Network Working Group
Request for Comments: 2608
Updates: 2165
Category: Standards Track

E. Guttman
C. Perkins
Sun Microsystems
J. Veizades
@Home Network
M. Day
Vinca Corporation

June 1999

Service Location Protocol, Version 2

Status of This Memo

Copyright Notice

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.

Table of Contents

1. Introduction

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. Traditionally, users have had to find services by knowing the name of a network host (a human readable text string) which is an alias for a network address. SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user supplies the desired type of service and a set of attributes which describe the service. Based on that description, the Service Location Protocol resolves the network address of the service for the user.

SLP provides a dynamic configuration mechanism for applications in local area networks. Applications are modeled as clients that need to find servers attached to any of the available networks within an enterprise. 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.

This document updates SLPv1 [RFC 2165], correcting protocol errors, adding some enhancements and removing some requirements. This specification has two parts. The first describes the required features of the protocol. The second describes the extended features of the protocol which are optional, and allow greater scalability.

1.1. Applicability Statement

SLP is intended to function within networks under cooperative administrative control. Such networks permit a policy to be implemented regarding security, multicast routing and organization of services and clients into groups which are not be feasible on the scale of the Internet as a whole.

SLP has been designed to serve enterprise networks with shared services, and it may not necessarily scale for wide-area service discovery throughout the global Internet, or in networks where there are hundreds of thousands of clients or tens of thousands of services.

2. Terminology

User Agent (UA)
A process working on the user's behalf to establish contact with some 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 the services.

Directory Agent (DA) A process which collects service advertisements. There can only be one DA present per given host.

Service Type Each type of service has a unique Service Type string.

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

Scope A set of services, typically making up a logical administrative group.

URL A Universal Resource Locator [8].

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 [9].

Syntax Syntax for string based protocols follow the conventions defined for ABNF [11].

Strings All strings are encoded using the UTF-8 [23] transformation of the Unicode [6] character set and are NOT null terminated when transmitted. Strings are preceded by a two byte length field.

<string-list> A comma delimited list of strings with the following syntax:

```
string-list = string / string `,' string-list
```

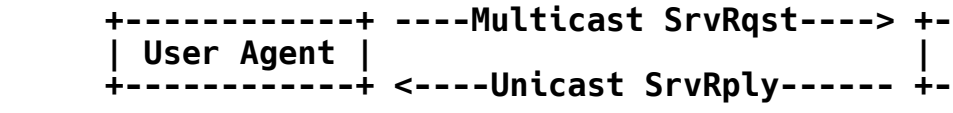
In format diagrams, any field ending with a \ indicates a variable length field, given by a prior length field in the protocol.

3. Protocol Overview

The Service Location Protocol supports a framework by which client applications are modeled as 'User Agents' and services are advertised by 'Service Agents.' A third entity, called a 'Directory Agent' provides scalability to the protocol.

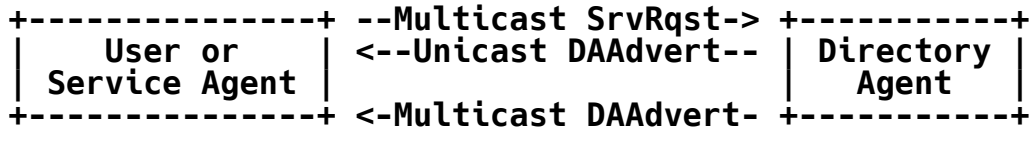
The User Agent issues a 'Service Request' (SrvRqst) on behalf of the client application, specifying the characteristics of the service which the client requires. The User Agent will receive a Service Reply (SrvRply) specifying the location of all services in the network which satisfy the request.

The Service Location Protocol framework allows the User Agent to directly issue requests to Service Agents. In this case the request is multicast. Service Agents receiving a request for a service which they advertise unicast a reply containing the service's location.



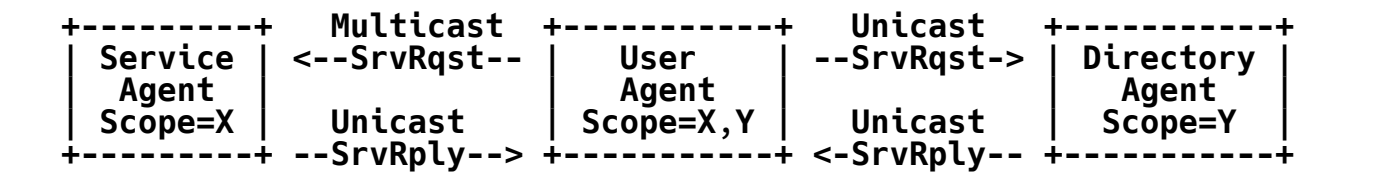
In larger networks, one or more Directory Agents are used. The Directory Agent functions as a cache. Service Agents send register messages (SrvReg) containing all the services they advertise to Directory Agents and receive acknowledgements in reply (SrvAck). These advertisements must be refreshed with the Directory Agent or they expire. User Agents unicast requests to Directory Agents instead of Service Agents if any Directory Agents are known.

