are not on this list. After there are no new replies, all DAs are presumed to have been discovered.

If a DA fails to respond after CONFIG_INTERVAL_6 seconds, the UA or Service Agent should use a different DA. DA addresses may be cached from previous discovery attempts, preconfigured, or by use of DHCP (see section 15.2). If no such DA responds, DA discovery should be used to find a new DA. Only after CONFIG_INTERVAL_7 seconds should it be assumed that no DA exists and multicast based Service Requests should be used.

5.3. Explanation of Terms of Predicate Grammar

A predicate has a simple structure, which depends on parentheses, commas and slashes to delimit the elements. Examples of proper usage are given throughout this document. The terms used in the grammar are as follows:

predicate:

Placed in a Service Request, this is interpreted by a Service Agent or Directory Agent to determine what information to return.

scope:

If this is absent in a Service Request, the request will match only services registered without a scope. If it is present, only services registered under that scope or are unscoped will match the request.

where-clause:

This determines which services the request matches. An empty where-clause will match all services. The request will be limited to services which have the specified Service Type, so the where-clause is not the sole factor in picking out which services match the request.

where-list:

The where-list is a logical expression. It can be a single expression, a disjunction or a conjunction. A single expression must apply for the where-clause to match. A disjunction matches if any expression in the OR list matches. A conjunction matches only if all elements in the AND list match.

Note that there is no logical negation operator: This is because there is no notion of returning "everything except" what matches a given criteria.

A where-list can be nested and complex. For example, the following requires that three subexpressions must all be true:

```
       (& (| <query-item> <query-item>)
          <query-item>
          (& <query-item> <query-item> <query-item>)
       )
```

Notice that white space, tabs or carriage returns can be added anywhere outside query-items. Each list has 2 or more items in it, and lists can be nested. Services which fulfill the entire logical expression match the where-clause.

'(' '|' <query-item> ')' and '(' '&' <query-item> ')' are degenerate expressions but they should be tolerated. They are equivalent to <query-item>.

query-item:

A query item has the form:

```
              '(' <attr-tag> <comp-op> <attr-val> ')'

        or

              '(' <keyword> ')'

        Examples of this would be:

              (SOME ATTRIBUTE == SOME VALUE)
              (RESERVED)
              (QUEUE LENGTH <= 234)
```

query-join:

The query-join is a comma delimited list of conditions which the service must satisfy in order to match the query. The items are considered to be logically conjoined. Thus the query-join:

```
ATTR1=VALUE1, KEYWORD1, KEYWORD2, ATTR2>=34
```

is equivalent to the where-list:

```
(& (ATTR1=VALUE1) (KEYWORD1) (KEYWORD2) (ATTR2>=34))
```

The query-join cannot be mixed with a where-list. It is provided as a convenient mechanism to provide a statement of necessary conditions without building a logical expression.

5.4. Service Request Predicate Grammar

Service Requests can precisely describe the services they need by including a Predicate the body of the Request. This Predicate must be constructed according to the grammar below.

```
 <predicate>  ::= <srvtype>['.'<na>]'/'<scope>'/'<where>'/'

 <srvtype>    ::= string representing type of service.  Only
                  alphanumeric characters, '+', and '-' are allowed.

 <na>         ::= string representing the Naming Authority.
                  Only alphanumeric characters, '+',
                  and '-' are allowed.  If this field is
                  omitted then "IANA" is assumed.

 <scope>      ::= string representing the directory agent scope.
                  '/', ',' (comma) and ':'  are not allowed in
```

```
                     this string.  The scopes "LOCAL" and "REMOTE"
                     are reserved.

    <attr-tag>   ::= class name of an attribute of a given Service
                     Type.  This tag cannot include the following
                     characters:  '(', ')', ',', '=', '!', '>',
                     '<', '/', '*', except where escaped (see 17.1.)

    <keyword>    ::= a class name of an attribute which will have
                     no values.  This string has the same limits
                     as the <attr-tag>, except that white space
                     internal to the keyword is illegal.

    <where>      ::= <where-any> |
                     <where-list> |
                     <query-join>

    <where-any>  ::=
                     That is NOTHING, or white space.

    <where-list> ::= '(' '&' <where-list> <query-list> ')' |
                     '(' '|' <where-list> <query-list> ')' |
                     '(' <keyword> ')'
                     '(' <attr-tag> <comp-op> <attr-val> ')'

    <query-list> ::= <where-list> |
                     <where-list> <query-list>
    <query-join> ::= <keyword> |
                     <join-item> |
                     <query-join> ',' <keyword> |
                     <query-join> ',' <join-item>

    <join-item>  ::= <attr-tag> <comp-op> <attr-val>

    <comp-op>    ::= "!=" | "==" | '<' | "<=" | '>' | ">="

    <attr-val>   ::= any string (see Section 20.5 for the ways
                     in which attr-vals are interpreted.)
                     Value strings may not contain '/', ','
                     '=', '<', '>', or '*' except where escaped (see 17.1.).

                     '(' and ')' may be used in attribute values
                     for the purpose of encoding a binary values.
                     Binary encodings (See 20.5) may
                     include the above reserved characters.
```

5.5. String Matching for Requests

All strings are case insensitive, with respect to string matching on queries. All preceding or trailing blanks should not be considered for a match, but blanks internal to a string are relevant.
For example, " Some String " matches "SOME STRING",
but not "some string".

String matching may only be performed over the same character sets. If a request cannot be satisfied due to a lack of support for the character set of the request a CHARSET_NOT_UNDERSTOOD error is returned.

String comparisons (using comparison operators such as '<' or '>=') are done using lexical ordering in the character set of the registration, not using any language specific rules. The ordering is strictly by the character value, i.e. "0" < "A" is true when the character set is US-ASCII, since "0" has the value of 48 and "A" has the value 65.

The special character '*' may precede or follow a string in order to allow substring matching. If the '*' precedes a string, it matches any attribute value which ends with the string. If the string ends with a '*', it matches any attribute value which begins with the string. Finally, if a string begins and ends with a '*', the string will match any attribute value which contains the string.

Examples:

```
     "bob*" matches "bob", "bobcat", and "bob and sue"
     "*bob" matches "bob", "bigbob", and "sue and bob"
     "*bob*" matches "bob", "bobcat", "bigbob", and "a bob I know"
```

String matching is done after escape sequences have been substituted.
See sections 17, 5.3, 17.1.

6. Service Reply Message Format

The format of the Service Reply Message is:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |          Service Location header (function = SrvRply)         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Error Code            |         URL Entry count       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                         <URL Entry 1> ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                               .                               |
  \                               .                               \
  |                               .                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                         <URL Entry N> ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Each Service Reply message is composed of a list of URL Entries.

The Error Code may have one of the following values:

0   Success

LANGUAGE_NOT_SUPPORTED
A SA or DA returns this when a request is received from a UA which is in a language for which there is no registered Service Information and the request arrived with the Monolingual bit set. See Section 17.

PROTOCOL_PARSE_ERROR
A SA or DA returns this error when a SrvRply is received which cannot be parsed or the declared string lengths overrun the message.

SCOPE_NOT_SUPPORTED
A DA will return this error if it receives a request which has a scope not supported by the DA. An SA will not return this error; it will simply not reply to the multicast request.

CHARSET_NOT_UNDERSTOOD
If the DA or SA receives a request or registration in a character set which it does not support, it will return this error.

Each <URL Entry> in the list has the form defined in Section 4.2. The URL entries in the reply have no delimiters between them, other than the length fields. The URL length fields indicate where the URL strings end. If the presence of an URL Authenticator block is signalled by the 'U' bit, the length of the authenticator block is determined by information within the block as discussed in section 4.3. A User Agent MAY use the authentication block to determine whether the Service Agent advertising the URL is, in fact, authorized to offer the indicated service. If, in a list of URL entries, some of the URLs indicate services which are in protected scopes (see section 16.1) while other URLs in the list indicate services which are not in protected scopes, the latter must still have Authentication Blocks, but the length of the authentcitor is shown as zero, and no authentication need be done.

7. Service Type Request Message Format

The Service Type Request is used to determine all the types of services supported on a network.

The request should be sent directly to a DA (though it may also be sent to the Service Location General Multicast Address), in order to find out all services available on the site network (which are advertised by Directory Agents and Service Agents.) If no DA is available, a User Agent MAY issue more than one request to insure that all replies have been received. In each subsequent request, a User Agent includes those Service Types that it is aware of. When no new replies arrive within CONFIG_INTERVAL_3 from a request, the User Agent can presume that it has acquired a complete set of available Service Types.

