Internet Engineering Task Force
INTERNET DRAFT
30 October 1997

Erik Guttman
Charles Perkins
Sun Microsystems
John Veizades
@Home Network
Michael Day
Intel

Service Location Protocol

draft-ietf-svrloc-protocol-v2-01.txt

Status of This Memo

This document is a submission by the Service Location Working Group of the Internet Engineering Task Force (IETF). Comments should be submitted to the srvloc@corp.home.net mailing list.

Distribution of this memo is unlimited.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (North Europe), ftp.nis.garr.it (South Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

The Service Location Protocol provides a scalable framework for the discovery and selection of network services. Using this protocol, computers using the Internet need little or no 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

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 has been designed to serve enterprise networks with shared services. In its in its current form, it may not scale for wide-area service discovery throughout the

global Internet. 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 establish contact with a useful service. The UA 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 information.

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

Directory Agent (DA)
: A process which collects information from SAs to provide a single repository of service information in order to centralize it for efficient access by UAs. 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.

IANA
: IANA stands for the Internet Assigned Numbers Authority.

Naming Authority
: The agency or group which catalogues given Service Types and Attributes. The default Naming Authority is IANA.

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 20.4).

Predicate
: A boolean expression of attributes, relations and logical operators. The predicate is used to find services which satisfy particular requirements. See section 6.3.

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

Scope
: A collection or set of services that make up a logical group. See section 17 and appendix D.

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 [5].

Address Specification
: This is the network layer protocol dependent mechanism for specifying an Agent. This is part of a URL.

SLPv1
: The version of Service Location Protocol specified in RFC 2165 [19].

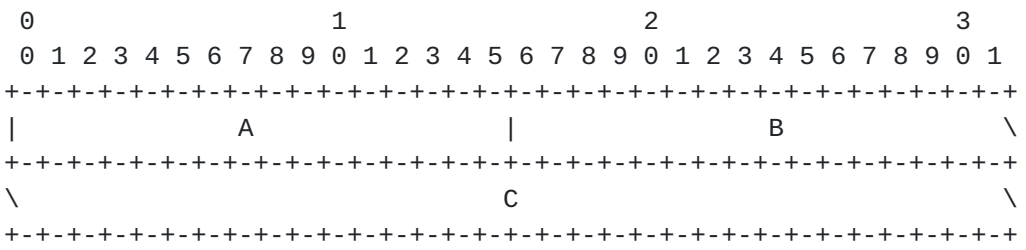
2.1. Notation Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [7].

Quoted Strings
: 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.

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.



A is a 2 byte field. B is of *arbitrary length*, where the length in bytes is typically indicated by A. C is of arbitrary length. A string field in a SLP message is 'omitted' by setting the length of the string to 0 and transmitting no character bytes.

Syntax for string based protocols will follow the conventions defined for ABNF [9]. Terms in angular brackets are defined formally in Section 20 or where they are introduced.

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. Strings used in the Service Location Protocol are NOT null-terminated.

3. Protocol Overview

The Service Location Protocol (SLP) provides a flexible and scalable framework for providing hosts with access to information about the existence, location, and configuration of networked services.

SLP is a request-reply protocol; in a typical operation a User Agent (UA) issues a request for service information and awaits one or more replies containing the requested information. Depending on the environment, replies will be sent to the UA by a Service Agent (SA), a Directory Agent (DA), or by both.

For smaller environments, SLP allows a simple peer-to-peer deployment consisting only of UAs and SAs. For larger environments, SLP allows the consolidation of service configuration data at one or more Directory Agents (DAs).

DAs, in addition to consolidating service information, allow information to be organized according to administrative, usage, or type domains using "scopes." Section 3.4 contains information on deploying SLP in larger environments using DAs and scopes.

SLP PDUs are normally transmitted in datagrams using UPD/IP. Requests may be unicast, multicast, or broadcast. When a UA multicasts or broadcasts a request, it will often recieve more than one reply. Replies must be unicast. In cases where a SLP PDU is too large to fit within a datagram, the UA, SA, or DA may unicast that PDU using TCP.

SLP allows a host to "bootstrap" itself, beginning with no knowledge of any services or SLP agents beyond its own UA. To bootstrap itself, the host must multicast or broadcast its first request.

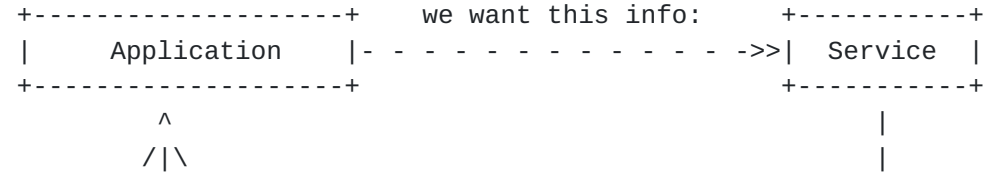
Hosts may also be preconfigured--statically or by using DHCP options--to issue requests to specific SAs or DAs, hence avoiding multicast or broadcast requests.

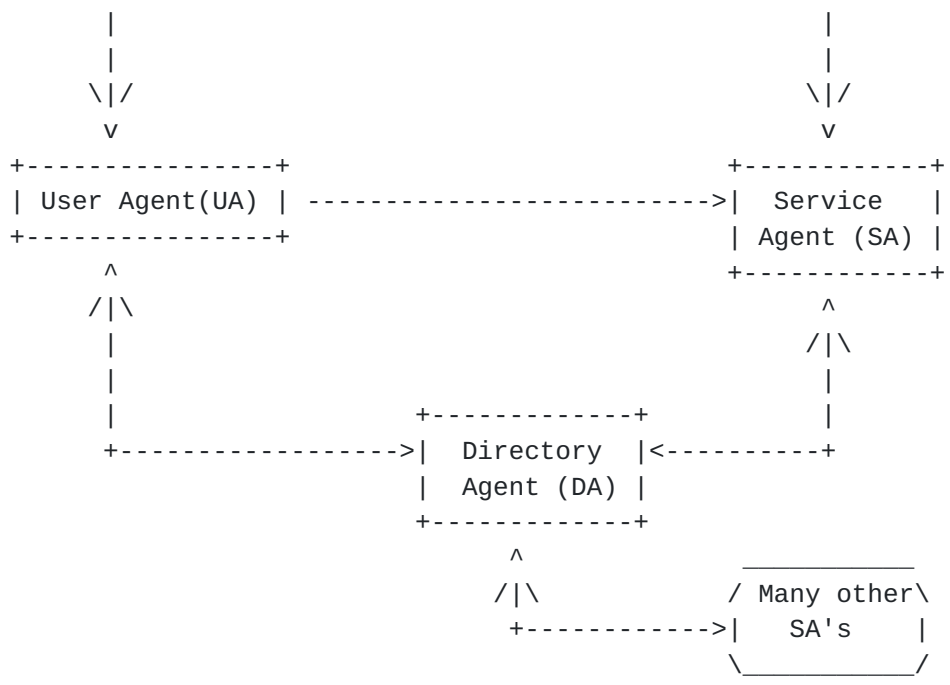
Certain conditions will influence the best strategy for deploying SLP in specific environments. Centralizing service information using DAs simplifies the process by which UAs obtain service information. However, it may not be practical to centralize service-related information that changes frequently. Also, specific environments may have special policies regarding broadcasting or multicasting.

This document specifies a range of usage models for SLP, beginning with a lightweight and simple minimal implementation for smaller or constrained environments. SLP can be scaled upward from the minimal implementation by deploying more richly featured UAs and SAs, and by adding DAs.

3.1. Protocol Operations

The diagram below shows the objects that engage in the SLP and their relationships with each other.





In the diagram above, the service contains information the application requires in order to use the service. (If the application knew this information, it could ask the service for it directly; but it doesn't and it can't.)

The service publishes its location and configuration by providing that information to the SA. When there is a DA present, the SA also passes the service's configuration and location on to the DA.

The application obtains the service's location and configuration by causing the UA to issue a service request to the SA. The UA awaits the service reply from the SA; when it arrives the UA makes the service information available to the application. The SLP includes a predicate grammar that allows the application to form service requests of varying specificity.

When one or more DAs are present, unless there is a very good reason not to (see, for example, Appendix E) the UA will issue the service request to a DA rather than to a SA. Information on using DAs is contained in section 3.4.

3.2. Minimal SLP Implementation Requirements

This section lists the required characteristics of UAs and SAs, which, together, comprise the absolute minimum functionality of a compliant SLP implementation.

Sections 3.3 and 3.4 discuss increasing degrees of greater SLP functionality, and how a more functional and scalable implementation of SLP can be deployed.

An implementation of SLP MUST, at a minimum, include a User Agent (UA) or a Service Agent (SA). A host supporting SLP MAY host one or more processes using UA and SA functions concurrently. An implementation MAY support both UA and SA funtionality. In order to deploy SLP, both UAs and SAs must be present on the network.

3.2.1. Minimal UA Requirements

A minimal UA implementation has the the following requirements:

- The UA must be able to send Service Requests (see Section 6) and receive Service Replies (see Sections 7).

- The UA must be able to send Service Requests and receive DA Advertisements, to discover DAs. The UA will do this initially and periodically. (See Section 21.1).

- The UA must be able to handle replies which overflow a single datagram (see Section 3.10.1).

- The UA must support a number of configurable parameters (see Section 22). The UA will work fine without any additional configuration, however.

- If the UA is configured to use a protected scope, it must reject unauthenticated Service or Attribute Replies (see Section 17.2.1).

For a complete list of UA characteristics see Section 24.3.

3.2.2. Minimal SA Requirements

A minimal SA implementation has the following requirements:

- The SA MUST perform both active and passive DA discovery (see Section 21.1 and 21.2) unless it is specifically configured not to perform DA Discovery (see Section 22.2).

- SAs must forward service registrations and deregistrations (see Section 13 and 15).

- SAs must reply to Service Requests (see Section 6).

- SAs must reply to UDP requests which overflow so that a UA can retransmit using TCP (see Section 3.10.1).

- SAs must be configurable with a number of parameters (see Section 22).

- If the SA is configured to work with protected scopes, it MUST generate and include signatures in all Service Reply, Service Register, and Service Deregister messages it sends. (See Section 17.2.1).

For a complete list of SA characteristics, see Section 24.2.

3.2.3. Peer-to-Peer Usage of SLP

A peer-to-peer usage of SLP is possible by having stations host both UA and SA functionality. Each station is therefore a peer, and peers make requests and provide responses directly to each other. This usage allows a lightweight implementation of SLP that works well in small environments or on smaller broadcast networks.

DAs are not required under a peer-to-peer usage of SLP. Instead of discovering centralized service information by issuing requests to DAs, SLP peers can discover services by multicasting or broadcasting service requests to other peers. Multicasting and broadcasting require usage of the convergence algorithm described in Section 21.4.

3.3. Interactive Service Selection

By supporting the Service Type Request and the Attribute Request SLP can enable interactive service discovery and Internationalization.

With interactive service selection, the user can discover all types of services present on a network. He or she can select a service type and discover all the attributes supported on the network by that type of service. The user can then form a query for specific service attributes; if such a query fails, the user can form a less specific attribute query, and so on.

Internationalization requires service attributes to be stored in multiple languages, thus making it possible for users from different communities to share resources in a more natural manner. However, internationalization of SLP service attributes requires support for the Attribute Request.

For interactivity and internationalization, UAs SHOULD send Service Type Requests and Attribute Requests by unicasting them over UDP (and

TCP in case of overflow); and by multicasting or broadcasting (if configured to do so) them over UDP.

SAs SHOULD respond to respond to Service Type Requests and Attribute Requests by unicasting the appropriate responses over UDP (and TCP in the case of overflow).

If the SA supports Service Type Requests or Attribute Requests, when it receives an incorrect Service Type or Attribute Request, it MUST return an error code to the requester. Error codes are in Section 25.

3.4. Using SLP Directory Agents In Larger Environments

Through the deployment of DAs and the usage of scopes, SLP can scale up to larger environments.

3.4.1. Directory Agents

DAs are consolidated stores of service location, configuration, and attribute information. They exist to enhance the performance and upward scalability of SLP. When DAs are present, SAs MUST register all of their service information with the DAs.

When DAs are present, UAs SHOULD unicast requests directly to a DA (when scoping rules allow), hence avoiding using the multicast and convergence algorithm to obtain service information. This decreases network utilization and increases the speed at which UAs can obtain service information.

A single DA can support many UAs. Moreover, many DAs can coreside on a network, which makes load balancing and redundancy possible. DAs reduce the load on SAs, which makes simpler implementations of SAs possible.

UAs can discover DAs using static configuration, DHCP options, or by multicasting (or broadcasting) Service Requests using the convergence algorithm in Section 21.4. n a large scale case, the configuration of UAs might be through SRV records in the DNS or TXT records pointing to a DA URL (see [16])

When DAs are deployed, are subject to the requirements listed in Section 24.1.

3.4.2. Scopes

Scopes exist to make SLP easier to use and to administer in larger environments, and require the presence of DAs. Scopes are a logical mechanism that allows the grouping of services into different sets. Services can be grouped according to organizational structure, physical location, type, and so on. Some scopes can be protected (subject to authentication of service information).