User and Service Agents discover Directory Agents two ways. First, they issue a multicast Service Request for the 'Directory Agent' service when they start up. Second, the Directory Agent sends an unsolicited advertisement infrequently, which the User and Service Agents listen for. In either case the Agents receive a DA Advertisement (DAAdvert).



Services are grouped together using 'scopes'. These are strings which identify services which are administratively identified. A scope could indicate a location, administrative grouping, proximity in a network topology or some other category. Service Agents and Directory Agents are always assigned a scope string.

A User Agent is normally assigned a scope string (in which case the User Agent will only be able to discover that particular grouping of services). This allows a network administrator to 'provision' services to users. Alternatively, the User Agent may be configured with no scope at all. In that case, it will discover all available scopes and allow the client application to issue requests for any service available on the network.



In the above illustration, the User Agent is configured with scopes X and Y. If a service is sought in scope X, the request is multicast. If it is sought in scope Y, the request is unicast to the DA. Finally, if the request is to be made in both scopes, the request must be both unicast and multicast.

Service Agents and User Agents may verify digital signatures provided with DAAdverts. User Agents and Directory Agents may verify service information registered by Service Agents. The keying material to use to verify digital signatures is identified using a SLP Security Parameter Index, or SLP SPI.

Every host configured to generate a digital signature includes the SLP SPI used to verify it in the Authentication Block it transmits. Every host which can verify a digital signature must be configured with keying material and other parameters corresponding with the SLP SPI such that it can perform verifying calculations.

SAs MUST accept multicast service requests and unicast service requests. SAs MAY accept other requests (Attribute and Service Type Requests). SAs MUST listen for multicast DA Advertisements.

The features described up to this point are required to implement. A minimum implementation consists of a User Agent, Service Agent or both.

There are several optional features in the protocol. Note that DAs MUST support all these message types, but DA support is itself

optional to deploy on networks using SLP. UAs and SAs MAY support these message types. These operations are primarily for interactive use (browsing or selectively updating service registrations.) UAs and SAs either support them or not depending on the requirements and constraints of the environment where they will be used.

Service Type Request	A request for all types of service on the network. This allows generic service browsers to be built.
Service Type Reply	A reply to a Service Type Request.
Attribute Request	A request for attributes of a given type of service or attributes of a given service.
Attribute Reply	A reply to an Attribute Request.
Service Deregister	A request to deregister a service or some attributes of a service.
Service Update	A subsequent SrvRqst to an advertisement. This allows individual dynamic attributes to be updated.
SA Advertisement	In the absence of Directory Agents, a User agent may request Service Agents in order to discover their scope configuration. The User Agent may use these scopes in requests.

In the absence of Multicast support, Broadcast MAY be used. The location of DAs may be staticly configured, discovered using SLP as described above, or configured using DHCP. If a message is too large, it may be unicast using TCP.

A SLPv2 implementation SHOULD support SLPv1 [22]. This support includes:

1. SLPv2 DAs are deployed, phasing out SLPv1 DAs.

2. Unscoped SLPv1 requests are considered to be of DEFAULT scope. SLPv1 UAs MUST be reconfigured to have a scope if possible.

3. There is no way for an SLPv2 DA to behave as an unscoped SLPv1 DA. SLPv1 SAs MUST be reconfigured to have a scope if possible.

4. SLPv2 DAs answer SLPv1 requests with SLPv1 replies and SLPv2 requests with SLPv2 replies.

5. SLPv2 DAs use registrations from SLPv1 and SLPv2 in the same way. That is, incoming requests from agents using either version of the protocol will be matched against this common set of registered services.

6. SLPv2 registrations which use Language Tags which are greater than 2 characters long will be inaccessible to SLPv1 UAs.

7. SLPv2 DAs MUST return only service type strings in SrvTypeRply messages which conform to SLPv1 service type string syntax, ie. they MUST NOT return Service Type strings for abstract service types.