The format of a Service Type Request is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Service Location header (function = SrvTypeRqst)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  length of prev resp string   |<Previous Responders Addr Spec>|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
\                  <Previous Responders Addr Spec>              \
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   length of naming authority  |   <Naming Authority String>   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
\            <Naming Authority String>, continued               \
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     length of Scope String    |         <Scope String>        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
\                   <Scope String>, continued                   \
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note that the <Previous Responders Addr Spec> is a comma delimited list. (See section 20.1.) The 'length of prev responder list' field indicates the length of the comma delimited list string. A previous responder list with 3 elements takes this form:

```
<addr-spec>,<addr-spec>,<addr-spec>
```

The Naming Authority, if included, will limit the replies to Service Type Requests to Service Types which have the specified Naming Authority. If this field is omitted (i.e., the length field is zero), the default Naming Authority ("IANA") is assumed. If the length field is -1, service types from all naming authorities are requested.

The Scope String Field, if included, will limit replies to Service Types which have the specified scope or are unscoped. If this field is omitted, all Service Types (from the specified Naming Authority) are returned.

8. Service Type Reply Message Format

The Service Type Reply has the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        Service Location header (function = SrvTypeRply)       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Error Code           |    number of service types    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   \                     <Service Type Item 1>                     \
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             . . .                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   \                     <Service Type Item N>                     \
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of a Service Type Item is as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | length of Service Type String |     <Service Type String>     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   \                 <Service Type String>, continued              \
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Error Code may have one of the following values:

```
    0        Success

    PROTOCOL_PARSE_ERROR
             A SA or DA returns this error when a SrvTypeRqst is
             received which cannot be parsed.

    SCOPE_NOT_SUPPORTED
             A DA which is configured to have a scope will return this
             error if it receives a SrvTypeRqst which is set to have a
             scope which it does not support.  An SA will not return
             this error, it will simply silently discard the multicast
             request.
```

CHARSET_NOT_UNDERSTOOD
If the DA receives a SrvTypeRqst in a character set which it does not support, it MUST use this error.

The service type's name is provided in the <Service Type String>. If the service type has a naming authority other than "IANA" it should be returned following the service type string and a "." character. See section 20.2.1 for the formal definition of this field. User Agents calculate Service Specific Multicast addresses based on a hash of the Service Type (see Section 3.6.2). This multicast address may then be used for issuing Service and Attribute Requests directly to SAs.

The following are examples of Service Type Strings which might be found in Service Type Replies:

```
service:lpr://
service:http://
service:nfs://
```

9. Service Registration Message Format

After a Service Agent has found a Directory Agent, it begins to register its advertised services one at a time. A Service Agent must wait for some random time uniformly distributed within the range specified by CONFIG_INTERVAL_11 before registering again. Registration is done using the Service Registration message specifying all attributes for a service. If the service registration in a protected scope 16.1, then the service MUST include both a URL Authentication block and an Attribute Authentication block (see section 4.3). In that case, the service agent MUST set both the 'U' bit and the 'A' bit (see section 4).

A Directory Agent must acknowledge each service registration request. If authentication blocks are included, the Directory Agent MUST verify the authentication before registering the service. This requires obtaining key information, either by preconfiguration, maintenance of a security association with the service agent, or acquiring the appropriate certificate.

The format of a Service Registration is:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |           Service Location header (function = SrvReg)         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                          <URL-Entry>                          \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  Length of Attr List String   |          <attr-list>          |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                    <attr-list>, Continued.                    \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |    (if present) Attribute Authentication Block ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The <URL-Entry> is defined at the end of Section 4.2. The <attr-list> is defined in Section 20.3. The Attribute Authentication Block, which is only present if the 'A' bit is set in the message header, is defined in section 4.3.

Service registration may use a connectionless protocol (e.g. UDP), or a connection oriented protocol (e.g. TCP). If the registration operation may contain more information than can be sent in one datagram, the Service Agent MUST use a connection oriented protocol to register itself with the DA. When a Service Agent registers the same attribute class more than once for a service instance, the Directory Agent overwrites the all the values associated with that attribute class for that service instance. Separate registrations must be made for each language that the service is to be advertised in.

If a SA attempts to register a service with a DA and the registration is larger than the site path MTU, then the DA will reply with a SrvAck, with the error set to INVALID_REGISTRATION and the 'Overflow' byte set.

An example of Service Registration information is:

```
   Lifetime (seconds):   16-bit unsigned integer
   URL (at least):       service:<srvtype>://<addr-spec>
   Attributes (if any):  (ATTR1=VALUE),KEYWORD,(ATTR2 = VAL1, VAL2)
```

In order to offer continuously advertised services, Service Agents should start the reregistration process before the Lifetime they used in the registration expires.

An example of a service registration (valid for 3 hours) is as follows:

```
   Lifetime:   10800
   URL:        service:lpr://igore.wco.ftp.com:515/draft
   Attributes: (SCOPE=DEVELOPMENT),
               (PAPER COLOR=WHITE),
               (PAPER SIZE=LETTER),
               UNRESTRICTED_ACCESS,
               (LANGUAGE=POSTSCRIPT, HPGCL),
               (LOCATION=12 FLOOR)
```

The same registration could be done again, as shown below, in German; however, note that "lpr", "service", and "SCOPE" are reserved terms and will remain in the language they were originally registered (English).

```
   Lifetime:   10800
   URL:        service:lpr://igore.wco.ftp.com:515/draft
   Attributes: (SCOPE=ENTWICKLUNG),
               (PAPIERFARBE=WEISS),
               (PAPIERFORMAT=BRIEF),
               UNBEGRENTZTER_ZUGANG,
               (DRUECKERSPRACHE=POSTSCRIPT,HPGCL),
               (STANDORT=11 ETAGE)
```

Scoped registrations must contain the SCOPE attribute. Unscoped registrations must be registered with all unscoped Directory Agents.

Registrations of a previously registered service are considered an update. If such an attribute registration is performed in a protected scope (see section 16.1), a new Attribute Authentication block must also be included, and the 'A' bit set in the registration message header.

The new registration's attributes replace the previous registration's, but do not effect attributes which were included previously and are not present in the update.

For example, suppose service:x://a.org has been registered with attributes A=1, B=2, C=3. If a new registration comes for service:x://a.org with attributes C=30, D=40, then the attributes for the service after the update are A=1, B=2, C=30, D=40.

In the example above, the SCOPE is set to DEVELOPMENT (in English) and ENTWICKLUNG (in German). Recall that all strings in a message must be in one language, which is specified in the header. The string SCOPE is *not* translated, as it is one of the reserved strings in the Service Location Protocol (see section 17.2.)

The Directory Agent may return a server error in the acknowledgment. This error is carried in the Error Codes field of the service location message header. A Directory Agent MUST decline to register a service if it is specified with an unsupported scope. In this case a SCOPE_NOT_SUPPORTED error is returned in the SrvAck. A Directory Agent MUST NOT accept Service Registrations which have an unsupported scope unless it is an unscoped Directory Agent, in which case it MUST accept all Service Registrations.

An unscoped Service Registration will match all requests. A request which specifies a certain scope will therefore return services which have that scope and services which are unscoped. It is strongly suggested that one should use scopes in all registrations or none. See Sections 16 and 3.7 for details.

When the URL entry accompanying a registration also contains an authentication block (section 4.3), the DA MUST perform the indicated authentication, and subsequently indicate the results in the Service Acknowledgement message.

10. Service Acknowledgement Message Format

A Service Acknowledgement is sent as the result of a DA receiving and processing a Service Registration or Service Deregistration. An acknowledgment indicating success must have the error code set to zero. Once a DA acknowledges a service registration it makes the information available to clients.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Service Location header (function = SrvAck)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Error Code           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Error Code may have one of the following values:

```
   0        Success
```

PROTOCOL_PARSE_ERROR
A DA returns this error when the SrvReg or SrvDereg is received which cannot be parsed or the declared string lengths overrun the message.

INVALID_REGISTRATION
A DA returns this error when a SrvReg or SrvDeReg is invalid. For instance, an invalid URL, unknown or malformed attributes, or deregistering an unregistered service all cause this error to be reported.

SCOPE_NOT_SUPPORTED
A DA which is configured to have a scope will return this error if it receives a SrvReq which is set to have a scope which it does not support.

CHARSET_NOT_UNDERSTOOD
If the DA receives a SrvReg or SrvDereg in a character set which it does not support, it will return this error.

AUTHENTICATION_ABSENT
If DA has been configured to require an authentication for any service registered in the requested scope, and there are no authentication blocks in the registration, the DA will return this error.

AUTHENTICATION_FAILED
If the registration contains an authentication block which fails to match the correct result as calculated (see section 4.3) over the URL or attribute data to be authenticated, the DA will return this error.

If the Directory Agent accpets a Service Registration, and already has an existing entry, it updates the existing entry with the new lifetime information and possibly new attributes and new attribute values. Otherwise, if the registration is acceptable (including all necessary authentication checks) the Directory Agent creates a new entry, and sets the 'F' bit in the Service Acknowledgement returned to the Service Agent.

11. Service Deregister Message Format

When a service is no longer available for use, the Service Agent must deregister itself from Directory Agents that it has been registered with. A service uses the following PDU to deregister itself.