Scoped DAs accept registrations and requests that are within their scope. For protected scopes, this means that the registrations and requests are signed using the scope's Public Key. DAs that are not scoped accept registrations and requests from any agent.

SAs and UAs must use scopes when they exist. UAs use scopes to focus their requests to specific groups of services. SAs use scopes to advertise (register) their service information as part of a specific group.

Unscoped services are discoverable and usable by anyone.

In the special case of DA Discovery, requests may be sent to DAs with scoping turned off (see Section 6.2).

UAs may issue requests to SAs with scoping turned off (see Section 17.1.3). In this case, SAs will ignore scoping when replying to the request.

All SLP agents can discover scopes by observing DA Advertisements, by using DHCP options, or by static configuration.

Scoping rules are detailed in Section 17.1.

3.4.3. Scaling Configuration Options

By using specific configuration settings, SLP agents can work better in larger environments.

The most effective mechanisms SLP offers for scaling up are DAs and scopes. When DAs and scopes are present on the network, further gains can be had by doing the following:

- Configuring UAs and SAs to have a predefined scope. These agents can then perform DA discovery and make requests using their scope. This will limit the number of replies.

- Turning off DA discovery and using DHCP options to gain DA and scoping information. This will limit the amount of bandwidth consumed by DA discovery.

- Adjusting the TTL for multicast requests downward to limit the scope of multicast requests. This focuses the multicast convergence algorithm on smaller subnetworks, which decreases the number of responses and increases the performance of service location.

3.5. Introduction to Directory Agents

A DA acts on behalf of many SAs. It acquires information from them and acts as a soft state cache. It is a single point of contact to supply that information to UAs.

The queries that a UA multicasts to SAs (in an environment without a DA) are the same as queries that the UA might unicast to a DA. A UA may cache information about the presence of alternate DAs to use in case a selected DA fails.

Aside from enhancing the scalability of the protocol (see Section D), running multiple DAs provides robustness of operation. The DAs may have replicated service information which remain accessible even when one of the DAs fail. DAs, 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 larger administrative domains.

Each SA MUST register with all DAs they are configured to use. UAs may choose among DAs they are configured to use. UAs and SAs determine which DAs to use based on scope rules described in Section 17.1.

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

3.6. URLs used in the Service Location Protocol

The Service Location Protocol uses URLs to indicate the location of services. URLs are used in a variety of Service Location Messages: SAs send them to register and deregister service advertisements, UAs obtain them in Service Replies and may send them in Attribute

Requests. Any URL which conforms to standard URL syntax conventions (see RFC 1738 [5]) may be used in these messages.

Service: URLs are useful for transmitting a service's location to a client application. Other standard URL schemes may also be used for this purpose.

3.6.1. The ``service:'' URL scheme

The service URL scheme is used specifically to communicate a Service Location. Many Service Types will be named by including a standard network service name after the ``service:'' scheme name.

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. See [12].

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

3.7. 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 specified by documents maintained with IANA will use their own Naming Authority string. The procedure for standardizing new Service Types is defined in [12].

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 UAs.

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 if one 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. The format of a Service Type string is described in Section 20.1.1.

3.8. 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.

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 6 and 13.

Service Types SHOULD be registered with IANA to allow for Internet-wide interoperability.

3.9. 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 UA 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.

3.10. Transmission of SLP messages

UAs MUST be able to issue requests to DAs using UDP and SAs using multicast/convergence. UAs will use TCP in the case of overflow as described below. UAs MAY be able to issue UDP requests to SAs, but this is not normally necessary.

SAs MUST be able to respond to UDP, TCP and multicast requests.

DAs MUST be able to respond to UDP and TCP requests, as well as multicast DA Discovery SrvRqsts.

See Section 21 for timing and retransmission rules on how these messages are exchanged.

3.10.1. Use of TCP

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. The only two reasons to use TCP are:

First, a registration or deregistration may be too large to fit into a datagram. If no MTU information is available for the route, assume that the MTU is 1400 which is enough to accommodate a IPv6 header and UDP header in an ethernet frame. This value is configurable (see Section 22). In this case a connection must be set up from a SA to a DA. UAs may establish a TCP connection with a DA (not an SA) to send requests which do not fit in a datagram to DAs.

Second, if the reply to a UA's request overflows a datagram, the DA or SA truncates the reply to fit in one datagram and sets the 'overflow' bit in the SLP header. A UA which receives such a reply MAY open a TCP connection with the DA or SA and retransmit the request. It MAY also attempt to make use of the truncated reply or reformulate a more restrictive request which will result in a smaller reply.

DAs and SAs MUST respond to connection requests; SAs whose registration data can overflow a datagram must be able to use TCP to send the registration.

A TCP 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_CLOSE_CONN seconds 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.

SAs and UAs use ephemeral ports for transmitting information to the service location port, which is 427.

3.10.2. Use of Multicast Addresses

SrvTypeRqst messages are sent to the Service Location General Multicast Address. SrvRqst messages used for DA Discovery are sent to the Directory Agent Discovery Multicast Address.

SAs must join multicast groups depending on which services they advertise (called Service-Specific multicast addresses). They also must join the Service Location General Multicast Address.

UAs send multicast SrvRqst or AttrRqst messages to the Service-Specific multicast group corresponding to the service type of the request.

Service-Specific Multicast addresses are computed by calculating a string hash on the service type string. The service type string is defined in Section 20.1.1. This string will always fall inside the ASCII range of the UTF8 [21] encoding due to its definition.

The multicast group they join is determined by the string hash function given below:

```
#define RANGE_SIZE 0x7f
/*
 *  SLPhash returns a hash value in the range 0-RANGE_SIZE 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 (RANGE_SIZE & h);  /* round to fit in range of addresses */
}
```

This value is added to the base range of Service Specific Discovery Addresses, to be assigned by IANA. These will be 128 contiguous multicast addresses from the administrative local multicast range.

3.10.3. 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.

3.10.3.1. Single Subnet

In isolated networks, broadcasts will work in place of multicast.

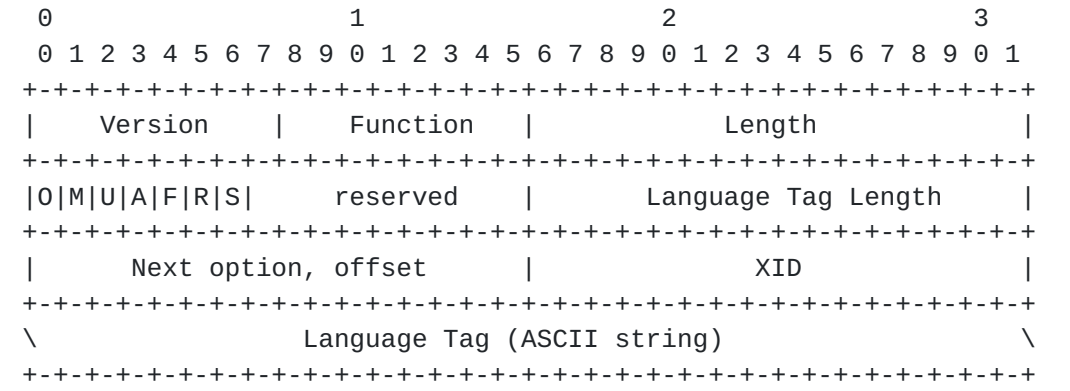
SAs SHOULD and DAs MUST listen for broadcast Service Location request messages to the Service Location port. This allows UAs which lack multicast capabilities to still make use of Service Location on a single subnet.

3.10.3.2. Multiple Subnets

In larger enterprises, a DA can be used to provide a central clearing house of information for UAs. The DA address can be dynamically configured with Agents using DHCP. The address can also be determined by static configuration. Using multicast DA discovery in enterprises with multiple subnets will require use of multicast discovery with multiple hops (i.e., TTL > 1 in the IP header). Note that the setting of the TTL in multicast packets sometimes must be interpreted according to conventional scoping agreements rather than strictly as the number of hops.

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.



The header for SLP v2 has changed from SLP v1 in the following ways. Dialect is no longer needed, as it is included in the Language Tag. The Language Code field is now interpreted as the length of the Language Tag which follows the header. Finally, since all characters

will be encoded in UTF-8, the Char Encoding field of SLP v1 is no longer necessary. This field is now used as an offset to the Next Option.

Version This protocol document defines version 2 of the Service Location protocol. A SLP implementation MAY support version 1 [RFC2165]. If a SLP message indicates it is sent using version 1 and this is not supported, a PROTOCOL_V1_REJECTED error is returned.

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	SrvRqst	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 3.10 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 18) that is indicated by the Service or Attribute Request.

U The 'URL Authentication Present' bit. See Sections 4.3, 4.4, 13, and 15 for the use of this field.

A The 'Attribute Authentication Present' bit. See Sections 4.3, 4.4, and 11 for the use of this field.

F If the 'Fresh bit' is set by the SA when it makes a Service Registration if it is to be considered 'fresh'. If this bit is not present, the registration is considered to be an update.

R The 'Multicast request' bit. This is set by the UA when it multicasts or broadcasts a request.

S The 'Specific Scoping' bit. This is set by Service Location Protocol entities wishing to exclude services not assigned to any scope, or by services unwilling to provide service to user agents not specifying the particular scope.

reserved MUST be zero.

Language Tag Length
Strings within the remainder of the message which follows are to be interpreted in the language specified (see Section 18) in the bytes following the header. The Language Tag is defined by RFC 1766 [2].

Next Option Offset
Options are added after the regular payload in the SLP packet. If no options are present, then this field is set to 0.

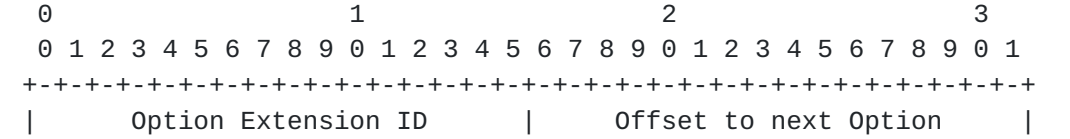
Transaction Identifier (XID)
The XID (transaction ID) field allows the requester to match replies to individual requests (see Section 4.2).

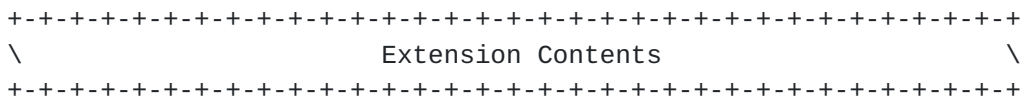
Note that, whenever there is an Attribute Authentication block, there will also be a URL Authentication block. Thus, it is an error to have the 'A' bit set without also having the 'U' bit set.

4.1. Service Location Extension Options

A service location extension option must be specified by a standards track document. The option may be defined to accompany any or all Service Location Messages. A conforming SLP implementation MUST be able to ignore Service Location Extension Options it does not recognize. It may be the case that future Options will be defined and standardized which will become requirements of SLP implementations. These documents will have to proceed as Standards Track specifications. Otherwise, support of all extension options is elective.

The format of a Service Location Extension Option is:





The Option Extension ID is defined by a Standards document which also defines the contents of the extension. The offset to next option is 0 if there is no option following or is set to the length of the Extension contents.

4.2. Retransmission and Transaction IDs (XIDs)

Retransmission is used by UAs and SAs to ensure reliable exchange of unicast messages with DAs. If a UA or SA sends a message to a DA fails to receive a response, the message will be sent again. The message is retried up to 3 times

Multicast requests are retransmitted, but according to different rules. See Section 21 for the details of the algorithm.

Replies received using the multicast convergence algorithm are accumulated until convergence has been detected.

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.

Retransmission of the same message 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_KEEP_RPLY is recommended.) Any registration or deregistration at a DA, or change of service information at a SA should flush this cache so that the information returned to the client is always valid.

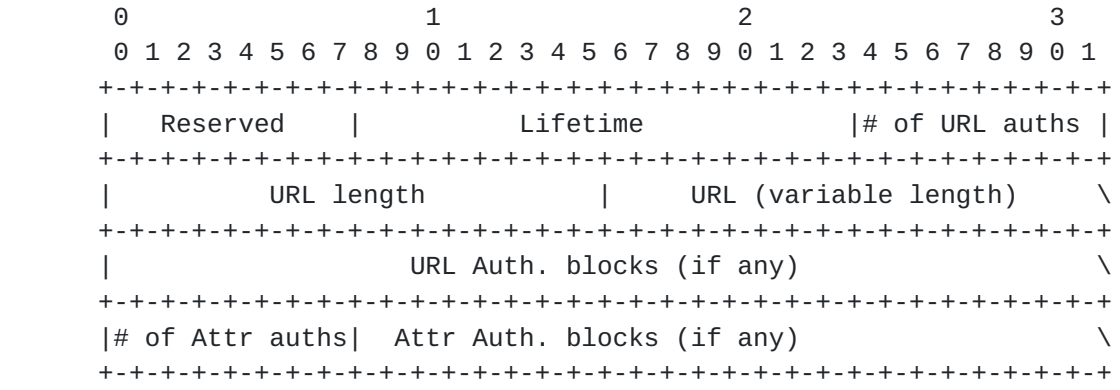
The requester creates the XID from an initial random seed and increments it by one for each request it makes. The XIDs will eventually wrap and continue incrementing from there. Requests are never sent with an XID of 0.