8. SLPv1 SrvRqsts and AttrRqsts by Service Type do not match Service URLs with abstract service types. They only match Service URLs with concrete service types.

SLPv1 UAs will not receive replies from SLPv2 SAs and SLPv2 UAs will not receive replies from SLPv1 SAs. In order to interoperate UAs and SAs of different versions require a SLPv2 DA to be present on the network which supports both protocols.

The use of abstract service types in SLPv2 presents a backward compatibility issue for SLPv1. It is possible that a SLPv1 UA will request a service type which is actually an abstract service type. Based on the rules above, the SLPv1 UA will never receive an abstract Service URL reply. For example, the service type 'service:x' in a SLPv1 AttrRqst will not return the attributes of 'service:x:y://orb'. If the request was made with SLPv2, it would return the attributes of this service.

4. URLs used with Service Location

A Service URL indicates the location of a service. This URL may be of the service: scheme [13] (reviewed in section 4.1), or any other URL scheme conforming to the URI standard [8], except that URLs without address specifications SHOULD NOT be advertised by SLP. The service type for an 'generic' URL is its scheme name. For example, the service type string for "http://www.srvloc.org" would be "http".

Reserved characters in URLs follow the rules in RFC 2396 [8].

4.1. Service: URLs

Service URL syntax and semantics are defined in [13]. Any network service may be encoded in a Service URL.

This section provides an introduction to Service URLs and an example showing a simple application of them, representing standard network services.

A Service URL may be of the form:

"service:"<srvtype>"://"<addrspec>

The Service Type of this service: URL is defined to be the string up to (but not including) the final `:' before <addrspec>, the address specification.

<addrspec> is a hostname (which should be used if possible) or dotted decimal notation for a hostname, followed by an optional `:' and port number.

A service: scheme URL may be formed with any standard protocol name by concatenating "service:" and the reserved port [1] name. For example, "service:tftp://myhost" would indicate a tftp service. A tftp service on a nonstandard port could be "service:tftp://bad.glad.org:8080".

Service Types SHOULD be defined by a "Service Template" [13], which provides expected attributes, values and protocol behavior. An abstract service type (also described in [13]) has the form

"service:<abstract-type>:<concrete-type>".

The service type string "service:<abstract-type>" matches all services of that abstract type. If the concrete type is included also, only these services match the request. For example: a SrvRqst or AttrRqst which specifies "service:printer" as the Service Type will match the URL service:printer:lpr://hostname and service:printer:http://hostname. If the requests specified "service:printer:http" they would match only the latter URL.

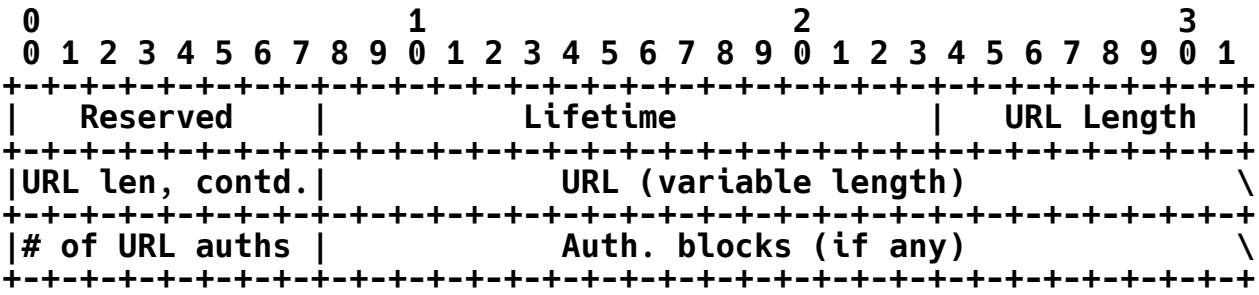
An optional substring MAY follow the last `.' character in the <srvtype> (or <abstract-type> in the case of an abstract service type URL). This substring is the Naming Authority, as described in Section 9.6. Service types with different Naming Authorities are quite distinct. In other words, service:x.one and service:x.two are different service types, as are service:abstract.one:y and service:abstract.two:y.

4.2. Naming Authorities

A Naming Authority MAY optionally be included as part of the Service Type string. 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 typically a string which uniquely identifies an organization. IANA is the implied Naming Authority when no string is appended. "IANA" itself MUST NOT be included explicitly.