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |           Service Location header (function = SrvDereg)       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         length of URL         |              URL              |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \              URL of Service to Deregister, contd.             \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |             (if present) authentication block .....
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  length of <tag spec> string  |            <tag spec>         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                     <tag spec>, continued                     \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Service Agent should retry this operation if there is no response from the Directory Agent. The Directory Agent acknowledges this operation with a Service Acknowledgment message. Once the Service Agent receives an acknowledgment indicating success, it can assume that the service is no longer advertised by the Directory Agent. The Error Code in the Acknowledgment of the Service Deregistration may have the same values as described in section 10.

The Service Deregister Information sent to the directory agent has the following form:

```
     service:<srvtype>://<addr-spec>
     Attribute tags (if any):  ATTR1,KEYWORD,ATTR2
```

This will deregister the specified attributes from the service information from the directory agent. If no attribute tags are included, the entire service information is deregistered in every language and every scope it was registered in. To deregister the printer from the preceding example, use:

```
      service:lpr://igore.wco.ftp.com:515/draft
```

If the service was originally registered with a URL entry containing a URL authentication block, then the Service Deregistration message header MUST have the 'U' bit set, and the URL entry is then followed by the authentication block, with the authenticator calculated over the URL data, the timestamp, and the length of the authenticator as explained in section 4.3. In this calculation, the lifetime of the URL data is considered to be zero, no matter what the current value for the remaining lifetime of the registered URL.

12. Attribute Request Message Format

The Attribute Request is used to obtain attribute information. The UA supplies a request and the appropriate attribute information is returned.

If the UA supplies only a Service Type, then the reply includes all attributes and all values for that Service Type. The reply includes only those attributes for which services exist and are advertised by the DA or SA which received the Attribute Request. Since different instances of a given service can, and very likely will, have different values for the attributes defined by the Service Type, the User Agent must form a union of all attributes returned by all service Agents. The Attribute information will be used to form Service Requests.

If the UA supplies a URL, the reply will contain service information corresponding to that URL.

Attribute Requests include a 'select clause'. This may be used to limit the amount of information returned. If the select clause is empty, all information is returned. Otherwise, the UA supplies a comma delimited list of attribute tags and keywords. If the attribute or keyword is defined for a service, it will be returned in the Attribute Reply, along with all registered values for that attribute. If the attribute selected has not been registered for that URL or Service Type, the attribute or keyword information is simply not returned.

The Attribute Request message has the following form:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Service Location header (function = AttrRqst)        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |length of prev resp list string|<Previous Responders Addr Spec>|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \         <Previous Responders Addr Spec>, continued            \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         length of URL         |              URL              |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                         URL, continued                        \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |        length of <Scope>      |           <Scope>             |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                      <Scope>, continued                       \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   length of <select-list>     |        <select-list>          |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                   <select-list>, continued                    \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The <Previous Responder Address List> functions exactly as introduced in Section 7. See also Section 20.1.

The URL can take two forms: Either it is simply a Service Type, such as "service:http:", or it can be a URL, such as "service:lpr://igore.wco.ftp.com:515/draft". In the former case, all attributes and the full range of values for each attribute for the Service Type is returned. In the latter case, only the attributes for the service whose URL is defined are returned.

The Scope String is provided so that Attribute Requests for Service Types can be made so that only the Attribute information pertaining to a specific scope will be returned. This field is ignored in the case when a full URL is sent in the Attribute Request. The rules for encoding of the Scope String are given in Section 5.4.

The select list takes the form:

```
 <select-list>  ::= <select-item> |
                    <select-item> ',' <select-list>

 <select-item>  ::= <keyword> | <attr-tag> | <partial-tag> '*'

 <partial-tag>  ::= the partial class name of an attribute
                    If followed by an '*', it matches all class names
                    which begin with the partial tag.  If preceded by a
                    '*' it matches all class names which end with
                    partial tag.  If both preceded and followed by '*'
                    it matches all class names which contain the
                    partial tag.
```

For definitions of <attr-tag> and <keyword> see 5.4.

An example of a select-list following the printer example is:

PAGES PER MINUTE, UNRESTRICTED_ACCESS, LOCATION

If sent to a Directory Agent, the number of previous responders is zero and there are no Previous Responder Address Specification. These fields are only used for repeated multicasting, exactly as for the Service Request.

13. Attribute Reply Message Format

An Attribute Reply Message takes the form:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Service Location header (function = AttrRply)        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Error Code            |  length of <attr-list> string |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                          <attr-list>                          \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Error Code may have the following values:

```
   0        Success
```

LANGUAGE_NOT_SUPPORTED
A SA or DA returns this when a request is received from a UA which is in a language for which there is no registered Service Information and the request arrived with the Monolingual bit set. See Section 17.

PROTOCOL_PARSE_ERROR
A DA or SA returns this error when a AttrRqst is received which cannot be parsed or the declared string lengths overrun the message.

SCOPE_NOT_SUPPORTED
A DA which is configured to have a scope will return this error if it receives an AttrRqst which is set to have a scope which it does not support. SAs will silently discard multicast AttrRqst messages for scopes they do not support.

CHARSET_NOT_UNDERSTOOD
If the DA receives an AttrRqst in a character set which it does not support, it will return this error. SAs will silently discard multicast AttrRqst messages which arrive using character sets they do not support.

The <attr-list> (attribute list) has the same form as the attribute list in a Service Registration, see Section 20.3 for a formal definition of this field.

An Attribute Request for "lpr" might elicit the following reply (UNRESTRICTED_ACCESS is a keyword):

```
(PAPER COLOR=WHITE,BLUE),
(PAPER SIZE=LEGAL,LETTER,ENVELOPE,TRACTOR FEED),
UNRESTRICTED_ACCESS,
(PAGES PER MINUTE=1,3,12),
(LOCATION=12th, NEAR ARUNA'S OFFICE),
(QUEUES=LEGAL,LETTER,ENVELOPE,LETTER HEAD)
```

If the message header has the 'A' bit set, the Attribute Reply will have an Attribute Authentication block set. In this case, the Attribute Authenticator must be returned with the entire list of attributes, exactly as it was registered by an SA in a protected scope. In this case, the URL was registered in a protected scope and the UA included a URL but not a select clause. If the AttrRqst specifies that only certain attributes are to be returned, the DA does not (typically cannot) compute a new Authenticator so it simply returns the attributes without an authenticator block.

A UA which wishes to obtain authenticated attributes for a service in a protected scope MUST therefore must include a particular URL and no select list with the AttrRqst.

14. Directory Agent Advertisement Message Format

Directory Agent Advertisement Messages have the following format:

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |          Service Location header (function = DAAdvert)        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |           Error Code          |          Length of URL        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                              URL                              \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |     Length of <Scope-list>    |           <Scope-list>        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  \                    <Scope-list>, continued                    \
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Error Code is set when a DA Advertisement is returned as the result of a Service Request. It will always be set to 0 in the case of an unsolicited DA Advertisement. The Error Code may take the values specified in Section 6.

The URL corresponds to the Directory Agent's location. The <Scope-list> is a comma delimited list of scopes which the DA supports, in the following format:

```
    <Scope-list>    ::=    <Scope> | <Scope-list> ',' <Scope>
    <Scope>         ::=    String representing a scope
```

See Section 5.4 for the lexical rules regarding <Scope>.

DA Advertisements sent in reply to a Directory Agent Discovery Request has the same format as the unsolicited DA Advertisement, for example:

```
   URL:        service:directory-agent://SLP-RESOLVER.CATCH22.COM
   SCOPE List: ADMIN
```

The Directory Agent can be reached at the Address Specification returned, and supports the SCOPE called "ADMIN".

15. Directory Agents

15.1. Introduction

A Directory Agent acts on behalf of many Service Agents. It acquires information from them and acts as a single point of contact to supply that information to User Agents.

The queries that a User Agent multicasts to Service Agents (in an environment without a Directory Agent) are the same as queries that the User Agent might unicast to a Directory Agent. A User Agent may cache information about the presence of alternate Directory Agents to use in case a selected Directory Agent fails.

Aside from enhancing the scalability of the protocol (see section 3.7), running multiple DAs provides robustness of operation. The DAs may have replicated service information which remain accessible even when one of the DAs fail. Directory Agents, in the future, may use mechanisms outside of this protocol to coordinate the maintenance of a distributed database of Service Location information, and thus scale to enterprise networks or larger administrative domains.

Each Service Agent must register with all DAs they are configured to use. UAs may choose among DAs they are configured to use.

Locally, Directory Agent consistency is guaranteed using mechanisms in the protocol. There isn't any Directory to Directory Agent protocol yet. Rather, passive detection of DAs by SAs ensures that eventually service information will be registered consistently between DAs. Invalid data will age out of the Directory Agents leaving only transient stale registrations even in the case of a failure of a Service Agent.

15.2. Finding Directory Agents

A User or Service Agent may be statically configured to use a particular DA. This is discouraged unless the application resides on a network where any form of multicast or broadcast is impossible.

Alternatively, a host which uses DHCP [2, 12] may use it to obtain a Directory Agent's address. DHCP options 78 and 79 have been assigned for this purpose [22].

The third way to discover DAs is dynamically. This is done by sending out a Directory Agent Discovery request (see Section 5.2).