An unsolicited DAAdvert has an XID of 0.

Requests all include XIDs which will match the XIDs of the replies. The replies may be returned in any order. A UA or SA may have multiple outstanding requests.

4.3. 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:



Reserved — MUST be zero. This field is reserved for longer lifetimes used by a wide area service location protocol.

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 authentication block (Section 4.4)

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 used for denoting a service access point, (see [12] for the formal definition.) Other URLs besides service: scheme URLs may be transmitted in Service Location Messages.

4.4. Authentication Blocks

Authentication blocks are used to authenticate service registrations and deregistrations. URLs are registered along with an URL

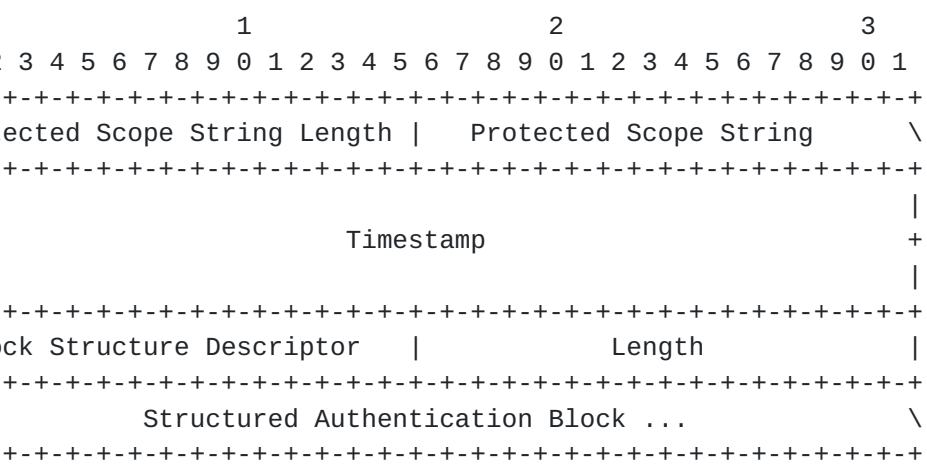
Authentication block to retain the authentication information in the URL entry for subsequent use by UAs who receive a Service Reply containing the URL entry. Service attributes are registered along with an Attribute Authentication block. Both authentication blocks have the format illustrated below.

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.

A single Authentication Block is returned with an AttrRply and SrvRply. A SrvReg may include multiple authentication blocks if more the service is to be registered in more than one protected scope, or if more than one cryptographic algorithm is supported by the Service Location Protocol deployment. A DAAdvert will include one Authentication Block per protected scope that the DA supports. Unsolicited DAAdverts (see Section 12) will always be made in using the default cryptographic algorithm (see below). DAAdverts sent as a reply to a SrvRqst will include only one Authenticator Block.

An authentication block has the following format:



Timestamp A 64-bit value formatted as specified by the Network

Time Protocol (NTP) [15].

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

Length
: The length of the Authentication Block

Structured Authentication Block
: An algorithm specification, and the authentication data produced by the algorithm.

The Structured Authentication Block contains a digital signature of the information being authenticated. It contains sufficient information to determine the algorithm to be used and the keys to be selected to verify the digital signature.

The digital signature is computed over the following ordered stream of data:

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 [15])
LENGTH OF SCOPE STRING	(2 bytes)
SCOPE STRING	(n bytes)

When producing a URL Authentication block, the authentication data produced by the algorithm identified within the Structured Authentication Block calculated over the following ordered stream of data:

LENGTH OF ATTRIBUTES	(2 bytes in network byte order)
ATTRIBUTES	(n bytes)
TIMESTAMP	(8 bytes in SNTP format [15])
LENGTH OF SCOPE STRING	(2 bytes)
SCOPE STRING	(n bytes)

Every Service Location Protocol entity (UA, SA, or DA) which is configured for use with protected scopes MUST implement "md5WithRSAEncryption" [4] and be able to associate it with BSD value == 1. A conforming SLP implementation MAY implement other digital signature systems.

In the case where BSD == 1 and OID == "md5WithRSAEncryption" is selected, the Structured Authentication Block will start with the ASN.1 Distinguished Encoding (DER) [8] for "md5WithRSAEncryption", which has the as its value the bytes (MSB first in hex):

"30 0d 06 09 2a 86 48 86 f7 0d 01 01 04 05 00"

This is then immediately followed by an ASN.1 Distinguished Encoding (as a "Bitstring") of the RSA encryption (using the Scope's private key) of a bitstring consisting of the OID for "MD5" concatenated by the MD5 [18] message digest computed over the fields above. The exact construction of the MD5 OID and digest can be found in RFC 1423 [4].

4.5. 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. UAs 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_LIFETIME. 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 Location Protocol Requests

SLP includes SrvRqst, AttrRqst and SrvTypeRqst messages. All UAs MUST be able to issue SrvRqst messages and SHOULD be able to issue AttrRqst and SrvTypeRqst messages. SAs issue only SrvRqst messages, and only for the purpose of DA discovery.

SAs MUST be able to respond to SrvRqsts and SHOULD be able to respond to AttrRqsts and SrvTypeRqsts.

UAs MUST use scoped DAs in preference to unscoped DAs and any DA in preference to multicasting to SAs. See Section 17.1 for rules concerning the use of scopes.

Multicast and broadcast requests MUST set the 'R' bit in the header. This will ease implementation of combined DA and SA services by making it apparent whether an arriving datagram was unicast or multicast. Multicast (or broadcast) service requests MUST be ignored by DAs.

Replies to SLP requests include a 2 byte error code. If the error code indicates failure the rest of the message SHOULD be omitted. The error codes which may be returned are:

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 which can satisfy it, or the request arrived with the Monolingual bit set and no registration is available in the specified language. See Section 18. Note: SrvTypeRqst messages do not contain a Language Tag, so they never elicit this error.

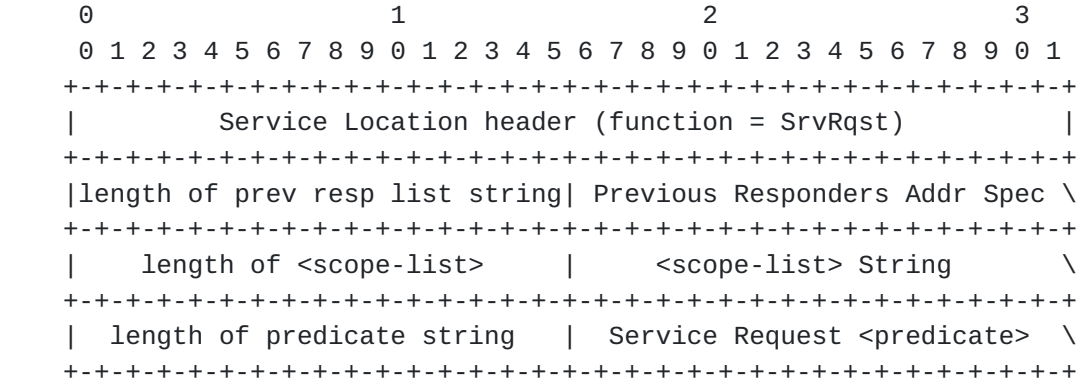
PROTOCOL_PARSE_ERROR
A SA or DA returns this error when a SrvRply is received which cannot be parsed or the declared string lengths overrun the message.

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.

6. Service Request Message Format

The Service Request is used by UAs to obtain URLs from a DA or SAs.

The format of the Service Request is as follows:



The <Previous Responders Addr Spec> is described in Sections 8 and 20.1.

See Section 17.1 for the interpretation of the scope-list field.

The predicate allows the UA to request the Service Type and specific required attributes of a services in a specific language. The minimal form of a Service Request Predicate is shown below:

"(service-type=" <srvtype> ["."<na>] ")"

Service Requests MAY include a scope term and a 'where clause'. The syntax of the query language is described in Section 6.4. The form of a Service Request is:

"(&(service-type=" <srvtype> ["."<na>] ")" <where> ")"

where:

- The <srvtype> refers to the Service Type. For each type of service available, there is a unique Service type name string. This term MUST be part of every Service Request. See Section 20.1.1.

- The <na> is the Naming Authority. This term is OPTIONAL. See Section 20.1.1.

- The <where> string contains a set of query terms which will indicate those service instances which the User Agent is interested in. This clause includes attributes, boolean operators and relations. (See Section 6.3.)

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 satisfy the scoping rules for requests (see Section 17.1).

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

6.1. Service Request Usage

The UA forms SrvRqsts using standard or conventionally known Service Type attributes. It MAY also issue AttrRqsts to obtain the attribute values for a Service Type before issuing SrvRqsts (see Section 11). Having obtained the attributes which describe a particular kind of service from an AttrRqsts, or using configured knowledge of a

service's attributes, the UA can build a predicate that describes the service needs of the user.

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:

```
(& (SERVICE-TYPE=LPR)(PAGES PER MINUTE=12)
   (UNRESTRICTED_ACCESS=*)
   (LOCATION=12th FLOOR))
```

Suppose there is no such printer. The DA responds with a SrvRply with 0 in the number of responses and no reply values. A SA silently discards the request.

The UA might then try a less restrictive query to find a printer, using only the 12th floor as "where" criteria.

```
(&(SERVICE-TYPE=LPR)(LOCATION=12th FLOOR))
```

In this case, there might be the 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.

6.2. Directory Agent Discovery Request

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

UAs and SAs which lack preconfigured or DHCP configured knowledge of a DA MUST multicast a Service Request to the DA Discovery Multicast Address. For the details of the multicast convergence algorithm see Section 21. The predicate included in this request is:

```
(service-type=directory-agent)
```

The <scope-list> in the DA discovery request may include only the string "*" in their <scope-list>. In this case, all DAs respond.

If it includes any scope strings, only those DAs which support the indicated scope reply, or those which support unscoped requests. A DA discovery request which has the 'S' bit set in the message header will not get responses from unscoped DAs.

Normally, a DA will not respond to multicast requests, nor will it respond to requests which are improperly scoped. This request is a special case, if a properly scoped DA Discovery request is received from the DA Discovery multicast group a DA MUST respond. If the request specifies a scope, the DA MUST NOT respond unless the scope request matches its scope. If a DA has no scope, it will only respond if there is no scope term or the request does not have the 'S' bit set.

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

```
URL returned:   service:directory-agent://slp-resolver.big.org
Scope returned: NONE (ie. the length field is set to 0 for the
                scope string.)

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

URL returned:   service:directory-agent://204.182.15.66
Scope returned: Legal Department  ('S' bit set)
```

The first DA supports only unscoped advertisements. The second supports advertisements which are unscoped, or are in the JANITORIAL SERVICES or ADMIN scope. The DA in the last example supports only the LEGAL DEPARTMENT scope, and explicitly NOT unscoped advertisements.

The DA Advertisement format is defined in Section 12.

If the goal is merely to discover any DA, the first DA Advert which is received will do. If the goal, however, is to discover all reachable DAs, the multicast convergence algorithm must be used, see Section 21.

6.3. 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 predicate uses the same syntax as a LDAP search filter [13]. This means that a SLP service request could be handled by a LDAP

server. The reverse is not true: SLP uses a greatly simplified attribute typing system. Thus a SA or DA cannot interpret an arbitrary LDAP query. There are some restrictions and assumptions which are necessary in order to interpret SrvRqst predicates in the context of SLP.

- A query for keywords uses a 'present' filter type. Thus, to query for services including the keyword 'X', the filter would be "X=*".

- Degenerate queries such as "(&" <filter> ")" and "(|" <filter> ")" must be tolerated. They are equivalent to <filter>.

- LDAPv2 defines several data types [14]. The only data types which are supported in SLP are String (equivalent to a Case Ignore String), Opaque (equivalent to a Case Exact String, since the binary values is encoded as a radix64 value). Integer and Boolean values are identically represented in SLP and LDAPv2.

6.4. Service Request Predicates

The Service Request Predicate follows the grammar of the LDAPv2 string search filter [13].

The Predicate MUST contain a simple term which defines a service type for the query. It MAY contain a simple term which defines the scope for the query (see Section 17.1.)

Restrictions which are not implied by the grammar are:

- LDAPv2 string text filters [13] and string based attribute encodings [14] are all ASCII based. For UTF8 character encoding, which SLP supports, either the LDAPv3 filters (and UTF8) must be used or the <a> production in RFC 1778, Section 2, must be expanded to include the entire range of allowed alphanumerics excluding those in the <p> (protected) rule.