Naming Authorities may define Service Types which are experimental, proprietary or for private use. Using a Naming Authority, one may either simply ignore attributes upon registration or create a local-use only set of attributes for one's site. 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 8.1 and 8.3. Service Types SHOULD be registered with IANA to allow for Internet-wide interoperability.

4.3. URL Entries



SLP stores URLs in protocol elements called URL Entries, which associate a length, a lifetime, and possibly authentication information along with the URL. URL Entries, defined as shown above, are used in Service Replies and Service Registrations.

5. Service Attributes

A service advertisement is often accompanied by Service Attributes. These attributes are used by UAs in Service Requests to select appropriate services.

The allowable attributes which may be used are typically specified by a Service Template [13] for a particular service type. Services which are advertised according to a standard template MUST register all service attributes which the standard template requires. URLs with schemes other than "service:" MAY be registered with attributes.

Non-standard attribute names SHOULD begin with "x-", because no standard attribute name will ever have those initial characters.

An attribute list is a string encoding of the attributes of a service. The following ABNF [11] grammar defines attribute lists:

```
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 = "\FF" 1*escape-val
safe-val = ; Any character except reserved.
safe-tag = ; Any character except reserved, star and bad-tag.
reserved = `(' / `)' / `,' / `\' / `!'  / `<' / `=' / `>' / `~' / CTL
escape-val = `\' HEXDIG HEXDIG
bad-tag = CR / LF / HTAB / `_'
 star = `*'
```

The <attr-list>, if present, MUST be scanned prior to evaluation for all occurrences of the escape character `\'. Reserved characters MUST be escaped (other characters MUST NOT be escaped). All escaped characters must be restored to their value before attempting string matching. For Opaque values, escaped characters are not converted - they are interpreted as bytes.

Boolean	Strings which have the form "true" or "false" can only take one value and may only be compared with '='. Booleans are case insensitive when compared.
Integer	Strings which take the form [-] 1*<digit> and fall in the range "-2147483648" to "2147483647" are considered to be Integers. These are compared using integer comparison.
String	All other Strings are matched using strict lexical ordering (see Section 6.4).
Opaque	Opaque values are sequences of bytes. These are distinguished from Strings since they begin with the sequence "\FF". This, unescaped, is an illegal UTF-8 encoding, indicating that what follows is a sequence of bytes expressed in escape notation which constitute the binary value. For example, a '0' byte is encoded "\FF\00".

A string which contains escaped values other than from the reserved set of characters is illegal. If such a string is included in an <attr-list>, <tag-list> or search filter, the SA or DA which receives it MUST return a PARSE_ERROR to the message.

A keyword has only an <attr-tag>, and no values. Attributes can have one or multiple values. All values are expressed as strings.

When values have been advertised by a SA or are registered in a DA, they can take on implicit typing rules for matching incoming requests.

Stored values must be consistent, i.e., x=4,true,sue,\ff\00\00 is disallowed. A DA or SA receiving such an <attr-list> MUST return an INVALID_REGISTRATION error.

6. Required Features

This section defines the minimal implementation requirements for SAs and UAs as well as their interaction with DAs. A DA is not required for SLP to function, but if it is present, the UA and SA MUST interact with it as defined below.

A minimal implementation may consist of either a UA or SA or both. The only required features of a UA are that it can issue SrvRqsts according to the rules below and interpret DAAdverts, SAAdverts and SrvRply messages. The UA MUST issue requests to DAs as they are discovered. An SA MUST reply to appropriate SrvRqsts with SrvRply or SAAdvert messages. The SA MUST also register with DAs as they are discovered.

UAs perform discovery by issuing Service Request messages. SrvRqst messages are issued, using UDP, following these prioritized rules:

1. A UA issues a request to a DA which it has been configured with by DHCP.

2. A UA issues requests to DAs which it has been statically configured with.

3. UA uses multicast/convergence SrvRqsts to discover DAs, then uses that set of DAs. A UA that does not know of any DAs SHOULD retry DA discovery, increasing the waiting interval between subsequent attempts exponentially (doubling the wait interval each time.) The recommended minimum waiting interval is CONFIG_DA_FIND seconds.

4. A UA with no knowledge of DAs sends requests using multicast convergence to SAs. SAs unicast replies to UAs according to the multicast convergence algorithm.

UAs and SAs are configured with a list of scopes to use according to these prioritized rules:

1. With DHCP.

2. With static configuration. The static configuration may be explicitly set to NO SCOPE for UAs, if the User Selectable Scope model is used. See section 11.2.

3. In the absence of configuration, the agent's scope is "DEFAULT".

A UA MUST issue requests with one or more of the scopes it has been configured to use.