Lastly, the agent may be informed passively as follows:

When a Directory Agent first comes on-line it sends an unsolicited DA Advertisement to the Service Location general multicast address. If a DA supports a particular scope or set of scopes these are placed in the reply. The class for this attribute is 'SCOPE'.

Every CONFIG_INTERVAL_9 a Directory Agent will send an unsolicited DA Advertisement. This will ensure that eventually it will be discovered by all applications which are concerned.

When a Directory Agent first comes up it begins with 0 as its XID, and increments this by one each time it sends an unsolicited DA Advertisement. When the counter wraps, it should go from 0xFFFF to 0x0100, not 0.

If the Directory Agent has stored all of the service information in a nonvolatile store, it should initially set the XID to 0x100, as it is not coming up 'stateless.' If it stores service registrations in memory only, it will restart without any state. It should indicate this by resetting its XID to 0.

All Service Agents which receive the unsolicited DA Advertisement should examine its XID. If the Directory Agent has never before been heard from or if the XID is less than it was previously and less than 256, the Service Agent should assume the DA does not have its service registration, even if it once did. If this is the case and the DA has the proper scope, the SA should register all service information with the Directory Agent, after waiting a random interval CONFIG_INTERVAL_10.

When a Service Agent or User Agent first comes on-line it must issue a Directory Agent Discovery Request unless it is using static or DHCP configuration, as described in 5.2.

A Service Agent registers information with ALL newly discovered Directory Agents when either of the above two events take place. When scopes are being used, a Service Agent SHOULD choose a set of scopes to be advertised in and need only register with Directory Agents that support the scopes in which they wish to be registered. Services MUST be registered with DAs that support their scope and those which have no scope, unless specifically configured not to do so (see section 22.1.)

Once a User Agent becomes aware of a Directory Agent it will unicast its queries there. In the event that more than one Directory Agent is detected, it will select one to communicate with. When scopes are supported, the User Agent will direct its queries to different Directory Agents depending on which scopes are appropriate domains for the query to be answered in.

The protocol will cause all DAs (of the same scope) to eventually obtain consistent information. Thus one DA should be as good as any other for obtaining service information. There may be temporary inconsistencies between DAs.

16. Scope Discovery and Use

The scope mechanism in the Service Location Protocol enhances its scalability. The primary use of scopes is to provide the capability to organize a site network along administrative lines. A set of services can be assigned to a given department of an organization, to a certain building or geographical area or for a certain purpose. The users of the system can be presented with these organizational elements as a top level selection, before services within this domain are sought.

A site network that has grown beyond a size that can be reasonably serviced by a few DAs can use the scope mechanism. DAs have the attribute class "SCOPE". The values for this attribute are a list of strings that represent the administrative areas for which this Directory Agent is configured. The semantics and language of the strings used to describe the scope are almost entirely the choice of the administrative entity of the particular domain in which these scopes exist. The values of SCOPE should be configurable, so the system administrator can set its value. The scopes "LOCAL" and "REMOTE" are reserved and SHOULD NOT be used. Use of these reserved values is to be defined in a future protocol document.

Services with the attribute SCOPE should only be registered with DAs which support the same scope or DAs which have no scope.

Directory Agents advertise their available scopes. A Service Agent may then choose a scope in which to register, and SHOULD register with all Directory Agents in that scope, as well as all DAs which have no scope. Failure to be comprehensive in registration according to this rule will mean that the service advertisement may not be available to all User Agents.

A Directory Agent which has a scope will return advertisements in response to Directory Agent Discovery requests with the scope

information included. Note that the "service:directory-agent" scheme is registered with the IANA naming authority (which is automatically selected by leaving the Naming Authority field empty.)

The query:

```
      directory-agent/MATH DEPT//
```

Could receive the following DA Advertisement:

```
   Returned URL:        service:directory-agent://diragent.blah.edu
   Returned SCOPE:      MATH DEPT
```

The same Directory Agent if it had no scope value would reply:

```
   Returned URL:        service:directory-agent://diragent.void.com
   Returned SCOPE:
```

If a Directory Agent supported more than one scope it would reply as:

```
   Returned URL:        service:directory-agent://srv.domain.org
   Returned SCOPE:      MATH DEPT,ENGLISH DEPT,CS DEPT
```

A DA which has no scope will reply to any Directory Agent Discovery Request.

Being a member of a scope means that an agent SHOULD use those Directory Agents that support its scope. User Agents send all requests to DAs which support the indicated scope. Services are registered with the DA(s) in their scope. For a UA to find a service that is registered in a particular scope it must send requests to a DA which supports the indicated scope. There is no limitation on scope membership built into the protocol; that is to say, a User Agent or Service Agent may be a member of more than one scope. Membership is open to all, unless some external authorization mechanism is added to limit access.

16.1. Protected Scopes

Scope membership MAY also define the security access and authorization for services in the scope; such scopes are called protected scopes. If a User Agent wishes to be sure that Service Agents are authorized to provide the service they advertise, then the User Agent should request services from a protected scope which has been configured to have the necessary authentication mechanism and keys distributed to the Service Agents within the scope. A directory agent distributing URLs for services in a protected scope

will reject any registrations or deregistrations for service agents which cannot provide cryptographically strong authentication to prove their authorization to provide the services.

For instance, if a campus registrar wishes to find a working printer to produce student grade information for mailing, the registrar would require the printing user agent to transmit the printable output only to those printing Service Agents which have been registered in the appropriate protected scope. Notice that each service agent is, under normal circumstances, validated two times: once when registering with the directory agent, and once when the user agent validates the URL received with the Service Reply. This protects against the possibilities of malicious Directory Agents as well as malicious Service Agents.

Note that services in protected scopes provide separate authentication for their URL entry, and for their attributes. This follows naturally from the needs of the protocol operation. User Agents which specify a service type and attributes needed for service in that service type will not receive attribute information from the directory agent; they will only receive the appropriate URL entries. Only the information returned needs to be authenticated.

User agents which receive attribute information for a particular URL (see section 12), on the other hand, need to authenticate the attributes when they are returned (see section 13). In this case, there may be much more data to authenticate, but this operation is also performed much less often, usually only while the user is browsing the available network resources.

17. Language and Character Encoding Issues

All Service Registrations declare the language in which the strings in the service attributes are written by specifying the appropriate code in the message header. For each language the Service advertises a separate registration takes place. Each of these registrations uses the same URL to indicate that they refer to the same service.

If a Service is fully deregistered (the URL is given in the Service Deregister request, without any attribute information) then the Service needs to be deregistered only once. This will effectively deregister the service in all languages it has been registered in.

If, on the other hand, attribute information is included in the Service Deregistration request, a separate Service Deregistration of selected attributes must be undertaken in each language in which service information has been provided to the DA by a Service Agent.

Service Registrations in different languages are mutually unintelligible. They share no information except for their service type and URL with which they were registered. No attempt is made to match queries with "language independence." Instead, queries are handled using string matching against registrations in the same language as the query.

Service Types which are standardized will have definitions for all attributes and value strings. Official translations to other languages of the attribute tags and values may be created and submitted as part of the standard; this is not feasible for all languages. For those languages which are not defined as part of the Service Type, a best effort translation of the standard definitions of the Service type's attribute strings MAY be used.

All Service Requests specify a requested language in the message header. The Directory Agent or Service Agent will respond in the same language as the request, if it has a registration in the same language as the request. If this language is not supported, and the Monolingual bit is not specified, a reply can be sent in the default language (which is English.) If the 'monolingual bit' flag in the header is set and the requested language is not supported, a SrvRply is returned with the error field set to LANGUAGE_NOT_SUPPORTED.

If a query is in a supported language on a SA or DA, but has a different dialect than the available service information, the query MUST be serviced on a best-effort basis. If possible, the query should be matched against the same dialect. If that is not possible, it MAY be matched against any dialect of the same language.

17.1. Character Encoding and String Issues

Values for character encoding can be found in IANA's database
      http://www.isi.edu/in-notes/iana/assignments/character-sets
and have the values referred by the MIBEnum value.

The encoding will determine the interpretation of all character data which follows the Service Location Protocol header. There is no way to mix ASCII and UNICODE, for example. All responses must be in the character set of the request, or use US-ASCII. If a request is sent to a DA or SA or a registration is sent to a DA, which is unable to manipulate or store the character set of the incoming message, the request will fail. The SA or DA returns a CHARSET_NOT_UNDERSTOOD error in a SrvAck message in this case. Requests using US-ASCII will never fail for this reason, since all SAs and DAs must be able to accept this character set.

Certain characters are illegal in certain contexts of the protocol. Since the protocol is largely character string based, in some contexts characters are used as protocol delimiters. In these cases the delimiting characters must not be used as 'data text.'