- <approx> filters, ``=~'', are interpreted as <equal> filters ``=''. For strings the <approx> filter MAY be implemented so that it returns true for the service with the 'greatest possible' match of the string sequence. For example: (x=~ABCD) applied to services S1 with x=AAAA, S2 with x=ABBBBB and S3 with x=ABCCCC would return S3 since it matches the longest sequence of characters in the request.

- <starval> terms are always interpreted as string values and may only be used with <equal> filters, with the two exceptions

- <greater> filters applied to ``*'' as the value will return true only for the item whose value is the MAXIMUM for the attribute. This only applies to an Integer attribute. Otherwise, it returns a PROTOCOL_PARSE_ERROR.

- <less> filters applied to ``*'' as the value will return true only for the item(s) whose value is the MINIMUM for the attribute. This only applies to an Integer attribute. Otherwise, it returns a PROTOTOL_PARSE_ERROR.

The following are examples of query predicates in Service Requests. When the 'S' is present, it means that the 'S' bit is set in the message header. See Section 17.1 for how details on how the <scope-list> works.

"(service-type=http)" <scope-list>= none
This is a minimal request string. It matches all http services.

"(service-type=lpr)" <scope-list>="SALES"
This request is for all lpr services either universally available or available in the SALES scope.

"(service-type=lpr)" <scope-list>="SALES", 'S'
This request is for all lpr services which have been explicitly configured to reside within in the SALES scope.

"(&(service-type=pop3)(user=wump))" <scope-list>="ADMIN"
This is a request for all pop3 services available in the ADMIN scope (or are unscoped) and which serve mail to the user 'wump'.

"(&(service-type=backup)(qlength<=*))" <scope-list>="BLDG 32", 'S'

This returns the backup service which has the shortest queue-length. (This assumes that the queue length is an integer based attribute). It will return this only for services registered with the BLDG 32 scope (not unscoped services.)

6.5. String Matching for Requests

All strings are case insensitive, with respect to string matching on queries. All preceding or trailing blanks which are not escaped should not be considered for a match. White space (SPACE, CR, LF, TAB) internal to a string value is folded to a single SPACE character

for the sake of string comparisons.

For example, " Some String " matches "SOME   STRING".

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, follow or be internal to a string value in order to indicate substring matching. The query including this character matches any character sequence which conforms to the letters which are not wildcarded.

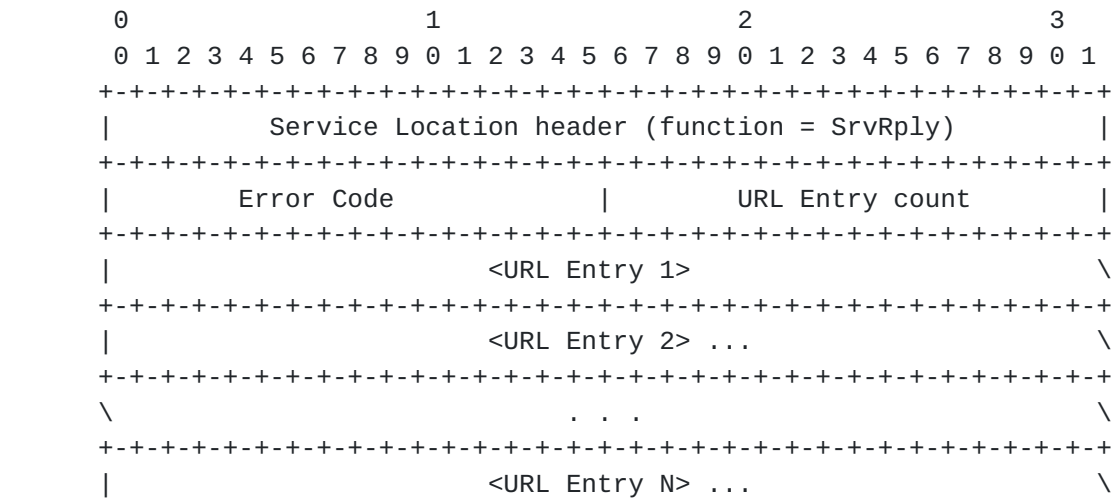
Examples:

"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"
"b*b" matches "bob" and "big dreams no grub"

String matching is done after escape sequences have been substituted. See Sections 18, 6.3, 19.

7. Service Reply Message Format

The format of the Service Reply Message is:



+-+

Each Service Reply message is is sent in response to a Service Request message. The XID field in the Service Reply is copied from the XID field in the Service Request message.

A successful Service Reply message is composed of a list of URL Entries, and an error code indicating the successful status of the request. If an error is encountered during the processing of the Service Request, the Error Code is set appropriately, according to the available errors in 25. If the error code indicates failure the rest of the message SHOULD be omitted. In addition to the errors listed in section 5, the EXPECTED_ATTRIBUTE_MISSING error may be returned. A DA or SA with access to the scheme definition for a service: URL [12] MAY return this error (along with the matching URLs) if the Service Request query does NOT contain a value for an "expected" attribute (i.e., an attribute marked with the 'X' flag in the scheme template). A SA will not return this error upon receiving a multicast request; it will silently discard the multicast request instead.

Each <URL Entry> in the list has the form defined in Section 4.3. If the presence of an URL Authentication block is signaled by the 'U' bit, the length of the authentication block is determined by information within the block as discussed in Section 4.4. The URL Authentication block will include the authentication block calculated using the algorithm specified in the SrvRqst if possible. If not, the Authentication block calculated using the default algorithm is supplied.

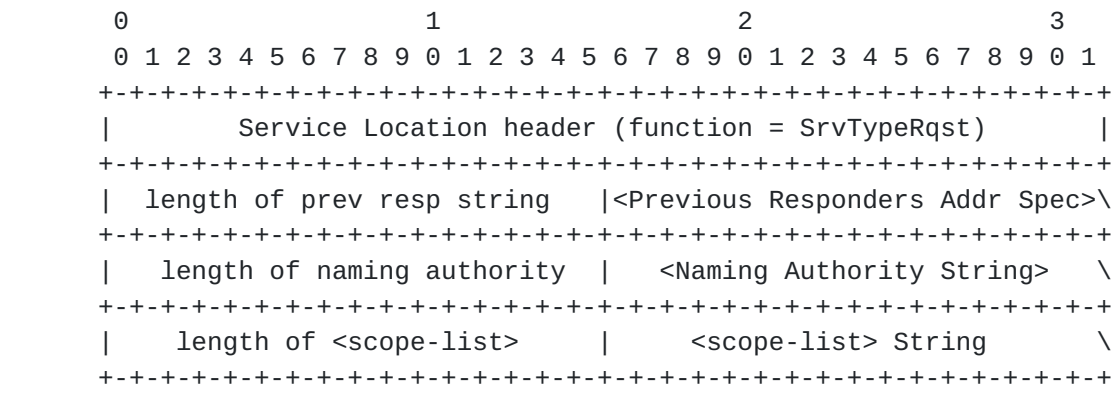
A UA MAY use the authentication block to determine whether the SA advertising the URL is, in fact, authorized to offer the indicated service.

If, in a list of URL entries, some of the URLs indicate services which are in protected scopes (see Section 17.2) 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 authentication block is shown as zero, and no authentication need be done.

8. Service Type Request Message Format

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

The format of a Service Type Request is:



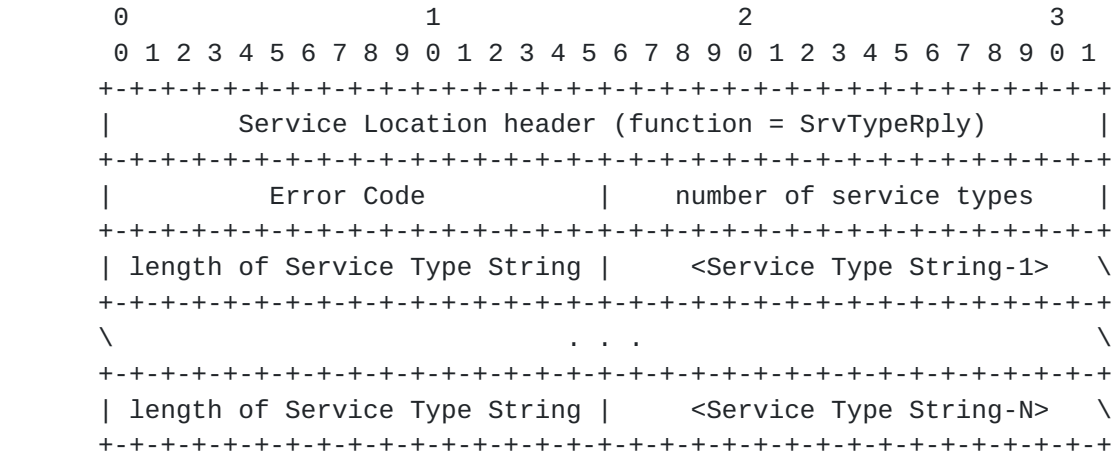
The <Previous Responders Addr Spec> is a comma delimited list. See Section 20.1.

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.

See Section 17.1 for the interpretation of the scope-list field.

9. Service Type Reply Message Format

The Service Type Reply has the following format:



The service type's name is provided in the <Service Type String>. If the service type has a naming authority other than "IANA" it MUST be

returned following the service type string and a "." character. See Section 20.1.1 for the formal definition of this field.

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

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

10. Attribute Request Message Format

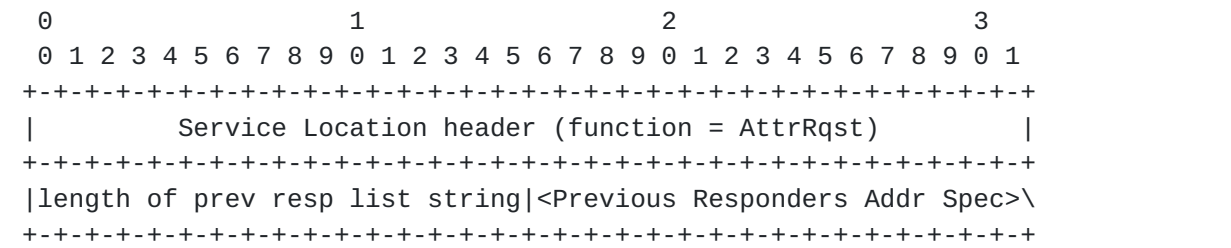
The Attribute Request is used to obtain attribute information. The UA supplies a request and the appropriate attribute information is returned.

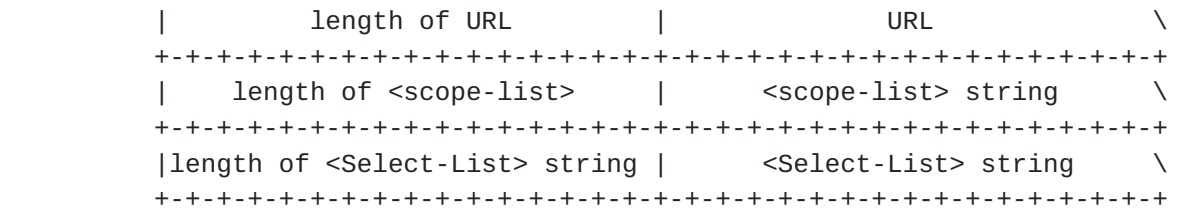
If the UA supplies only a Service Type, then the reply includes all attributes and all values for that Service Type. The reply includes only those attributes for which services exist and are advertised by the DA or SA which received the Attribute Request. Since different instances of a given service can, and very likely will, have different values for the attributes defined by the Service Type, the UA 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 MAY include a 'select clause'. This limits 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:





The <Previous Responder Address List> functions exactly as introduced in Section 8. See also Section 20.1.

The URL field can take two forms: Either it is simply a Service Type (see Section 20.1.1), such as "service:http:" or "nfs:". All attributes and the full range of values for each attribute of all services of the given Service Type is returned.

The URL field may also be a full URL, such as "service:lpr://igore.wco.ftp.com:515/draft" or "ftp://max.net/znoo". In this, only the attributes for the service of the specified URL is defined are returned.

See Section 17.1 for the interpretation of the scope-list field.

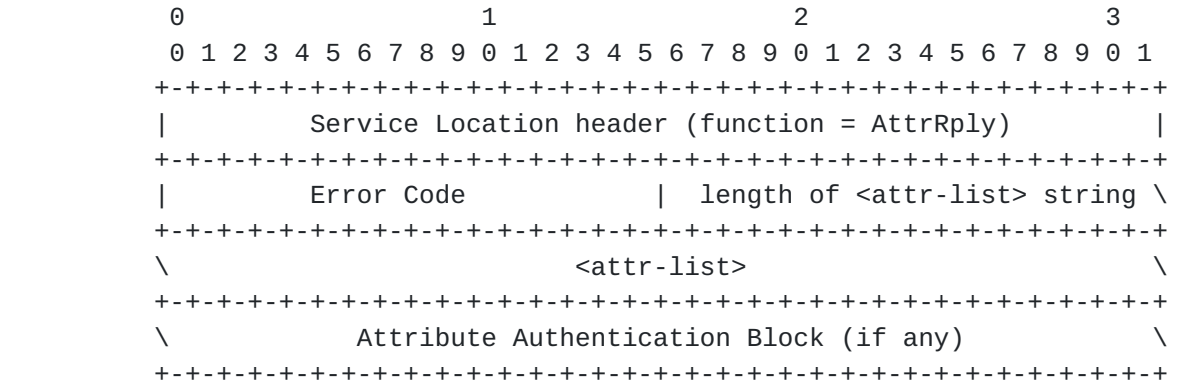
The select list takes the form of a <Select-List>, see 20. The items on the list are attribute tags or keywords, which can be either complete tags or include '*' characters for wildcard string matching.