A UA which has been statically configured with NO SCOPE LIST will use DA or SA discovery to determine its scope list dynamically. In this case it uses an empty scope list to discover DAs and possibly SAs. Then it uses the scope list it obtains from DAAdverts and possibly SAAdverts in subsequent requests.

The SA MUST register all its services with any DA it discovers, if the DA advertises any of the scopes it has been configured with. A SA obtains information about DAs as a UA does. In addition, the SA MUST listen for multicast unsolicited DAAdverts. The SA registers by sending SrvReg messages to DAs, which reply with SrvReg messages to indicate success. SAs register in ALL the scopes they were configured to use.

6.1. Use of Ports, UDP, and Multicast

DAs MUST accept unicast requests and multicast directory agent discovery service requests (for the service type "service:directory-agent").

SAs MUST accept multicast requests and unicast requests both. The SA can distinguish between them by whether the REQUEST MCAST flag is set in the SLP Message header.

The Service Location Protocol uses multicast for discovering DAs and for issuing requests to SAs by default.

The reserved listening port for SLP is 427. This is the destination port for all SLP messages. SLP messages MAY be transmitted on an ephemeral port. Replies and acknowledgements are sent to the port

from which the request was issued. The default maximum transmission unit for UDP messages is 1400 bytes excluding UDP and other headers.

If a SLP message does not fit into a UDP datagram it MUST be truncated to fit, and the OVERFLOW flag is set in the reply message. A UA which receives a truncated message MAY open a TCP connection (see section 6.2) with the DA or SA and retransmit the request, using the same XID. It MAY also attempt to make use of the truncated reply or reformulate a more restrictive request which will result in a smaller reply.

SLP Requests messages are multicast to The Administratively Scoped SLP Multicast [17] address, which is 239.255.255.253. The default TTL to use for multicast is 255.

In isolated networks, broadcasts will work in place of multicast. To that end, SAs SHOULD and DAs MUST listen for broadcast Service Location messages at port 427. This allows UAs which do not support multicast the use of Service Location on isolated networks.

Setting multicast TTL to less than 255 (the default) limits the range of SLP discovery in a network, and localizes service information in the network.

6.2. Use of TCP

A SrvReg or SrvDeReg may be too large to fit into a datagram. To send such large SLP messages, a TCP (unicast) connection MUST be established.

To avoid the need to implement TCP, one MUST insure that:

- UAs never issue requests larger than the Path MTU. SAs can omit TCP support only if they never have to receive unicast requests longer than the path MTU.
- UAs can accept replies with the 'OVERFLOW' flag set, and make use of the first result included, or reformulate the request.
- Ensure that a SA can send a SrvRply, SrvReg, or SrvDeReg in a single datagram. This means limiting the size of URLs, the number of attributes and the number of authenticators transmitted.

DAs MUST be able to respond to UDP and TCP requests, as well as multicast DA Discovery SrvRqsts. SAs MUST be able to respond to TCP unless the SA will NEVER receive a request or send a reply which will exceed a datagram in size (e.g., some embedded systems).

A TCP connection MAY be used for a single SLP transaction, or for multiple transactions. Since there are length fields in the message headers, SLP Agents can send multiple requests along a connection and read the return stream for acknowledgments and replies.

The initiating agent SHOULD close the TCP connection. The DA SHOULD wait at least CONFIG_CLOSE_CONN seconds before closing an idle connection. DAs and SAs SHOULD close an idle TCP connection after CONFIG_CLOSE_CONN seconds to ensure robust operation, even when the initiating agent neglects to close it. See Section 13 for timing rules.

6.3. Retransmission of SLP messages

Requests which fail to elicit a response are retransmitted. The initial retransmission occurs after a CONFIG_RETRY wait period. Retransmissions MUST be made with exponentially increasing wait intervals (doubling the wait each time). This applies to unicast as well as multicast SLP requests.

Unicast requests to a DA or SA should be retransmitted until either a response (which might be an error) has been obtained, or for CONFIG_RETRY_MAX seconds.

Multicast requests SHOULD be reissued over CONFIG_MC_MAX seconds until a result has been obtained. UAs need only wait till they obtain the first reply which matches their request. That is, retransmission is not required if the requesting agent is prepared to use the 'first reply' instead of 'as many replies as possible within a bounded time interval.'

When SLP SrvRqst, SrvTypeRqst, and AttrRqst messages are multicast, they contain a <PRList> of previous responders. Initially the <PRList> is empty. When these requests are unicast, the <PRList> is always empty.