17.1.1. Substitution of Character Escape Sequences

The Service Location Protocol has an 'escape mechanism' which is consistent with HTTP 2.0 [5] and SGML [16]. If the character sequence "&#" is followed by one or more digits, followed by a semicolon ';' the entire sequence is interpreted as a single character. The digits are interpreted as a decimal value in the character set of the request, as specified by the header. Thus, in US-ASCII &#44; would be interpreted as a comma. Substitution of these escape strings must be done in all <attr-list> and strings present in SrvReq and AttrRqst messages. Only numerical character references are accepted, not 'Entity References,' as defined in HTML. These escape values should only be used to provide a mechanism for including reserved characters in attribute tag and value strings.

The interpretation of these escape values is different than in HTML in one respect: In HTML the escape values are considered to be in the ISO Latin-1 character set. In Service Location they are interpreted in the character set defined in the header of the message.

This escape mechanism allows characters like commas to be included in attribute tags and values, which would otherwise be illegal as the comma is a protocol delimiter.

Attribute tags and values of different languages are considered to be mutually unintelligible. A query in one language SHOULD use service information registered in that language.

17.2. Language-Independent Strings

Some strings, such as Service Type names, have standard definitions. These strings should be considered as tokens and not as words in a language to be translated.

Reserved String	Section	xDefinition
SCOPE	3, 15	Used to limit the matching of requests.
SERVICE	6, 9	The URL scheme of all Service Location information registered with a DA or returned from a Service Request.

<srvtype>	20.2.1	Used in all service registrations and replies.
domain names	20.4	A fully qualified domain name, used in registrations and replies.
IANA	3.3	The default naming authority.
LOCAL	16	Reserved.
REMOTE	16	Reserved.
TRUE	20.5	Boolean true.
FALSE	20.5	Boolean false.

18. Service Location Transactions

18.1. Service Location Connections

When a Service Location Request or Attribute Request results in a UDP reply from a Service or Directory Agent that will overflow a datagram, the User Agent can open a connection to the Agent and reissue the request over the connection. The reply will be returned with the overflow bit set (see section 4). The reply will contain as much data as will fit into a single datagram. If no MTU information is available for the route, assume that the MTU is 1400; this value is configurable (see section 22).

When a request results in overflowed data that cannot be correctly parsed (say, because of duplicate or dropped IP datagrams), a User Agent that wishes to reliably obtain the overflowed data must establish a TCP connection with the Directory Agent or Service Agent with the data. When the request is sent again with a new XID, the reply is returned over the connection.

When registration data exceeds one datagram in length, the Service Registration should be made by establishing a connection with a Directory Agent and sending the registration over the connection stream.

Directory Agents and Service Agents must respond to connection requests; services whose registration data can overflow a datagram must be able to use TCP to send the registration. User Agents should be able to make Service and Attribute Requests using TCP. If they fail to implement this, they must be able to interpret partial replies and/or reissue requests with more selective criteria to reduce the size of the replies.

A connection initiated by an Agent may be used for a single transaction. It may also be used for multiple transactions. Since there are length fields in the message headers, the Agents may send

multiple requests along a connection and read the return stream for acknowledgments and replies.

The initiating agent is responsible for closing the TCP connection. The DA should wait at least CONFIG_INTERVAL_12 before closing an idle connection. DAs and SAs SHOULD eventually close idle connections to ensure robust operation, even when the agent which opened a connection neglects to close it.

18.2. No Synchronous Assumption

There is no requirement that one transaction complete before a given host begins another. An agent may have multiple outstanding transactions, initiated either using UDP or TCP.

18.3. Idempotency

All Service Location actions are idempotent. Of course registration and deregistration will change the state of a DA, but repeating these actions with the same XID will have exactly the same effect each time. Repeating a registration with a new XID has the effect of extending the lifetime of the registration.

19. Security Considerations

The Service Location Protocol provides for authentication of Service Agents as part of the scope mechanism, and consequently, integrity of the data received as part of such registrations. Service Location does not provide confidentiality. Because the objective of this protocol is to advertise services to a community of users, confidentiality might not generally be needed when this protocol is used in non-sensitive environments. Specialized schemes might be able to provide confidentiality, if needed in the future. Sites requiring confidentiality should implement the IP Encapsulating Security Payload (ESP) [3] to provide confidentiality for Service Location messages.

Using unprotected scopes, an adversary might easily use this protocol to advertise services on servers controlled by the adversary and thereby gain access to users' private information. Further, an adversary using this protocol will find it much easier to engage in selective denial of service attacks. Sites that are in potentially hostile environments (e.g. are directly connected to the Internet) should consider the advantages of distributing keys associated with

protected scopes prior to deploying the sensitive directory agents or service agents.

Service Location is useful as a bootstrap protocol. It may be used in environments in which no preconfiguration is possible. In such situations, a certain amount of "blind faith" is required: Without any prior configuration it is impossible to use any of the security mechanisms described above. Service Location will make use of the mechanisms provided by the Security Area of the IETF for key distribution as they become available. At this point it would only be possible to gain the benefits associated with the use of protected scopes if some cryptographic information can be preconfigured with the end systems before they use Service Location. For User Agents, this could be as simple as supplying the public key of a Certificate Authority. See Appendix B.

20. String Formats used with Service Location Messages

The following section supplies formal definitions for fields and protocol elements introduced in the sections indicated.

Protocol Element	Defined in	Used in
<Previous Responders' Addr Spec>	20.1	SrvReq
Service Request <predicate>	5.4	SrvReq
URL	20.2	SrvReg, SrvDereg, SrvRply
<attr-list>	20.3	SrvReg, SrvRply, AttrRply
<Service Registration Information>	9	SrvReg
<Service Deregister Information>	11	SrvDereg
<Service Type String>	20.2.1	AttrRqst

20.1. Previous Responders' Address Specification

The previous responders' Address Specification is specified as

```
<Previous Responders' Address Specification> ::=
       <addr-spec> |
       <addr-spec>, <Previous Responders' Address Specification>
```

i.e., a list separated by commas with no intervening white space. The Address Specification is the address of the Directory Agent or Service Agent which supplied the previous response. The format for Address Specifications in Service Location is defined in section 20.4. The comma delimiter is required between each <addr-spec>. The use of dotted decimal IP address notation should only be used in environments which have no Domain Name Service.

Example:

```
RESOLVO.NEATO.ORG,128.127.203.63
```

20.2. **Formal Definition of the ``service:''** Scheme

A URL with a ``service:'' scheme is used in the SrvReg, SrvDereg, SrvRply and AttrRqst messages in Service Location. URLs are defined in RFC 1738 [6]. A URL with the ``service:'' scheme must contain at least:

```
<url> ::= service:<srvtype>://<addr-spec>
```

where:

service	the URL scheme for Service Location, to return Replies.
<srvtype>	a string; Service Types may be standardized by developing a specification for the "service type"-specific part and registering it with IANA. See sections 20.2.1 and 3.3.
<addr-spec>	the service access point of the service. It is the network address or domain name where the service can be accessed. See section 20.4.

The ``service:'' scheme may be followed by any legal URL. The 'minimal' service URL provides a service type and an access point for a particular service. The protocol used to access the service at the given service access <addr-spec> may be implicit in the Service Type name. If this is not the case, the Service Type MUST be defined in such a way that attribute information will include all necessary configuration and protocol information. A User Agent MUST therefore be able to use either a ``service:''  URL alone or a ``service:'' URL in conjunction with service attributes to make use of a service.

20.2.1. Service Type String

The Service Type is a string describing the type of service. These strings may only be comprised of alphanumeric characters, '+', and '-'. Upper case is considered equivalent to lower case in Service Type names.

If the Service Type name is followed by a '.' and a string (which has the same limitations) the 'suffix' is considered to be the Naming Authority of the service. If the Naming Authority is omitted, IANA is assumed to be the Naming Authority.

Service Types developed for in-house or experimental use may have any name and attribute semantics provided that they do not conflict with the standardized Service Types.

20.3. Attribute Information

The <attr-list> is returned in the Attribute Reply if the Attribute Request does not result in an empty result.

```
<attr-list> ::= <attribute> | <attribute>, <attr-list>
<attribute> ::= (<attr-tag>=<attr-val-list>) | <keyword>
<attr-val-list> ::= <attr-val> | <attr-val>, <attr-val-list>
```

An <attr-list> must be scanned prior to evaluation for all occurrences of the string "&#" followed by one or more digit followed by ';'. See Section 17.1.1.

A keyword has only an <attr-tag>, and no values.

A comma cannot appear in an <attr-val>, as the comma is used as the multiple value delimiter. Examples of an <attr-list> are:

```
(SCOPE=ADMINISTRATION)
(COLOR=RED, WHITE, BLUE)
(DELAY=10 MINS),BUSY,(LATEST BUILD=10-5-95),(PRIORITY=L,M,H)
```

The third example has three attributes in the list. Color can take on the values red, white and blue. There are several other examples of replies throughout the document.

20.4. Address Specification in Service Location

The address specification used in Service Location is:

```
<addr-spec> ::= [<user>:<password>@]<host>[:<port>]

<host>      ::= Fully qualified domain name |
                dotted decimal IP address notation
```

When no Domain Name Server is available, SAs and DAs must use dotted decimal conventions for IP addresses. Otherwise, it is preferable to use a fully qualified domain name wherever possible as renumbering of host addresses will make IP addresses invalid over time.