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

"PAGES PER MINUTE,UNRESTRICTED_ACCESS,LOCATION"

11. Attribute Reply Message Format

An Attribute Reply Message takes the form:



The <attr-list> (attribute list) has the same form as the attribute list in a Service Registration, see Section 20.2 for a formal definition of this field.

If the AttrRqst included a Service Type in the URL field, all attribute results must be coallated. For example, a SA receives an AttrRqst and there are three services of the specified type (in the requested scope, etc.). The first service supports the attribute "(A=1,2)", the second service, "(A=3,4)" and the third, "(A=5,6)". The SA returns the attribute "A" in the AttrRply as "(A=5,2,3,1,4,6)". (Note that the SA does not need to sort the return values in any way.)

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

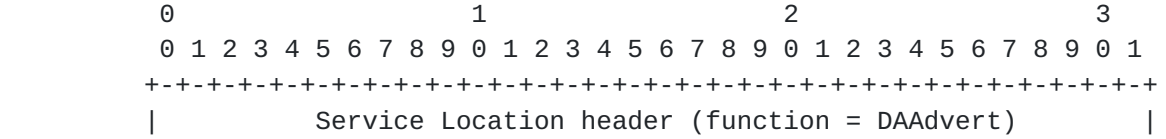
```
(PAPER COLOR=WHITE,BLUE),
(PAPER SIZE=LEGAL,LETTER,ENVELOPE,TRACTOR FEED),
UNRESTRICTED_ACCESS,
(PAGES PER MINUTE=1,3,12),
(LOCATION=12th, NEAR ARUNA'S OFFICE),
(QUEUES=LEGAL,LETTER,ENVELOPE,LETTER HEAD)
```

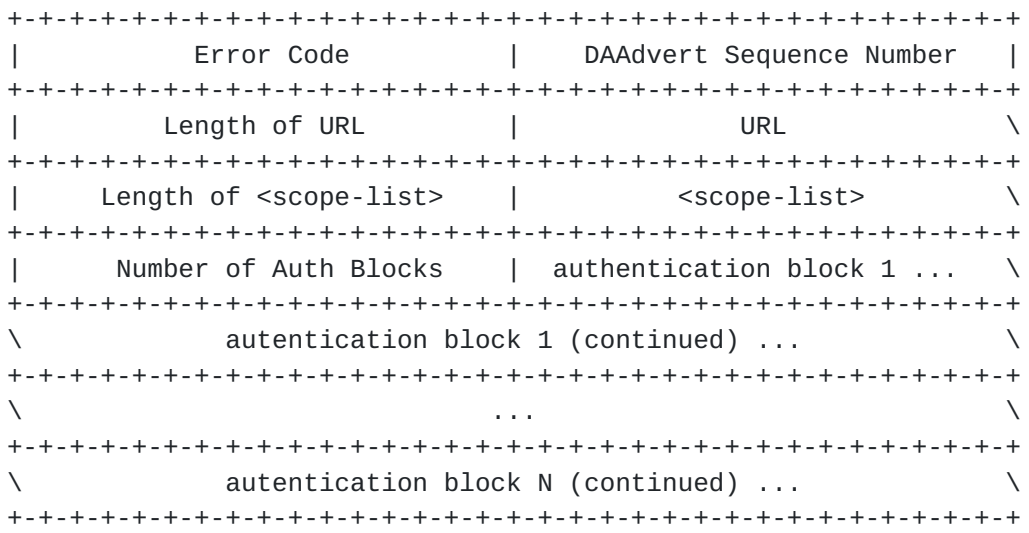
If the message header has the 'A' bit set, the Attribute Reply will have an Attribute Authentication block set. In this case, the Attribute Authentication Block must be returned with the entire list of attributes, exactly as it was registered by an SA in a protected scope. In this case, the URL was registered 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 Authentication so it simply returns the attributes without an authentication block.

The SA or DA returns the Attribute Authentication 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.

12. Directory Agent Advertisement Message Format

DA Advertisement Messages have the following format:





The Error Code is set when a DA Advertisement is returned as the result of a Service Request, as specified in Section 5. The Error Code will always be set to 0 in the case of an unsolicited DA Advertisement.

The DAAdvert Sequence Number is 0 when the DA comes up initially and increases by one each time the DAAdvert is sent unsolicited. DAs which store service advertisements in a nonvolatile store will set their initial sequence number to 256. The sequence number will be used by SAs to determine if they must reregister services when a DA is discovered. See Section 16.1 which describes unsolicited DAAdverts and how SAs respond to them.

The URL corresponds to the DA's location.

There MUST be at least one authentication block for each protected scope. There MAY be more than one authentication block for each protected scope if more than one authentication algorithm (identified by the BSD field) is used.

See Section 17.1 for the definition of the <scope-list>.

If the 'U' bit in the SLP header is set, an authentication block is included to allow the SA and UA to ascertain whether the DA Advertisement is valid. See 4.4. See Section 6.4 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
```

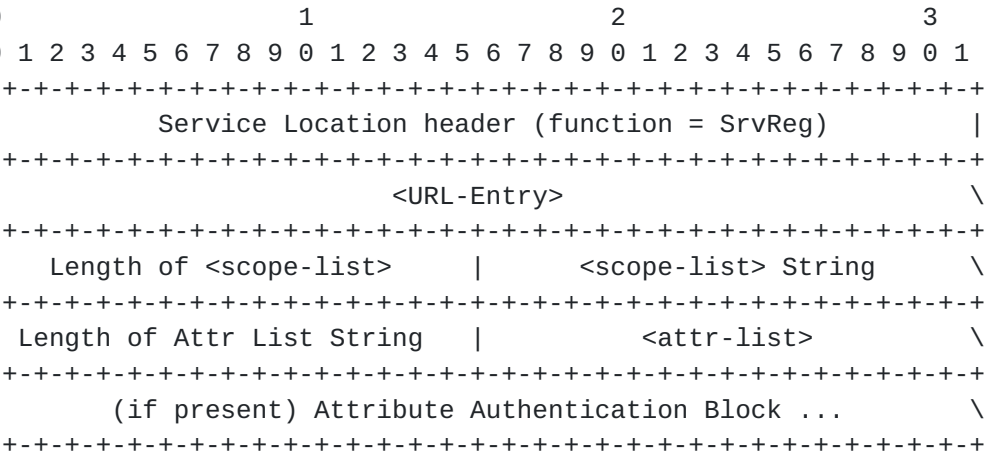
The DA can be reached at the Address Specification returned, and supports the SCOPE called "ADMIN".

13. Service Registration Message Format

After a SA has found a DA, due to either active DA Discovery (see Section 21.1) or passive DA Discovery (see Section 21.2) it begins to register its advertised services one at a time. A SA must wait for some random time uniformly distributed within the range specified by CONFIG_REG_ACTIVE seconds 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 17.2, then the service MUST include both a URL Authentication block and an Attribute Authentication block (see Section 4.4). In that case, the service agent MUST set both the 'U' bit and the 'A' bit (see Section 4).

A DA must acknowledge each service registration request. If authentication blocks are included, the DA 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:



The <URL-Entry> is defined at the end of Section 4.3. The <attr-list> is defined in Section 20.2. The Attribute Authentication Block, which is only present if the 'A' bit is set in the message header, is defined in Section 4.4.

The <scope-list> is defined in Section 17.1.

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 SA MUST use a connection oriented protocol to register itself with the DA.

When a SA registers the same attribute class more than once for a service instance, the DA 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>
scope-list:           omitted (ie. no items, 0 bytes)
Attributes (if any):  (ATTR1=VALUE),KEYWORD,(ATTR2 = VAL1, VAL2)
```

In order to offer continuously advertised services, SAs 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
scope-list: DEVELOPMENT
Attributes: (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" and "service" 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
scope-list: ENTWICKLUNG
Attributes: (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 DAs.

Registrations of a previously registered service are considered an update. If such an attribute registration is performed in a protected scope (see Section 17.2), 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 19.1.)

The DA may return a server error in the acknowledgment. This error is carried in the Error Codes field of the service location message header. The SrvReg may result in the error codes described in Section 14.

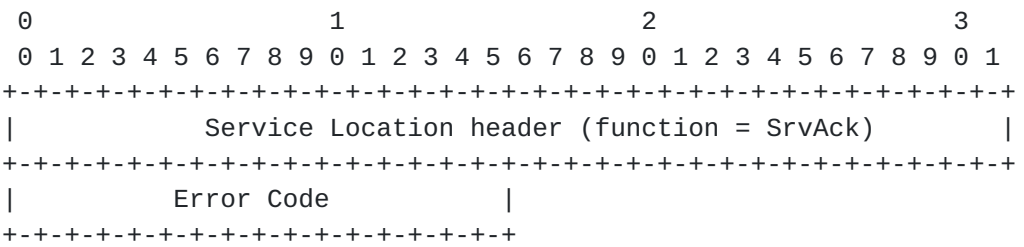
There are various rules concerning scopes: Which DAs a SA MUST register with and which DAs accept which registrations. See Section 17.1.

When the URL entry accompanying a registration also contains an authentication block (Section 4.4), the DA MUST perform the indicated authentication, and subsequently indicate the results in the Service Acknowledgement message.

14. 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.



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 SrvRqst which is set to have a scope which it does not support.

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.

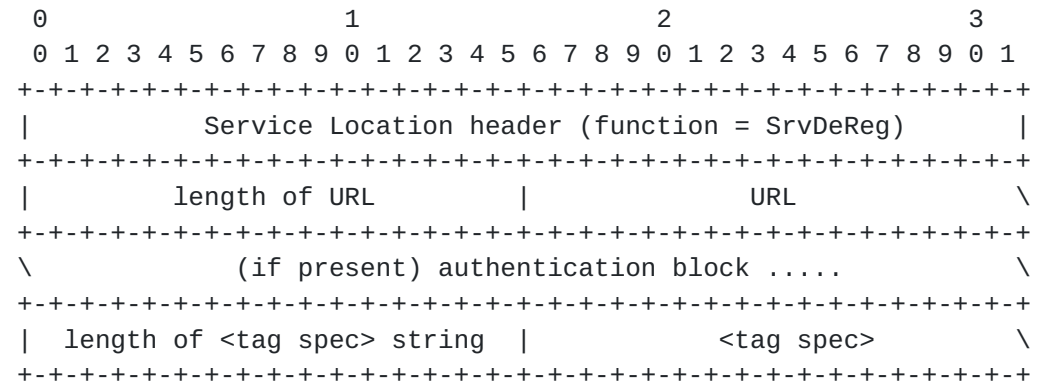
AUTHENTICATION_FAILED
If the registration contains an authentication block which fails to match the correct result as calculated (see Section 4.4) over the URL or attribute data to be authenticated, the DA will return this error.

INTERNAL_ERROR
A DA will return this if it cannot satisfy a request due to an internal failure. This might occur, for example, if the DA could not allocate sufficient resources. The SA MUST assume its request was not satisfied.

If the DA accepts 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 DA creates a new entry, and sets the 'F' bit in the Service Acknowledgement returned to the SA.

15. Service Deregister Message Format

When a service is no longer available for use, the SA must deregister itself from DAs that it has been registered with. A service uses the following PDU to deregister itself from all scopes the service was registered in.



The SA should retry this operation if there is no response from the DA. The DA 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 DA. The Error Code in the Acknowledgment of the Service Deregistration may have the same values as described in Section 14.

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 the Language Tag is omitted or 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 authentication block calculated over the URL data, the timestamp, and the length of the authentication as explained in Section 4.4. 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.

16. Directory Agents

16.1. Finding Directory Agents

A UA or SA 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 [11, 1] may use it to obtain a DA's address. DHCP options 78 and 79 have been assigned for this purpose [17]. SLP Agents SHOULD be configurable using DHCP.

The third way to discover DAs is dynamically. This is done by sending out a Directory Agent Discovery request (see Section 6.2).

Lastly, the agent may be informed passively as follows:

When a DA first comes on-line, and periodically, it sends an unsolicited DA Advertisement to the Service Location general multicast address. See Section 21 for details. If a DA supports a particular scope or set of scopes these are placed in its DAAdvert.

When a SA or UA first comes on-line it must issue a Directory Agent Discovery Request unless it is using static or DHCP configuration, as described in 6.2.

A SA registers information with ALL newly discovered DAs when either of the above two events take place.

Once a UA becomes aware of a DA it will unicast its queries there. In the event that more than one DA is detected, it will select one to communicate with.

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.

17. 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 in a department can be configured to request services from a particular scope. A scope is not an attribute of the service instance, because it is completely independent of the characteristics of the service instance.