Any DA or SA which sees its address in the <PRList> MUST NOT respond to the request.

The message SHOULD be retransmitted until the <PRList> causes no further responses to be elicited or the previous responder list and the request will not fit into a single datagram or until CONFIG_MC_MAX seconds elapse.

UAs which retransmit a request use the same XID. This allows a 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. XIDs SHOULD be randomly chosen to avoid

duplicate XIDs in requests if UAs restart frequently.

6.4. Strings in SLP messages

The escape character is a backslash (UTF-8 0x5c) followed by the two hexadecimal digits of the escaped character. Only reserved characters are escaped. For example, a comma (UTF-8 0x29) is escaped as `\29', and a backslash `\' is escaped as `\5c'. String lists used in SLP define the comma to be the delimiter between list elements, so commas in data strings must be escaped in this manner. Backslashes are the escape character so they also must always be escaped when included in a string literally.

String comparison for order and equality in SLP MUST be case insensitive inside the 0x00-0x7F subrange of UTF-8 (which corresponds to ASCII character encoding). Case insensitivity SHOULD be supported throughout the entire UTF-8 encoded Unicode [6] character set.

The case insensitivity rule applies to all string matching in SLPv2, including Scope strings, SLP SPI strings, service types, attribute tags and values in query handling, language tags, previous responder lists. Comparisons of URL strings, however, is case sensitive.

White space (SPACE, CR, LF, TAB) internal to a string value is folded to a single SPACE character for the sake of string comparisons. White space preceding or following a string value is ignored for the purposes of string comparison. For example, " Some String " matches "SOME STRING".

String comparisons (using comparison operators such as `<=' or `>=') are done using lexical ordering in UTF-8 encoded characters, not using any language specific rules.

The reserved 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.

6.4.1. Scope Lists in SLP

Scope Lists in SLPv2 have the following grammar:

```
scope-list = scope-val / scope-val `,' scope-list
scope-val = 1*safe
 safe = ; Any character except reserved.
reserved = `(' / `)' / `,' / `\' / `!'  / `<' / `=' / `>' / `~' / CTL
      / `;' / `*' / `+'
escape-val = `\' HEXDIG HEXDIG
```

Scopes which include any reserved characters must replace the escaped character with the escaped-val format.

7. Errors

If the Error Code in a SLP reply message is nonzero, the rest of the message MAY be truncated. No data is necessarily transmitted or should be expected after the header and the error code, except possibly for some optional extensions to clarify the error, for example as in section D.1.

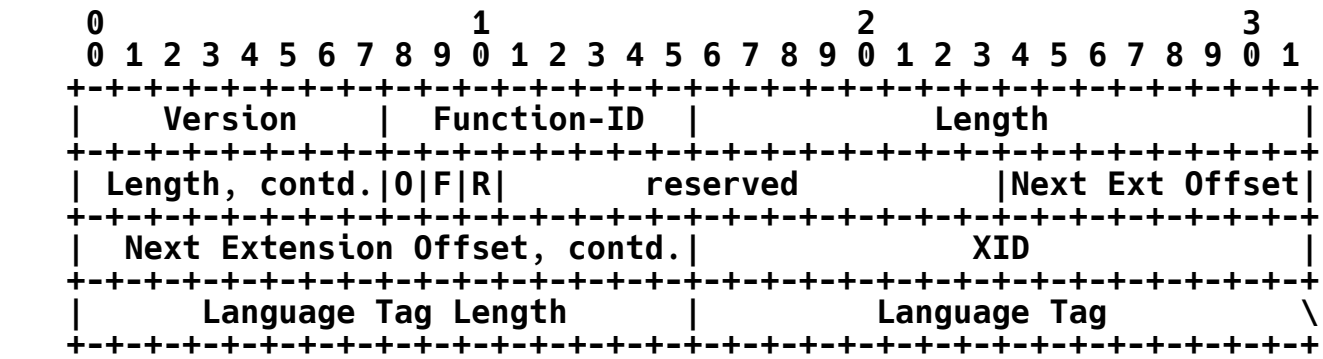
Errors are only returned for unicast requests. Multicast requests are silently discarded if they result in an error.