Generally, just the host domain name (or address) is returned. When there is a non-standard port for the protocol, that should be returned as well. Some applications may make use of the <user>:<password>@ syntax, but its use is not encouraged in this context until mechanisms are established to maintain confidentiality.

Address specification in Service Location is consistent with standard URL format [6].

20.5. Attribute Value encoding rules

Attribute values, and attribute tags are CASE INSENSITIVE for purposes of lexical comparison.

Attribute values are strings containing any characters with the exception of '(', ')', '=', '>', '<', '/', '*', and ',' (the comma) except in the case described below where opaque values are encoded. These characters may be included using the character value escape mechanism described in section 17.1.1.

While an attribute can take any value, there are three types of values which differentiate themselves from general strings: Booleans, Integers and Opaque values.

- Boolean values are either "TRUE" or "FALSE". This is the case regardless of the language (i.e. in French or Telugu, Boolean TRUE is "TRUE", as well as in English.) Boolean attributes can take only one value.

- Integer values are expressed as a sequence of numbers. The range of allowable values for integers is "-2147483648" to "2147483647". No other form of numeric representation is interpreted as such except integers. For example, hexadecimal numbers such as "0x342" are not interpreted as integers, but as strings.

- Opaque values (i.e. binary values) are expressed in radix-64 notation. The syntax is:

```
     <opaque-val>    ::=  (<len>:<radix-64-data>)
     <len>           ::=  number of bytes of the original data
     <radix-64-data> ::=  radix-64 encoding of the original data
```

  <len> is a 16-bit binary number. Radix-64 encodes every 3 bytes of binary data into 4 bytes of ASCII data which is in the range of characters which are fully printable and transferable by mail. For a formal definition of the Radix-64 format see RFC 1521 [7], MIME Part One, Section 5.2 Base64 Content Transfer Encoding, page 21.

21. Protocol Requirements

In this section are listed various protocol requirements for User Agents, Service Agents, and Directory Agents.

21.1. User Agent Requirements

A User Agent MAY:

- Provide a way for the application to configure the default DA, so that it can be used without needing to find it each initially.

- Be able to request the address of a DA from DHCP, if configured to do so.

- Ignore any unauthenticated Service Reply.

- Be able to issue requests in any language or character set provided that it can switch to the default language and character set if the request can not be serviced by DAs and SAs at the site.

- Require an authentication block in any URL entry returned as part of a Service Request, before making use of the advertised service.

A User Agent SHOULD:

- Try to contact DHCP to obtain the address of a DA.

- Use a scope in all requests, if possible.

- Issue requests to scoped DAs if the UA has been configured with a scope.

- Listen on the Service Location General Multicast address for unsolicited DA Advertisements. This will increase the set of Directory Agents available to it for making requests. See Section 15.2.

- Be able to be configured to require an authentication block in any received URL entry advertised as belonging to a protected scope, before making use of the service.

If the UA does not listen for DA Advertisements, new DAs will not be passively detected. A UA which does not have a configured DA and has not yet discovered one and is not listening for unsolicited DA Advertisements will remain ignorant of DAs. It may then do a DA discovery before each query performed or it may simply use multicast queries to Service Agents.

A User Agent MUST:

- Be able to unicast requests and receive replies from a DA. Transactions should be made reliable by using retransmission of the request if the reply does not arrive within a timeout interval.

- Be able to detect DAs using a Directory Agent Discovery request issued when the UA starts up.

- Be able to send requests to a multicast address. Service Specific Multicast addresses are computed based on a hash of the Service Type. See Section 3.6.2.

- Be able to handle numerous replies after a multicast request. The implementation may be configurable so it will either return the first reply, all replies until a timeout or keep trying till the results converge.

- Ignore any unauthenticated Service Reply or Attribute Reply when an appropriate IPSec Security Association for that Reply exists.

- Whenever it obtains its IP address from DHCP in the first place, also attempt to obtain scope information, and the address of a DA, from DHCP.

- Use the IP Authentication Header or IP Encapsulating Payload in all Service Location messages, whenever an appropriate IPSec Security Association exists.

- Be able to issue requests using the US-ASCII character set.

- If configured to use a protected scope, be able to use "md5WithRSAEncryption" [4] to verify the signed data.

21.2. Service Agent Requirements

A Service Agent MAY be able to:

- Get the address of a local Directory Agent by way of DHCP.

- Accept requests in non-US-ASCII character encodings. This is encouraged, especially for UNICODE [1] and UTF-8 [25] encodings.

- Register services with a DA in non-US-ASCII character encodings. This is encouraged, especially for UNICODE [1] and UTF-8 [25] encodings.

A Service Agent SHOULD be able to:

- Listen to the service-specific multicast address of the service it is advertising. The incoming requests should be filtered: If the Address Specification of the SA is in the Previous Responders Address Specification list, the SA SHOULD NOT respond. Otherwise, a response to the multicast query SHOULD be unicast to the UA which sent the request.

- Listen for and respond to broadcast requests and TCP connection requests, to the Service Location port.

- Be configurable to calculate authentication blocks and thereby be enabled to register in protected scopes. This requires that the service agent be configured to possess the necessary keys to calculate the authenticator.

A Service Agent MUST be able to:

- Listen to the Service Location General Multicast address for queries (e.g., Service Type Requests). If the query can be replied to by the Service Agent, the Service Agent MUST do so. It MUST check first to make sure it is not on the list of 'previous responders.'

- Listen to the Service Location General Multicast address for unsolicited DA Advertisements. If one is detected, and the DA has the right scope, (or has no scope), all services which are currently being advertised MUST be registered with the DA (unless configured to only use a single DA (see section 22.1), or the DA has already been detected, subject to certain rules (see section 15.2)).

- Whenever it obtains its IP address from DHCP in the first place, also attempt to obtain scope information, and the address of a DA, from DHCP.

- Unicast registrations and deregistrations to a DA. Transactions should be made reliable by using retransmission of the request if the reply does not arrive within a timeout interval.

- Be able to detect DAs using a Directory Agent Discovery request issued when the SA starts up (unless configured to only use a single DA, see section 22.1.)

- Use the IP Authentication Header or IP Encapsulating Payload in all Service Location messages, whenever an appropriate IPSec Security Association exists.

- Be able to register service information with a DA using US-ASCII character encoding. It must also be able to reply to requests from UAs which use US-ASCII character encoding.

- Reregister with a DA before the Lifetime of registered service information elapses.

- If configured to use a protected scope, be able to use "md5WithRSAEncryption" [4] to produce the signed data.

21.3. Directory Agent Requirements

A Directory Agent MAY:

- Accept registrations and requests in non-US-ASCII character encodings. This is encouraged, especially for UNICODE [1] and UTF-8 [25] encodings.

A Directory Agent SHOULD:

- Be able to configure certain scopes as protected scopes, so that registrations within those scopes require the calculation of cryptographically strong authenticators. This requires that the DA be able to possess the keys needed for the authentication, or that the DA be able to acquire a certificate generated by a trusted Certificate Authority [24], before completing Service Registrations for protected scopes.

A Directory Agent MUST be able to:

- Send an unsolicited DA Advertisements to the Service Location General Multicast address on startup and repeat it periodically. This reply has an XID which is incremented by one each time. If the DA starts with state, it initializes the XID to 0x0100. If it starts up stateless, it initializes the XID to 0x0000.

- Ignore any unauthenticated Service Registration or Service Deregistration from an entity with which it maintains a security association.

- Listen on the Directory Agent Discovery Multicast Address for Directory Agent Discovery requests. Filter these requests if the Previous Responder Address Specification list includes the DA's Address Specification.

- Listen for broadcast requests to the Service Location port.

- Listen on the TCP and UDP Service Location Ports for unicast requests, registrations and deregistrations and service them.

- Provide a way in which scope information can be used to configure the Directory Agent.

- Expire registrations when the service registration's lifetime expires.

- When a Directory Agent has been configured with a scope, it MUST refuse all requests and registrations which do not have this scope. The DA replies with a SCOPE_NOT_SUPPORTED error. There is one exception: All DAs MUST respond to DA discovery requests which have no scope.

- When a Directory Agent has been configured without a scope, it MUST accept ALL registrations and requests.

- Ignore any unauthenticated Service Location messages when an appropriate IPSec Security Association exists for that request.

- Use the IP Authentication and IP Encapsulating Security Payload in Service Location messages whenever an appropriate IPSec Security Association exists.

- Accept requests and registrations in US-ASCII.

- If configured with a protected scope, be able to authenticate (at least by using "md5WithRSAEncryption" [4]) Service Registrations advertising services purporting to belong to such configured protected scopes.

22. Configurable Parameters and Default Values

There are several configuration parameters for Service Location. Default values are chosen to allow protocol operation without the need for selection of these configuration parameters, but other values may be selected by the site administrator. The configurable parameters will allow an implementation of Service Location to be more useful in a variety of scenarios.