Services in a scope effectively offer their service only to users that indicate that particular scope in their user requests. Just as sets can be non-disjoint, services can belong to several scopes. Users, on the other hand, are not viewed as belonging to particular scopes, but instead as having access to services in one or more scopes. This access can be acquired in several ways, as further detailed below.

Services that do not belong to any scope effectively offer their service to any user agent. Since a user agent selects its desired service by specifying the desired attributes of that service, any service satisfying those attributes will satisfy the needs of the user agent. Thus, a user agent typically does not care about which service agent provides the needed service. Cases in which this is not true require the user agent to specify that only services from the specific scope are desired, but such cases always arise from administrative considerations, and not from the specific values assigned to service attributes. The 'S' bit is supplied by user agents requiring the exclusion of services not assigned to the specific requested scope or scopes.

A site network that has grown beyond a size that can be reasonably serviced by a few DAs can use the scope mechanism. DAs may be configured with a scope list, representing the administrative categories which they serve. 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 administration Block 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.

DAs advertise their available scopes, together with a "service:directory-agent:" URL indicating its location. These advertisements are sent periodically starting as soon as the DA first becomes available. The UA and SA may also solicit these DA

advertisements when they first come on line. Thus, UAs and SAs are consistently informed of available scopes provided by scoped DAs.

UAs may select or be configured with one or more scopes to use. This selection may be automated through the use of DHCP or it may be done by the individual client application or service programmatically. UAs send all requests to DAs which support the scope or scopes they select. A SA is configured with a scope (or scopes) in which to register. It MUST register with all DAs in that scope that it can discover. Failure to be comprehensive in registration according to this rule may mean that the service advertisement may not be available to all UAs.

17.1. Rules Governing Scopes

For the sake of illustration, imagine that a service is a candy M&M, and that scopes divide the candy up into sets all containing the same color. Users want candy, but some users are more particular than others, and some users are constrained by their administration to only eat candy of a particular color.

Rules for User Agents, for various combinations of settings of the 'S' bit, and contents of the <scope-list>:

<'S'=on, scope-list="">	I want colorless M&Ms.
<'S'=on, scope-list="green">	I want green M&Ms.
<'S'=off, scope-list="green">	I want green M&Ms, or else colorless M&Ms.
<'S'=off, scope-list="">	I want colorless M&Ms.

Rules for Service Agents, for various combinations of settings of the 'S' bit, and contents of the <scope-list>:

<'S'=on, scope-list="">	I am a colorless M&M.
<'S'=on, scope-list="green">	I am a green M&M, and always appear colorful.
<'S'=off, scope-list="green">	I am a green M&M, but I'm happy to appear colorless.
<'S'=off, scope-list="">	I am a colorless M&M.

Rules for Directory Agents, for various combinations of settings of the 'S' bit, and contents of the <scope-list>:

<'S'=on, scope-list="">	I am a jar for colorless M&Ms.
<'S'=on, scope-list="green">	I am a jar for green M&Ms only.
<'S'=off, scope-list="green">	I am a jar for green M&Ms, but I'm happy to accept colorless.
<'S'=off, scope-list="">	I am a jar for colorless M&Ms.

The intention motivating this algorithm for satisfying scoped service requests is to allow a smooth transition between administration of unscoped service domains into scoped service domains. Notice that if a DA is installed and configured to offer scoped service, that user agents sending requests to that DA will typically be able to find the existing unscoped services until the services are configured for scope membership. This should enable system administrators to become more familiar with the scope model without the need for "flag days" or discontinuous changeovers of services.

If it is intended that all user agents be configured to request services from particular scopes, then while the user agents receive the necessary scope configuration information they will work as before in the unscoped administration until the time finally arrives when all user agents are configured with scopes. After that time, all service agents can be configured to register their services with the 'S' bit set, and the user agents will all continue to work as before.

17.1.1. Scope Strings in SLP Messages

Scope strings are found in several service location messages. The <scope-list> takes the form of a <String -List > (see Section 20) of scope names. The only characters which are reserved in scope strings are commas ',', asterisk '*' and the backslash character '\'.

If this scope list is omitted, the message is said to be 'not scoped'.

will explicitly exclude unscoped requests, as will be explained below. The scope list may include '*' when multicasting requests to SAs, which specifies that scopes will be ignored for the purposes of handling the request. The list may also include any other valid scope string.

17.1.2. Registration

Services which are registered with a non-null scope list are considered 'scoped'. A service registration which has no scope string is considered unscoped. That means it will be registered with all DAs which do not explicitly refuse unscoped registrations. A DA does this by setting the 'S' bit in its DAAdvert.

A DA may be configured to be unscoped or scoped. An unscoped DA accepts only unscoped registrations. SAs MUST register all unscoped services with an unscoped DA. By default, a scoped DA will accept unscoped registrations. A DA MUST be configurable to deny unscoped registrations.

Scopes determine which requests a DA will accept. A DA 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. SAs MUST register scoped services with every DA they are aware of, that supports the service's scopes.

DAs return a SrvAck in response to a SrvReg or SrvDeReg as defined in Section 13 and 15.

SA registration with DAs occurs with built in delays, depending on whether the DA was discovered actively (see Section 21.1) or passively (see Section 21.2).

17.1.3. Query Handling

In every case, the rules specified in Section 17.1 apply.

UAs MUST use scoped DAs in preference to unscoped DAs, and unscoped DAs in preference to multicasting requests to SAs. UAs MUST use scoped requests when making requests of scoped DAs. UAs SHOULD use requests with no scope when multicasting requests to SAs. UAs MUST use requests with no scope when sending a request to an unscoped DA.

17.1.3.1. DA Query Handling

An unscoped DA accepts only unscoped requests. It will send the corresponding reply with a SCOPE_NOT_SUPPORTED error if a scoped service registration is received.

A scoped DA MUST accept only requests which have the scope of the DA or where the request is unscoped (ie. the <scope-list> is omitted). The DA will respond with information which matches the request in the scope of the request AND unscoped service information. If the

request has the 'S' bit set in the message header, the DA will respond only with scoped service information.

If a scoped DA receives a request which does not include a supported scope in its <scope-list>, or includes the string "*" the DA replies with a SCOPE_NOT_SUPPORTED error in the reply corresponding to the request. This rule does not apply DA Discovery requests, which are described in Section 6.2.

DAs will return a reply with 0 results if they support the request (ie. do not return an error) and they have no matches to the request.

17.1.3.2. SA Query Handling

SAs apply the same rules for matching requests except that if there is no match they will drop the request if there are no matches. They will never reply with a SCOPE_NOT_SUPPORTED error.

SAs which receive a request where the <scope-list> is set to the string "*" does not apply scope matching rules to the request. This may mean that a SrvRply or AttrRply will not include digital signatures for services which registered in a protected scope. In order to receive these digital signatures in the reply, the UA MUST specify the protected scope explicitely in the Service Request. See 17.2.

17.2. 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 UA wishes to be sure that SAs are authorized to provide the service they advertise, then the UA should request services from a protected scope which has been configured to have the necessary authentication mechanism and keys distributed to the SAs 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 SAs 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 DAs as well as malicious SAs.

Note that services in protected scopes provide separate authentication for their URL entry, and for their attributes. This is so that the URLs in SrvRply message can be authenticated (see Section 7). This is the common case. In interactive mode, a user may wish to obtain the set of all attributes a service supports using an AttrRqst. The AttrRply for this request, made in a protected scope, will include a digital signature over the attributes (see Section 11).

17.2.1. Protected Scope Rules

Protected Scopes make use of structured Authenticator Blocks. These are described in Section 4.4.

DAs and SAs are configured for use in a protected scope by configuring them with the Public and Private keys of the protected scope. UAs are configured for use in a protected scope by configuring them with the Public key of the protected scope. See Section22.

- SAs which have been configured with a protected scope MUST include Authentication Blocks with all SrvReg and SrvDeReg messages sent to DAs.

- SAs which have been configured with a protected scope MUST include URL Authentication Blocks in SrvRply messages and Attribute Authentication Blocks in AttrRply messages if the corresponding SrvRqst or AttrRqst included the protected scope explicitely in the <scope-list> in the request.

- SAs must reject DAAdverts they receive for a protected scope which do not contain an Authenticator Block which the SA can verify.

- DAs which have been configured with a protected scope must verify the signatures of SrvReg and SrvDeReg messages from SAs. The DA returns a SrvAck reply. If the signature fails, the DA returns an AUTHENTICATION_FAILED error. If the BSD of the Authentication Block is not supported by the DA, the DA returns an AUTHENTICATION_ALGO_UNKNOWN error. If the SA fails to include an authenticator where it should, the DA returns an AUTHENTICATION_ABSENT error.

- DAs which have been configured with a Public Key for a protected scope MUST include URL Authentication Blocks in SrvRply messages

and Attribute Authentication Blocks in AttrRply messages. They must also include Authentication Blocks in their DAAdvert messages (see Section 12.)

- If the UA is configured with a protected scope, and if the UA receives a SrvRply, AttrRply or DAAdvert from that protected scope, the UA MUST verify the reply or advertisement before accepting it.

18. Language Internationalization Issues

18.1. Language Tags and Dialects

Service location messages often include a Language Tag in the header. A SrvDeReg message MAY omit a Language Tag if the service is to be deregistered in ALL languages it has been registered in. If the Language Tag is omitted, the Language Tag length in the header is set to 0.

If the Language Tag used includes a dialect, the dialect is to be used the following way: Requests which can be fulfilled by matching a language and dialect will be preferred to those which match only the language portion. Otherwise, dialects have no effect on matching requests.

18.2. Scope Strings are not Language Specific

A scope string, which is used to configure DAs and which are used by both UAs and SAs for various operations (see Section 17.1) is NOT language specific. Thus, the language declared in a SrvRqst used for DA discovery has no effect on whether the request predicate matches the scopes of the DA or not. The DAAdvert does not include a Language Tag as the scope strings are not language specific.

18.3. Declaring the language of registrations

All Service Registrations declare the language in which the strings in the service attributes are written by specifying a Language Tag 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.

A Service is fully deregistered if the URL is given in the Service Deregister message without any attribute information or Language Tag sent in the message.

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 SA.

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.

18.4. Translation of Attribute Strings

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.

18.5. Declaring the language of a Request

Service and Attribute requests specify a requested language in the message header. The DA or SA 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 MAY be sent in the default language (which is English) on a best effort basis.

In the case where there are services registered in the type requested in a SrvRqst (or AttrRqst), but not in a language which the service information supplies, the following two rules apply: If the 'monolingual bit' flag in the header is set, a SrvRply (or AttrRply) is returned with the error field set to LANGUAGE_NOT_SUPPORTED. A SrvRqst may include a predicate (or an AttrRqst may contain a tag list) in an unsupported language. If any of the attribute strings are not language literal (see Section 19.1), the SrvRply (or AttrRply) 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.

When there are several replies returned in one message and the reply includes attributes which were registered with separate dialects, the dialect portion of the Language Tag in the reply is dropped. This could occurs primarily with an attribute reply.

19. Substitution of Character Escape Sequences

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.'

Characters which are reserved as part of the encoding of attribute tags or values MUST be escaped if they are included in Attribute Lists or Predicates. Any character in these strings MAY be escaped.

The escape mechanism uses the reserved backslash character '\' (ASCII 0x5c). This character is followed by two hexadecimal digits of the escaped character. Note that some characters are multibyte, using UTF8 encoding.

Examples:

Character to encode	Unicode	UTF8	SLP Escaped value
','	0x0029	0x0029	\29
e accent aigue	0x0039	0xc3a9	\c3\a9
not equals glyph	0x2260	0xe289a0	\e2\89\a0

Characters in URLs which are reserved according to the URL syntax specification must be escaped using the URL escape character convention. In this case a 'percent' is followed by the hex encoding of the escaped character exactly as defined above for the 'slash' character. The current URL syntax specification limits all encoded characters to a subset of the ASCII character set. The update to this defines a way to transmit UTF8 encoded characters outside of the ASCII range.

19.1. Language-Independent Strings

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

Reserved String	Section	xDefinition
SCOPE	3, 16	Used to limit the matching of requests.
SERVICE	7, 13	The URL scheme of all Service Location information registered with a DA or returned from a Service Request.
<srvtype>	20.1.1	Used in all service registrations and replies.
domain names	20.3	A fully qualified domain name, used in registrations and replies.
IANA	3.7	The default naming authority.
LOCAL	17	Reserved.
REMOTE	17	Reserved.
TRUE	20.4	Boolean true.
FALSE	20.4	Boolean false.
SERVICE-TYPE	6	Standard term in predicates.

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	SrvRqst
Service Request <predicate>	6.4	SrvRqst
URL	[12]	SrvReg, SrvDeReg, SrvRply
<attr-list>	20.2	SrvReg, SrvRply, AttrRply
<Service Registration Information>	13	SrvReg
<Service Deregister Information>	15	SrvDeReg
<Service Type String>	20.1.1	AttrRqst
<String-List>		DAAdvert, SrvTypeRply attr vals
<Select-List>		AttrRqst, SrvDeReg

20.1. Previous Responders' Address Specification

The previous responders' Address Specification is specified as a <String-List> of <addr-spec> terms.