LANGUAGE_NOT_SUPPORTED = 1: There is data for the service type in the scope in the AttrRqst or SrvRqst, but not in the requested language.
PARSE_ERROR = 2: The message fails to obey SLP syntax.
INVALID_REGISTRATION = 3: The SrvReg has problems -- e.g., a zero lifetime or an omitted Language Tag.
SCOPE_NOT_SUPPORTED = 4: The SLP message did not include a scope in its <scope-list> supported by the SA or DA.
AUTHENTICATION_UNKNOWN = 5: The DA or SA receives a request for an unsupported SLP SPI.
AUTHENTICATION_ABSENT = 6: The DA expected URL and ATTR authentication in the SrvReg and did not receive it.
AUTHENTICATION_FAILED = 7: The DA detected an authentication error in an Authentication block.
VER_NOT_SUPPORTED = 9: Unsupported version number in message header.
INTERNAL_ERROR = 10: The DA (or SA) is too sick to respond.
DA_BUSY_NOW = 11: UA or SA SHOULD retry, using exponential back off.
OPTION_NOT_UNDERSTOOD = 12: The DA (or SA) received an unknown option from the mandatory range (see section 9.1).
INVALID_UPDATE = 13: The DA received a SrvReg without FRESH set, for an unregistered service or with inconsistent Service Types.
MSG_NOT_SUPPORTED = 14: The SA received an AttrRqst or SrvTypeRqst and does not support it.
REFRESH_REJECTED = 15: The SA sent a SrvReg or partial SrvDereg to a DA more frequently than the DA's min-refresh-interval.

8. Required SLP Messages

All length fields in SLP messages are in network byte order. Where ' tuples' are defined, these are sequences of bytes, in the precise order listed, in network byte order.

SLP messages all begin with the following header:



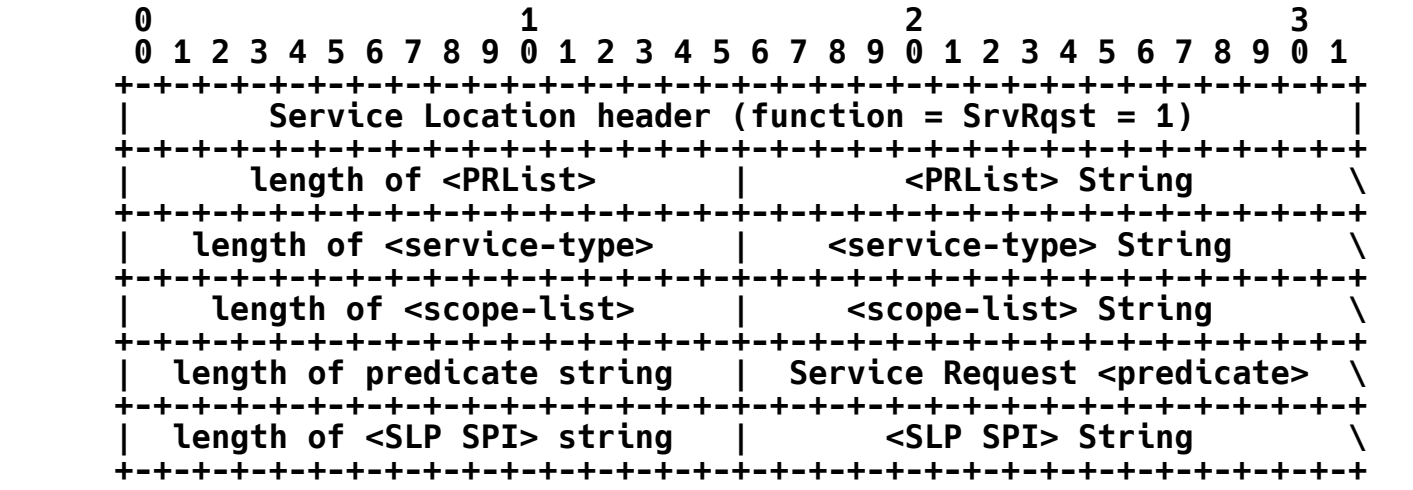
Message Type	Abbreviation	Function-ID
Service Request	SrvRqst	1
Service Reply	SrvRply	2
Service Registration	SrvReg	3
Service Deregister	SrvDeReg	4
Service Acknowledge	SrvAck	5
Attribute Request	AttrRqst	6
Attribute Reply	AttrRply	7
DA Advertisement	DAAdvert	8
Service Type Request	SrvTypeRqst	9
Service Type Reply	SrvTypeRply	10
SA Advertisement	SAAdvert	11

SAs and UAs MUST support SrvRqst, SrvRply and DAAdvert. SAs MUST also support SrvReg, SAAdvert and SrvAck. For UAs and SAs, support for other messages are OPTIONAL.