Multicast vs. Broadcast

All Service Location entities must use multicast by default. The ability to use broadcast messages must be configurable for UAs and SAs. Broadcast messages are to be used in environments where not all Service Location

entities have hardware or software which supports multicast.

Multicast Radius

Multicast requests should be sent to all subnets in a site. The default multicast radius for a site is 32. This value must be configurable. The value for the site's multicast TTL may be obtained from DHCP using an option which is currently unassigned.

Directory Agent Address

The Directory Agent address discovery mechanism must be configurable. There are three possibilities for this configuration: A default address, no default address and the use of DHCP to locate a DA as described in section 15.2. The default value should be use of DHCP, with "no default address" used if DHCP does not respond. In this case the UA or SA must do a Directory Agent Discovery query.

Directory Agent Scope Assignment

The scope or scopes of a DA must be configurable. The default value for a DA is to have no scope if not otherwise configured.

Path MTU

The default path MTU is assumed to be 1400. This value may be too large for the infrastructure of some sites. For this reason this value MUST be configurable for all SAs and DAs.

Keys for Protected Scopes

If the local administration designates certain scopes as "protected scopes", the agents making use of those scopes have to be able to acquire keys to authenticate data sent by services along with their advertised URLs for services within the protected scope. For instance, service agents would use a private key to produce authentication data. By default, service agents use "md5WithRSAEncryption" [4] to produce the signed data, to be be included with service registrations and deregistrations (see appendix B, 4.3). This authentication data could be verified by user agents and directory agents that possess the corresponding public key.

22.1. Service Agent: Use Predefined Directory Agent(s)

A Service Agent's default configuration is to do passive and active DA discovery and to register with all DAs which are properly scoped.

A Service Agent SHOULD be configurable to allow a special mode of operation: They will use only preconfigured DAs. This means they will *NOT* actively or passively detect DAs.

If a Service Agent is configured this way, knowledge of the DA must come through another channel, either static configuration or by the use of DHCP.

The availability of the Service information will not be consistent between DAs. The mechanisms which achieve eventual consistency between DAs are ignored by the SA, so their service information will not be distributed. This leaves the SA open to failure if the DA they are configured to use fails.

22.2. Time Out Intervals

These values should be configurable in case the site deploying Service Location has special requirements (such as very slow links.)

Interval name	Section	Default Value	Meaning
CONFIG_INTERVAL_0	4.1	1 minute	Cache replies by XID.
CONFIG_INTERVAL_1	4.4	10800 seconds (ie. 3 hours)	registration Lifetime, after which ad expires
CONFIG_INTERVAL_2	5	each second, backing off gradually	Retry multicast query until no new values arrive.
CONFIG_INTERVAL_3	5	15 seconds	Max time to wait for a complete multicast query response (all values.)
CONFIG_INTERVAL_4	9	3 seconds	Wait to register on reboot.
CONFIG_INTERVAL_5	5.2	3 seconds	Retransmit DA discovery, try it 3 times.
CONFIG_INTERVAL_6	5.2	5 seconds	Give up on requests sent to a DA.
CONFIG_INTERVAL_7	5.2	15 seconds	Give up on DA discovery
CONFIG_INTERVAL_8	5.1	15 seconds	Give up on requests sent to SAs.
CONFIG_INTERVAL_9	15.2	3 hours	DA Heartbeat, so that SAs passively detect new DAs.
CONFIG_INTERVAL_10	15.2	1-3 seconds	Wait to register services

			on passive DA discovery.
CONFIG_INTERVAL_11	9	1-3 seconds	Wait to register services on active DA discovery.
CONFIG_INTERVAL_12	18.1	5 minutes	DAs and SAs close idle connections.

A note on CONFIG_INTERVAL_9: While it might seem advantageous to have frequent heartbeats, this poses a significant risk of generating a lot of overhead traffic. This value should be kept high to prevent routine protocol operations from using any significant bandwidth.

23. Non-configurable Parameters

IP Port number for unicast requests to Directory Agents:

UDP and TCP Port Number:	427

Multicast Addresses

Service Location General Multicast Address:	224.0.1.22
Directory Agent Discovery Multicast Address:	224.0.1.35

A range of 1024 contiguous multicast addresses for use as Service Specific Discovery Multicast Addresses will be assigned by IANA.

Error Codes:

No Error	0
LANGUAGE_NOT_SUPPORTED	1
PROTOCOL_PARSE_ERROR	2
INVALID_REGISTRATION	3
SCOPE_NOT_SUPPORTED	4
CHARSET_NOT_UNDERSTOOD	5
AUTHENTICATION_ABSENT	6
AUTHENTICATION_FAILED	7

24. Acknowledgments

This protocol owes some of the original ideas to other service location protocols found in many other networking protocols. Leo McLaughlin and Mike Ritter (Metricom) provided much input into early version of this document. Thanks also to Steve Deering (Xerox) for providing his insight into distributed multicast protocols. Harry Harjono and Charlie Perkins supplied the basis for the URL based

wire protocol in their Resource Discovery Protocol. Thanks also to
Peerlogic, Inc. for supporting this work.

A. Appendix: Technical contents of ISO 639:1988 (E/F): "Code for the representation of names of languages"

Two-letter lower-case symbols are used. The Registration Authority for ISO 639 [15] is Infoterm, Osterreiches Normungsinstitut (ON), Postfach 130, A-1021 Vienna, Austria. Contains additions from ISO 639/RA Newsletter No.1/1989

aa	Afar
ab	Abkhazian
af	Afrikaans
am	Amharic
ar	Arabic
as	Assamese
ay	Aymara
az	Azerbaijani
ba	Bashkir
be	Byelorussian
bg	Bulgarian
bh	Bihari
bi	Bislama
bn	Bengali; Bangla
bo	Tibetan
br	Breton
ca	Catalan
co	Corsican
cs	Czech
cy	Welsh
da	Danish
de	German
dz	Bhutani
el	Greek
en	English
eo	Esperanto
es	Spanish
et	Estonian
eu	Basque
fa	Persian
fi	Finnish
fj	Fiji
fo	Faeroese
fr	French
fy	Frisian
ga	Irish
gd	Scots Gaelic
gl	Galician
gn	Guarani
gu	Gujarati
ha	Hausa
he	Hebrew
hi	Hindi
hr	Croatian
hu	Hungarian
hy	Armenian
ia	Interlingua
in	Indonesian
ie	Interlingue
ik	Inupiak
is	Icelandic
it	Italian
ja	Japanese
jw	Javanese
ka	Georgian
kk	Kazakh
kl	Greenlandic
km	Cambodian
rw	Kinyarwanda
kn	Kannada
ko	Korean
ks	Kashmiri
ku	Kurdish
ky	Kirghiz
la	Latin
ln	Lingala
lo	Laothian
lt	Lithuanian
lv	Latvian, Lettish
mg	Malagasy
mi	Maori
mk	Macedonian
ml	Malayalam
mn	Mongolian
mo	Moldavian
mr	Marathi
ms	Malay
mt	Maltese
my	Burmese
na	Nauru
ne	Nepali
nl	Dutch
no	Norwegian
oc	Occitan
om	(Afan) Oromo
or	Oriya
pa	Punjabi
pl	Polish
ps	Pashto, Pushto
pt	Portuguese
qu	Quechua
rm	Rhaeto-Romance
rn	Kirundi
ro	Romanian
ru	Russian

sa Sanskrit	ta Tamil	ug Uigar
sd Sindhi	te Telugu	uk Ukrainian
sg Sangro	tg Tajik	ur Urdu
sh Serbo-Croatian	th Thai	uz Uzbek
si Singhalese	ti Tigrinya	
sk Slovak	tk Turkmen	vi Vietnamese
sl Slovenian	tl Tagalog	vo Volapuk
sm Samoan	tn Setswana	
sn Shona	to Tonga	wo Wolof
so Somali	tr Turkish	
sq Albanian	ts Tsonga	xh Xhosa
sr Serbian	tt Tatar	
ss Siswati	tw Twi	yi Yiddish
st Sesotho		yo Yoruba
su Sundanese		
sv Swedish		za Zhuang
sw Swahili		zh Chinese
		zu Zulu

B. SLP Certificates

Certificates may be used in SLP in order to distribute the public keys of trusted protected scopes. Assuming public keys, this appendix discusses the use of such certificates in the Service Location Protocol.

Possession of the private key of a protected scope is equivalent to being a trusted SA. The trustworthiness of the protected scope depends upon all of these private keys being held by trusted hosts, and used only for legitimate service registrations and deregistrations.

With access to the proper Certificate Authority (CA), DAs and UAs do not need (in advance) hold public keys which correspond to these protected scopes. They do require the public key of the CA. The CA produces certificates using its unique private key. This private key is not shared with any other system, and must remain secure. The certificates declare that a given protected scope has a given public key, as well as the expiration date of the certificate.