The Address Specification is the address of the DA or SA which supplied the previous response. The format for Address Specifications in Service Location is defined in Section 20.3. 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
```

SAs or DAs which are listed in a Previous Responders list in a multicast Service Location request will silently discard the request.

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 D should be used.

20.1.1. 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.

The Service Type of a service: scheme URL is encoded as the string up to and including the last ":". This includes the Naming Authority string.

URL	Service Type String
service:lpr://biglaser.dog.com	service:lpr:
service:wp:http://dir.cat.org	service:wp:
service:game.cs312://morb.lug.edu	service:game.cs312:
ftp://ftpserv.mouse.gov	ftp:

20.1.2. String List

String lists transmitted by the service location protocol are items with a ',' delimiter.

```
str-list = str-item / str-item ',' str-list
```

The definition of <str-item> depends on the string list. For a DAAdvert, the <str-item> is a scope attribute value (see <strval> in Section 20.2). The other important case is described in Section 20.1, which may be included with SrvRqst, AttrRqst and SrvTypeRqst messages.

20.1.3. Select List

Select Lists are String List used in AttrRqst and SrvDeReg messages. The <str-item> is a <tag-filter> with the following syntax:

```
tag-filter = simple-tag / substring
simple-tag = 1*filt-char
substring = [initial] any [final]
initial = 1*filt-char
  any = '*' *(filt-char '*')
final = 1*filt-char
filt-char = Any character excluding <reserved> and <bad-tag> See
      Section 20.2.
```

20.2. 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 ')' / attr-tag
attr-val-list = attr-val / attr-val ',' attr-val-list
attr-tag = 1*safe-tag
attr-val = intval / strval / boolval / opaque
intval = [-]1*DIGIT
strval = 1*safe-val
boolval = "TRUE" / "FALSE"
opaque = "(" 1*DIGIT ":" 1*r64chars ")"
safe-val = Any character except reserved.
safe-tag = Any character except reserved, star and bad-tag.
reserved = '(' / ')' / ',' / '\' / '!'  / '<' / '=' / '>' / 'tilde'
bad-tag = CR / LF / HT
 star = '*'
r64chars = DIGIT / ALPHA / '=' / '/' / '-'
```

An <attr-list> must be scanned prior to evaluation for all occurrences of the escape character '\'. Reserved characters in attributes placed in attribute lists must be escaped. All escaped characters must be restored to their value before used for string matching. See Section 19.

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

```
(OPERATOR=Joe Javaman)
(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 takes the value "RED, WHITE, BLUE" (the commas in the value string are escaped). Priority, on the other hand, takes the three values "L", "M", and "H". There are several other examples of attribute lists throughout the document.

20.3. 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. Passwords SHOULD NOT be sent with SLP messages (as cleartext) unless an IPSec ESP is used for delivery of the SLP message.

Address specification in Service Location is consistent with standard URL format [5].

20.4. Attribute Value encoding rules

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

The syntax of attribute values is described in Section 20.2.

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 strings are encoded in. Boolean attributes can take only one value.

- Integer values are expressed as a sequence of DIGITs. 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>           ::=  1*DIGIT
<radix-64-data> ::=  radix-64 encoding of the original data
```

<len> is a 16-bit binary number expressed as a decimal string. For example, if the opaque value is 34 bytes long before encoding the <len> field would be the string "34".

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 2045 [6], Section 6.8.

The Opaque value encoding includes otherwise reserved characters in attribute values: '(', ')'. It may include another reserved character: '='. These characters are escaped in SrvRqst predicates. For example "(&(service-type=thing)(att=\(3:AAAA\)))" is a query for a service of type "thing" which has an attribute "att" with a value of a 3 zero bytes.

21. Protocol Timing Rules

There are four protocol actions which require timing rules. Active DA Discovery, Passive DA Advertising, reliable unicast requests to a DA and multicast/convergence.

21.1. Active DA Discovery

After a UA or SA restarts (say, after rebooting of a system or loading of the network kernel), their initial DA discovery request should be delayed for some random time uniformly distributed from 0 to CONFIG_START_WAIT seconds.

The UA or SA sends the DA Discovery request using a SrvRqst, as described in Section 6.2 and receives a DAAdvert as a reply, as described in Section 12.

DA Discovery requests should be sent according to CONFIG_DA_RETRY. The second and third transmission of the DA Discovery request MUST include a previous responders list of all DAs which have responded. See Section 20.1.

SAs which discover DAs actively must wait CONFIG_REG_ACTIVE seconds before registering their services.

21.2. Passive DA Advertising

Upon startup, then every CONFIG_DA_BEAT seconds a DA will multicast (or broadcast) an unsolicited DAAdvert. This will ensure that eventually it will be discovered by all applications which are concerned.

All SAs which receive the unsolicited DAAdvert should examine its Sequence Number. If the DA has never before been heard from or if the Sequence Number is less than it was previously and less than 256, the SA 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 DA, after waiting a random interval CONFIG_REG_PASSIVE.

While it might seem advantageous to have frequent heartbeats, this generates traffic throughout the network in proportion to the number of DAs. Thus, CONFIG_DA_BEAT should be specified with a value in seconds long enough to prevent routine protocol operations from using any significant bandwidth.

21.3. Reliable Unicast to DAs

If a DA fails to respond to a unicast UDP message in CONFIG_DA_RETRY seconds, the message should be retried. If a DA fails to respond after CONFIG_DA_MAX seconds, the UA or SA should use a different DA.

DA addresses may be cached from previous discovery attempts, preconfigured, or by use of DHCP (see Section 16.1).

If no such DA responds, DA discovery should be used to find a new DA. Care should be taken not to do Active DA Discovery more than once per CONFIG_DA_FIND seconds.

21.4. Multicast/Convergence

This algorithm is used by UAs to send requests to SAs using the Service Location General Multicast Address. The algorithm is used by both UAs and SAs to send DA Discovery requests to DAs using the DA Discovery Multicast Address.

Requests to SAs are multicast repeatedly (with a recommended wait interval of CONFIG_MC_RETRY) until there are no new responses, or CONFIG_MC_MAX seconds have elapsed. DA discovery requests use different timing for repeated requests, CONFIG_DA_RETRY.

Repeated requests include a previous responder list, see Section 20.1.

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.

22.1. 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_KEEP_RPLY	4.2	1 minute	Cache replies by XID.
CONFIG_LIFETIME	4.5	10800 seconds (3 hours)	registration Lifetime, after which ad expires
CONFIG_MC_RETRY	4.2	each second, backing off gradually	Retry multicast query until no new values arrive.
CONFIG_MC_MAX	6	15 seconds	Max time to wait for a complete multicast query response (all values.)
CONFIG_START_WAIT	13	3 seconds	Wait to perform DA discovery on reboot.
CONFIG_DA_RETRY	6.2	2 seconds	Retransmit DA discovery, try it 3 times.
CONFIG_DA_MAX	6.2	6 seconds	Give up on requests sent to a DA.
CONFIG_DA_BEAT	16.1	3 hours	DA Heartbeat, so that SAs passively detect new DAs.
CONFIG_DA_FIND	6.2	900 seconds	Minimum interval to wait before repeating Active DA discovery.
CONFIG_REG_PASSIVE	16.1	1-3 seconds	Wait to register services on passive DA discovery.
CONFIG_REG_ACTIVE	13	1-3 seconds	Wait to register services on active DA discovery.
CONFIG_CLOSE_CONN	3.10	5 minutes	DAs and SAs close idle connections.

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.

Directory Agent Address
: The DA 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 16.1. 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 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 MUST able to acquire keys to authenticate data sent by services along with their advertised URLs for services within the protected scope. SAs and DAs require private keys for the scope, where all entities require public keys for the scope. See Section 17.2.1 which defines how these keys are used. By default, service agents use "md5WithRSAEncryption" [4] to produce the signed data, to be be included with service registrations and deregistrations (see appendix B, 4.4). This authentication data could be verified by user agents and directory agents that possess the corresponding

public key.

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

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

A SA 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 SA 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.

23. Security Considerations

The Service Location Protocol provides for authentication of Service Information registered by SAs and DA Advertisements. This allows UAs to obtain information about services with confidence that is reasonably complete and accurate.

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 use protected scopes.

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 UAs, this could be as simple as supplying the public key of a Certificate Authority. See Appendix B.

24. Protocol Requirements

In this section are listed various protocol requirements for UAs, SAs, and DAs.

24.1. Directory Agent Requirements

Protocol Feature	Section	Requirement Level
Announce using multicast on startup.	21.2	MUST
Announce using multicast, regularly.	21.2	MUST
Reply to unicast requests from UAs.	17.1.3.1	MUST
Reply to multicast DADiscovery requests.	6.2	MUST
Support TCP when overflow occurs.	3.10.1	MUST
Accept de/registration from SAs.	17.1.2	MUST
Age out expired service registrations.	4.5	MUST
Reply to message using DA scoping rules.	17.1.3.1	MUST
Obey language rules for de/reg, requests.	18	MUST
Announce and verify using pro. scopes.	17.2.1	MUST
Be able to ignore unrecognized options.	4.1	MUST

24.2. Service Agent Requirements

Protocol Feature	Section	Requirement Level
Perform Active DA discovery on start up.	21.1	MUST
Perform Passive DA discovery always.	21.2	MUST
Forward Registrations discovered DAs.	17.1.2	MUST
Forward Deregistrations to DAs if needed.	17.1.2	MUST
Age out expired service registrations.	4.5	MUST
Obey pro. scope rules: in all messages.	17.2.1	MUST
Obey scope rules: forwarding to DAs.	17.1	MUST
Obey scope rules: responding to UAs.	17.1	MUST
Obey language rules on reply or register.	18	MUST
Support TCP for overflowed SLP messages.	3.10.1	MUST
Respond to multicast & unicast SrvRqst.	3.10	MUST
Also respond to AttrRqst and SrvTypeRqst.	3.10	SHOULD
Be able to ignore unrecognized options.	4.1	MUST

24.3. User Agent Requirements

Protocol Feature	Section	Requirement Level
Perform Active DA discovery on start up.	21.1	MUST
Perform Passive DA discovery always.	21.2	SHOULD
Obey rules: preferring DAs to SAs, etc.	5	MUST
Obey scope rules: where to send requests.	17.1	MUST
Obey pro. scope rules: reject bad info.	17.2.1	MUST
Support TCP for overflowed SLP messages.	3.10.1	MUST
Be able to ignore unrecognized options.	4.1	MUST
Send SrvRqst to SAs using multicast.	21.4	MUST
Send SrvRqst to DAs using unicast.	21.3	MUST
Send SrvTypeRqst and AttrRqst messages.	5	SHOULD
Send requests to SAs using unicast.	5	MAY

24.4. Common Requirements for all SLP Agents

Protocol Feature	Section	Requirement Level
Static Configurability.	22	MUST
DHCP Configurability.	16.1	SHOULD
Support protected scope configuration.	22	SHOULD
For protected scopes, support md5/RSA.	RFC 1423[4]	MUST
Support UTF-8 character encoding.	4	MUST

25. Non-configurable Parameters

IP Port number for unicast requests to DAs:

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
AUTHENTICATION_ABSENT	6
AUTHENTICATION_FAILED	7
AUTHENTICATION_ALGO_UNKNOWN	8
PROTOCOL_V1_REJECTED	9
EXPECTED_ATTRIBUTE_MISSING	64
INTERNAL_ERROR	128

26. Acknowledgments

Early work on this protocol was done by Leo McLaughlin, Mike Ritter and Scott Kaplan. Charlie Perkins and Harry Harjono's Resource Discovery Protocol was merged with the Service Location Protocol. Jeff Schiller assisted in shaping the security architecture specified in this document. This protocol has benefited from the techniques, focus and minimal requirements of the Ping Discovery Service (PDS) from Intel in its recent revision.

A. Version 2 Notes

This document revises the Service Location Protocol as specified in RFC 2165. There are some open issues which are resolved here. Some of these are protocol corrections (which result in a change in the on the protocol as it is exchanged over the wire.) Some of the changes to the document are merely clarifications and document restructuring.

Changes since draft-ietf-svrloc-protocol-v2-00.txt:

- URL length was 1 byte, now it is 2.
- Deregistration without language tag or attr tags fully deregisters a service.
- We clarified the monolingual bit, so that it is clear that LANGUAGE_NOT_SUPPORTED is returned only when there are services available, but not in an available language.
- We added a new error: INTERNAL_ERROR, for the case where a DA or SA cannot respond.
- We restored the header to as close a form to the v1 version as possible and noted why it could not be the same.
- We clarified the 'S' bit.
- Michael Day added as an editor/author.
- PDS added to the acknowledgement section.
- ISOC copyright header added.
- We added 'service-type' to the list of reserved, non-translatable strings.
- We fixed the bibliography.
- We added a section on minimal UA and SA requirements.
- We rewrote the introduction and added an appendix on SLP and management.
- We updated the requirements tables at the end.