The ASCII (mail-safe) string format for the certificate is the following list of tag and value pairs:

```
"certificate-alg=" 1*ASN1CHAR       CRLF
"scope-charset="   1*DIGIT          CRLF
"scope="           1*RADIX-64-CHAR  CRLF
"timestamp="       16HEXDIGIT       CRLF
"public-key="      1*RADIX-64-CHAR  CRLF
"cert-digest="     1*RADIX-64-CHAR  CRLF

ASN1CHAR          = DIGIT | '.'
HEXDIGIT          = DIGIT | 'a'..'f' | 'A'..'F'
RADIX-64-CHAR     = DIGIT | 'a'..'z' | 'A'..'Z' | '+' | '/' | '='
```

The radix-64 notation is described in RFC 1521 [7]. Spaces are ignored in the computation of the binary value corresponding to a Radix-64 string. If the value for scope, public-key or cert-digest is greater than 72 characters, the Radix-64 notation may be broken up on to separate lines. The continuation lines must be preceded by one or more spaces. Only the tags listed above may start in the first column of the certificate string. This removes ambiguity in parsing the Radix-64 values (since the tags consist of legal Radix-64 values.)

The certificate-alg is the ASN.1 string for the Object Identifier value of the algorithm used to produce the "cert-digest". The scope-charset is a decimal representation of the MIBEnum value for the character set in which the scope is represented.

The radix-64 encoding of the scope string will allow the ASCII rendering of a scope string any character set.

The 8 byte NTP format timestamp is represented as 16 hex digits. This timestamp is the time at which the certificate will expire.

The format for the public key will depend on the type of cryptosystem used, which is identified by the certificate-alg. When the CA generated the certificate holding the public key being obtained, it used the message digest algorithm identified by certificate-alg to calculate a digest D on the string encoding of the certificate, excepting the cert-digest. The CA then encrypted this value using the CA's private key to produce the cert-digest, which is included in the certificate.

The CA generates the certificate off-line. The mechanism to distibute certificates is not specified in the Service Location Protocol, but may be in the future. The CA specifies the algorithms to use for message digest and public key decryption. The DA or SA

need only obtain the certificate, have a preconfigured public key for the CA and support the algorithm specified in the certificate-alg in order to obtain certified new public keys for protected scopes.

The DA or UA may confirm the certificate by calculating the message digest D, using the message digest algorithm identified by the certificate-alg. The input to the message digest algorithm is the string encoding of the certificate, excepting the cert-digest. The cert-digest is decrypted using the CA's public key to produce D'. If D is the same as D', the certificate is legitimate. The public-key for the protected scope may be used until the expiration date indicated by the certificate timestamp.

The certificate may be distributed along untrusted channels, such as email or through file transfer, as it must be verified anyhow. The CA's public key must be delivered using a trusted channel.

C. Example of deploying SLP security using MD5 and RSA

In our site, we have a protected scope "CONTROLLED". We generate a private key - public key pair for the scope, using RSA. The private key is maintained on a secret key ring by all SAs in the protected scope. The public key is available to all DAs which support the protected scope and to all UAs which will use it.

In order to register or deregister a URL, the data required to be authenticated (as described in section 4.3) is digestified using MD5 [23] to create a digital signature, then encrypted by RSA with the protected scope's private key. The output of RSA is used in the authenticator data field of the authenticator block.

The DA or UA discovers the appropriate method for verifying the authentication by looking inside the authentication block. Suppose that the "md5WithRSAEncryption" [4] algorithm has to be used to verify the signed data. The DA or UA calculates the message digest of the URL Entry by using md5, exactly as the SA did. The authenticator block is decrypted using the public key for the "CONTROLLED" scope, which is stored in the public key ring of the UA or DA under the name "CONTROLLED". If the digest calculated by the UA or DA matches that of the SA, the URL Entry has been validated.

D. Example of use of SLP Certificates by mobile nodes

Say a mobile node needs to make use of protected scopes. The mobile node is first preconfigured by adding a single public key to its public key ring: We will call it the CA-Key. This key will be used

to obtain SLP certificates in the format described in Appendix B. The corresponding private key will be used by the CA to create the certificates in the necessary format.

The CA might be operated by a system administrator using a computer which is not connected to any networks. The certificate's duration will depend on the policy of the site. The duration, scope, and public key for the protected scope, are used as input to 'md5sum'. This sum is then encrypted with RSA using the CA's private key. The radix 64 encoding of this is added to the mail-safe string based certificate encoding defined in Appendix B.

The certificate, say for the protected scope "CONTROLLED" could be made available to the mobile node. For example, it might be on a web page. The mobile node could then process the certificate in order to obtain the public key for the CONTROLLED scope. There is still no reason to *trust* this key is really the one to use (as in Appendix C). To trust it, calculate the md5 checksum of the ascii encoded certificate, excluding the cert-digest. Next, decrypt the cert-digest using the CA's public key and RSA. If the cert-digest matches the output of MD5, the certificate may be trusted (until it expires).

The mobile node requires only one key (CA-key) in order to obtain others dynamically and make use of protected scopes. Notice that we do not define any method for access control by arbitrary UAs to SAs in protected scopes. This could be done fairly easily by limiting access to the public key certificates.

E. Appendix: For Further Reading

Three related resource discovery protocols are NBP and ZIP which are part of the AppleTalk protocol family [13], the Legato Resource Administration Platform [26], and the Xerox Clearinghouse system [21]. Domain names and representation of addresses are used extensively in the Service Location Protocol. The references for these are RFCs 1034 and 1035 [18, 19]. Example of a discovery protocol for routers include Router Discovery [11] and Neighbor Discovery [20].

References

[1] Unicode Technical Report #4. The unicode standard, version 1.1 (volumes 1 and 2). Technical Report (ISBN 0-201-56788-1) and (ISBN 0-201-60845-6), Unicode Consortium, 1994.

[2] S. Alexander and R. Droms. DHCP Options and BOOTP Vendor Extensions. RFC 1533, October 1993.

[3] R. Atkinson. IP Encapsulating Security Payload. RFC 1827, August 1995.

[4] D. Balenson. Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers. RFC 1423, February 1993.

[5] T. Berners-Lee and D. Connolly. Hypertext Markup Language - 2.0. RFC 1866, November 1995.

[6] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738, December 1994.

[7] N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC 1521, September 1993.

[8] Scott Bradner. Key words for use in RFCs to Indicate Requirement Levels, January 1997. draft-bradner-key-words-03.txt (work in progress).

[9] CCITT. Specification of the Abstract Syntax Notation One (ASN.1). Recommendation X.208, 1988.

[10] CCITT. The Directory Authentication Framework. Recommendation X.509, 1988.

[11] Stephen E. Deering, editor. ICMP Router Discovery Messages. RFC 1256, September 1991.

[12] Ralph Droms. Dynamic Host Configuration Protocol. RFC 1541, October 1993.

[13] S. Gursharan, R. Andrews, and A. Oppenheimer. Inside AppleTalk. Addison-Wesley, 1990.

[14] E. Guttman. The service: URL scheme, November 1996. draft-ietf-svrloc-service-scheme-00.txt (work in progress).

[15] Geneva ISO. Code for the representation of names of languages. ISO 639:1988 (E/F), 1988.

[16] ISO 8879, Geneva. Information Processing -- Text and Office Systems - Standard Generalized Markup Language (SGML). <URL:http://www.iso.ch/cate/d16387.html>, 1986.

[17] D. Mills. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC 2030, October 1996.

[18] P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, November 1987.

[19] P. Mockapetris. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. RFC 1035, November 1987.

[20] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP version 6 (IPv6). RFC 1970, August 1996.

[21] D. Oppen and Y. Dalal. The clearinghouse: A decentralized agent for locating named objects in a distributed environment. Technical Report Tech. Rep. OPD-78103, Xerox Office Products Division, 1981.

[22] C. Perkins. DHCP Options for Service Location Protocol, August 1996. draft-ietf-dhc-slp-00.txt (work in progress).

[23] Ronald L. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.

[24] Bruce Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley, New York, NY, USA, 1994.

[25] X/Open Preliminary Specification. File System Safe UCS Transformation Format (FSS_UTF). Technical Report Document Number: P316, X/Open Company Ltd., 1994.

[26] Legato Systems. The Legato Resource Administration Platform. Legato Systems, 1991.

Authors' Addresses

Questions about this memo can be directed to:

John Veizades
@Home Network
385 Ravendale Dr.
Mountain View, CA 94043

Phone: +1 415 944 7332
Fax: +1 415 944 8500

Email: veizades@home.com

Erik Guttman
Sun Microsystems
Gaisbergstr. 6
69115 Heidelberg Germany

Phone: +1 415 336 6697

Email: Erik.Guttman@eng.sun.com

Charles E. Perkins
Sun Microsystems
2550 Garcia Avenue
Mountain View, CA 94043

Phone: +1 415 336 7153
Fax: +1 415 336 0670

EMail: cperkins@Corp.sun.com

Scott Kaplan

346 Fair Oaks St.
San Francisco, CA 94110

Phone: +1 415 285 4526

Email: scott@catch22.com