Changes between RFC 2165 and version-00 of this document:

Clarifications:

SrvTypeRqst definition
: SrvTypeRqst does not include a list of previously received service types as was erroneously indicated.

Opaque value syntax
: The ``(`` and ``)'' characters enclosing an opaque value are to be taken literally. The length of the opaque value is to be encoded as a decimal number.

Scopes
: Are NOT language specific strings. All scope rules have been coalesced into a single section. The rules for when to use unscoped requests and registrations have been clarified as well as the consequences of receiving them and the behavior of unscoped DAs.

Requirements
: The requirements section has been redone in a simpler matrix style, with references to the protocol section, reflecting *all* protocol requirements.

Wild Card values in search filters
: Wild card values are only intended to be used for string MATCHING not string COMPARISONS.

Service Type Reply strings
: The definition of the reply strings needs to be clarified and allow for arbitrary URL schemes.

Security considerations
: The security section has been updated to reflect the changes made to the certificate format section and the ability to sign DAAdverts.

AttrRqst select-list length
: The length of the select-list is the length of the select-list string, which is the common convention in SLP.

Modifications:

Crypto Algorithm field in Service requests
: The UA requires a way to specify which crypto algorithm it can use, if it is not the default, so the DA or SA can determine what to use.

Version Number
: Since the wire protocol has changed, the version number of the protocol takes the version number 2.

Character Encoding
: Character encodings are all be in UTF8 [21]. This conforms to current IESG recommendations and simplifies the protocol.

Service Request Predicates
: Service Request Predicate grammar have been replaced by the request grammar used by LDAP string search filters. Some conventions are necessary to ensure proper structure of requests. The simple 'list format' queries is no longer supported.

Character escapes in string encoding
: To be consistent with the LDAP string search filters syntax and to simplify the encoding rules in SLP, all character escapes in SLP string based messages reserve the backslash character as an escape for reserved characters. Character escapes in URLs conform to standard URL syntax conventions.

Fresh Bit Handling
: The Fresh bit is transmitted by SAs to DAs to indicate freshness, not returned by DAs to SAs in the acknowledgement of registration. This allowed for a race condition.

URL handling
: SLPv1 only supports service: URLs. It now supports arbitrary URLs provided that they follow standard URL syntax. This support modifies SrvRply, AttrRqst, SrvReg, DAAdvert and SrvDeReg.

Further Modifications:

DAAdvert format

The DAAdvert message contains new information: It indicates whether it was sent as the result of a service request or at the DA's own initiative. It can also carry a digital signature of the DAAdvert. The DA may also mark some of the scopes it advertises in as 'protected' and provide a pointer to the protected scope's certificate. DAAdverts no longer override the XID in the header, but instead carry a 'DAAdvert sequence number'.

Request Flagging

The header indicates whether a request was multicast or unicast. This aids interpretation of the request by SAs and DAs implemented on platforms which receive all datagram requests from the same socket and cannot distinguish between the 'destination' address of the datagram they received.

String Interpretation

Strings are interpreted in line with other string based query protocols: Interior white space is folded to a single white space rather than interpreted literally for string matching.

Language Tagging

The reserved Dialect field is used for inclusion of RFC 1766 language tags (or whatever obsoletes them.) This is in accord with the new guidelines for internationalization.

SLP Certificate format

The certificate format needs an additional field indicating the name of the CA. There also needs to be text which encourages the use of standard certificate formats if possible (X.509, etc.)

Requirements

The requirement that SAs handle SrvTypeRqst and AttrRqst has been relaxed from a SHOULD to a MAY. The requirement that a UA be able to issue a SrvTypeRqst and AttrRqst has been relaxed from a MUST to a SHOULD. The requirement that UAs be able to use multicast/convergence has been relaxed from a MUST to a SHOULD (provided that DHCP is used for DA discovery!) This allows conformant implementations to be much more lightweight.

Service Location Extension Options

These options allow for the protocol to be enhanced without

requiring the main protocol specification standardization to be derailed.

Open Issues:

Service Specific Multicast Addresses

Use of Service Specific Discovery Multicast Addresses as defined in RFC 2165 is deprecated until a multicast address range allocation mechanism is defined using Administratively Scoped Multicast Addresses. We need to assign the range of service specific multicast addresses for use by UAs and SAs.

Lifetime may be too short

It may be that timeouts for soft state are too short. We may want to make the URL entry's Lifetime field be longer than 2 bytes worth of seconds.

Allow the tag list in SrvDeReg to include wildcards

This would allow for deletions of attributes with patterns, such as all entries with a common prefix. This facilitates efficient name deletion based on a hierarchical structure.

Include a MIN, MAX and BEST operator for queries

MIN and MAX would be used for integer query terms - to allow the service with the smallest or largest value for a given attribute to be returned. BEST would be a string match which would return the service(s) for which the longest substring match could be found.

Allow increasing ring multicast discovery

This would allow the 'nearest' services to be discovered.

Use multicast based de/registration of services

This would eliminate a scaling issue in the protocol but would require the use of MTCP-like techniques so that SAs could moderate their refresh intervals so that SLP registration take only a fixed amount of network bandwidth.

LDAP search filters

Should LDAPv2 filters [13, 20, 14] be used, since they rely on Draft Standard protocols? Or, should SLP use the new LDAPv3 filters (which have not yet gone to PS)? The latter handle internationalization far better!

B. SLP Certificates

Certificates may be used in SLP in order to distribute the public keys of trusted protected scopes. This section defines a very simple format for certificates which provides only the most basic functionality required for deploying protected scopes. If a public key infrastructure has already been deployed, for instance using X.509 certificates, these SHOULD be used instead. Implementation of SLP Certificates is entirely OPTIONAL.

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-auth=" 1*ALPHANUMERIC   CRLF
"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 2045 [6]. 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-auth identifies the authority who created the certificate. This enables a very unsophisticated mechanism for enabling the use of multiple certificate authorities: An end system may be configured with public keys for multiple certificate authorities. In each case the public key will be identified by a 'certificate-auth' string as part of the configuration. When certificates are obtained, they may use the 'certificate-auth' field to determine which public key to use to verify the certificate's validity.

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 distribute 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 4.4) is digestified using MD5 [18] to create a digital signature, then encrypted by RSA with the protected scope's private key. The output of RSA is used in the authentication data field of the authentication 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 authentication 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. Scaling and Deployment of the Service Location Protocol

The Service Location Protocol is meant to be deployable in a variety of network configurations. The protocol is ideal for 'plug and play' networks, which have the minimum of services offered. It also will scale to sites, with many routed networks, even over wide-area low bandwidth links.

The key to SLP's scalability is how it is configured in the default case and how it can be configured for use in larger deployments.

The default configuration for SLP requires it to discover DAs and if possible, to use them. This ensures that even in a larger deployment SLP will not use multicast or broadcast more than necessary. Thus SLP will not require much network bandwidth for its operation.

If there are no DAs, as would be the case in the simplest configuration, UAs will multicast (or broadcast) their requests to SAs. This allows a network of client applications and services to be set up with no configuration; no DNS, no DHCP, and so on. This is a useful feature for extremely small networks, such as in a conference room.

DAs are provided for scalability. The protocol works identically whether DAs are present or not (UAs issue the same requests to either SAs or DAs, and get the same replies.) The only difference is efficiency and scalability.

The scoping feature of SLP provides further scalability. Services which are scoped will be attracted to the subset of DAs which support their scope. UAs will go to the DAs which supply service information in the scope they are configured to use, or the scope which the user selects if the UA is not configured with a scope.

In some cases multicast cannot be used to discover DAs. In others, administration Blocks may decide that they would prefer to control the way DAs are used. Here, DHCP may be used to configure UAs and SAs with the location of DAs. This is particularly useful for configuring hosts which are networked via slow wide-area links. It would be a waste of scarce network bandwidth to use a multicast routing protocol to distribute DA advertisements over such a link.

There is no guarantee that a DA of a particular scope will contain the same service information as another DA in the same scope. This consistency is only achieved when all SAs which advertise services in that scope are aware of the same set of DAs. This may not be possible, if multicast routing does not pervade the entire site or if DHCP does not uniformly configure SAs with the same scope to use the same DAs.

Indeed, it may not even be desirable to allow such pervasive use of multicast nor even be required for useful 'best effort' service provision for there to be a consistent view of services from all DAs of the same scope. This is a decision which network administrators of SLP in large sites need to consider. The more consistency they wish to achieve between DAs, the more complicated their DHCP configuration will be, or the more SLP based multicast traffic they will have in their networks.

E. Using SLP For Network and Systems Management

When deploying SLP for purposes of network or systems management, there are some important points to consider. These include ease of administration, having a low impact on the network, and providing SLP support for the entire range of systems on the network. (Note that using SLP to discover managed systems will only find managed systems that support SLP.)

Network and systems management applications need to continue working when other network components are broken. Further, management applications frequently need to obtain information that has a very short lifetime. These considerations motivate some of the recommendations that are listed below.

The Service Location Protocol can provide a number of applications for network and systems management. For example:

- A manager can use SLP to discover managed systems without performing brute-force scanning of the network.

- A manager can select manageable systems based on specific attributes, such as support for a specific management protocol, agent, instrumentation group, and software distribution format.

- A managed system can use SLP to find its manager(s), to which the managed system can then send traps and issue trouble tickets.

- A managed system can use SLP to publish basic information about itself, such as its operating system vendor, type and version; its MAC address(es); operator information; management agent(s) hosted by that system, and so on.

- A manager can periodically search for new hosts using SLP.

A good strategy to follow when using SLP for manageability includes the following points:

- Peer-to-peer implementation. Each managed system, along with each management application, implements UA and SA functionality. The system advertises its management services. These service types are defined elsewhere, for example [10].

- Bootstrap capability. Each managed system, along with each management application can begin functioning with no knowledge of exisiting manageability services beyond a knowledge of its own UA. For example, a management application can, with no foreknowledge, use SLP to build a list of manageable stations and their basic capabilities.

This bootstrap capability requires the use of multicast or broadcast with convergence. Multicasting and broadcasting should be avoided when possible. However, bootstrap capability is a key aspect of systems management. (What happens when the DA is broken and needs to be managed?) Therefore, managers and managed stations should use DAs for discovery when they exist, but must also have the ability to perform multicast or broadcast with convergence.

- DAs for support of mobile and occasionally connected stations, as well as for scalability. DAs provide a good way to support the discovery of mobile and occasionally connected systems because they can act as SLP proxies for such systems, which may be connected via a slow link. In addition, DAs provide scalability to SLP.

F. Full Copyright Statement

Copyright (C) The Internet Society (1997). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implmentation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

References

[1] S. Alexander and R. Droms. DHCP Options and BOOTP Vendor Extensions. RFC 2132, March 1997.

[2] H. Alvestrand. Tags for the Identification of Languages. RFC 1766, March 1995.

[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, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738, December 1994.

[6] N. Borenstein and N. Freed. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, November 1996.

[7] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, March 1997.

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

[9] D. Crocker and P. Overell. Augmented BNF for Syntax Specifications: ABNF. RFC 2234, November 1997.

[10] M. Day. Manageability service: Schemes. November 1997. draft-ietf-svrloc-sysman-00.txt (work in progress).

[11] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997.

[12] E. Guttman, C. Perkins, and J. Kempf. Service Templates and service: Schemes. draft-ietf-svrloc-service-scheme-05.txt, November 1997. (work in progress).

[13] T. Howes. A String Representation of LDAP Search Filters. RFC 1960, June 1996.

[14] T. Howes, S. Kille, W. Yeong, and C. Robbins. The String Representation of Standard Attribute Syntaxes. RFC 1778, March 1995.

[15] D. Mills. Simple Network Time Protocol (SNTP) Version 4 for

IPv4, IPv6 and OSI. RFC 2030, October 1996.

[16] R. Moats and M. Hamilton. Finding Stuff (How to discover services). draft-ietf-svrloc-discovery-05.txt, October 1997. (work in progress).

[17] C. Perkins. DHCP Options for Service Location Protocol, May 1997. draft-ietf-dhc-slp-02.txt (work in progress).

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

[19] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service Location Protocol. RFC 2165, July 1997.

[20] W. Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. RFC 1777, March 1995.

[21] F. Yergeau. UTF-8, a transformation format of unicode and ISO 10646. RFC 2044, October 1996.

Authors' Addresses

Questions about this memo can be directed to:

	Erik Guttman Sun Microsystems Bahnstr. 2 74915 Waibstadt Germany	Charles Perkins Sun Microsystems 901 San Antonio Road Palo Alto, CA 94040 USA
Phone:	+49 7263 911701	+1 650 786 6464
Fax:		+1 650 786 6445
Email:	Erik.Guttman@sun.com	cperkins@sun.com

	John Veizades @Home Network 385 Ravendale Dr. Mountain View, CA 94043 USA	Michael Day Intel
Phone:	+1 650 569 5243	+1 801 763 2341
Fax:		
Email:	veizades@home.net	Michael_Day@ccm.ut.intel.com