- Length is the length of the entire SLP message, header included.
- The flags are: OVERFLOW (0x80) is set when a message's length exceeds what can fit into a datagram. FRESH (0x40) is set on every new SrvReg. REQUEST MCAST (0x20) is set when multicasting or broadcasting requests. Reserved bits MUST be 0.
- Next Extension Offset is set to 0 unless extensions are used. The first extension begins at 'offset' bytes, from the message's beginning. It is placed after the SLP message data. See Section 9.1 for how to interpret unrecognized SLP Extensions.
- XID is set to a unique value for each unique request. If the request is retransmitted, the same XID is used. Replies set the XID to the same value as the xid in the request. Only unsolicited DAAdverts are sent with an XID of 0.
- Lang Tag Length is the length in bytes of the Language Tag field.
- Language Tag conforms to [7]. The Language Tag in a reply MUST be the same as the Language Tag in the request. This field must be encoded 1*8ALPHA *("-" 1*8ALPHA).

If an option is specified, and not included in the message, the receiver MUST respond with a PARSE_ERROR.

8.1. Service Request



In order for a Service to match a SrvRqst, it must belong to at least one requested scope, support the requested service type, and match the predicate. If the predicate is present, the language of the request (ignoring the dialect part of the Language Tag) must match the advertised service.

<PRList> is the Previous Responder List. This <string-list> contains dotted decimal notation IP (v4) addresses, and is iteratively multicast to obtain all possible results (see Section 6.3). UAs SHOULD implement this discovery algorithm. SAs MUST use this to discover all available DAs in their scope, if they are not already configured with DA addresses by some other means.

A SA silently drops all requests which include the SA's address in the <PRList>. An SA which has multiple network interfaces MUST check if any of the entries in the <PRList> equal any of its interfaces. An entry in the PRList which does not conform to an IPv4 dotted decimal address is ignored: The rest of the <PRList> is processed normally and an error is not returned.

Once a <PRList> plus the request exceeds the path MTU, multicast convergence stops. This algorithm is not intended to find all instances; it finds 'enough' to provide useful results.

The <scope-list> is a <string-list> of configured scope names. SAs and DAs which have been configured with any of the scopes in this list will respond. DAs and SAs MUST reply to unicast requests with a

SCOPE_NOT_SUPPORTED error if the <scope-list> is omitted or fails to include a scope they support (see Section 11). The only exceptions to this are described in Section 11.2.

The <service-type> string is discussed in Section 4. Normally, a SrvRqst elicits a SrvRply. There are two exceptions: If the <service-type> is set to "service:directory-agent", DAs respond to the SrvRqst with a DAAdvert (see Section 8.5.) If set to "service:service-agent", SAs respond with a SAAdvert (see Section 8.6.) If this field is omitted, a PARSE_ERROR is returned - as this field is REQUIRED.

The <predicate> is a LDAPv3 search filter [14]. This field is OPTIONAL. Services may be discovered simply by type and scope. Otherwise, services are discovered which satisfy the <predicate>. If present, it is compared to each registered service. If the attribute in the filter has been registered with multiple values, the filter is compared to each value and the results are ORed together, i.e., "(x=3)" matches a registration of (x=1,2,3); "(!(Y=0))" matches (y=0,1) since Y can be nonzero. Note the matching is case insensitive. Keywords (i.e., attributes without values) are matched with a "presence" filter, as in "(keyword=*)".

An incoming request term MUST have the same type as the attribute in a registration in order to match. Thus, "(x=33)" will not match ' x=true', etc. while "(y=foo)" will match 'y=FOO'. "(|(x=33)(y=foo))" will be satisfied, even though "(x=33)" cannot be satisfied, because of the `|' (boolean disjunction).

Wildcard matching MUST be done with the '=' filter. In any other case, a PARSE_ERROR is returned. Request terms which include wildcards are interpreted to be Strings. That is, (x=34*) would match 'x=34foo', but not 'x=3432' since the first value is a String while the second value is an Integer; Strings don't match Integers.

Examples of Predicates follow. <t> indicates the service type of the SrvRqst, <s> gives the <scope-list> and <p> is the predicate string.

<t>=service:http <s>=DEFAULT <p>= (empty string)
This is a minimal request string. It matches all http services advertised with the default scope.

<t>=service:pop3 <s>=SALES,DEFAULT <p>=(user=wump)
This is a request for all pop3 services available in the SALES or DEFAULT scope which serve mail to the user `wump'.

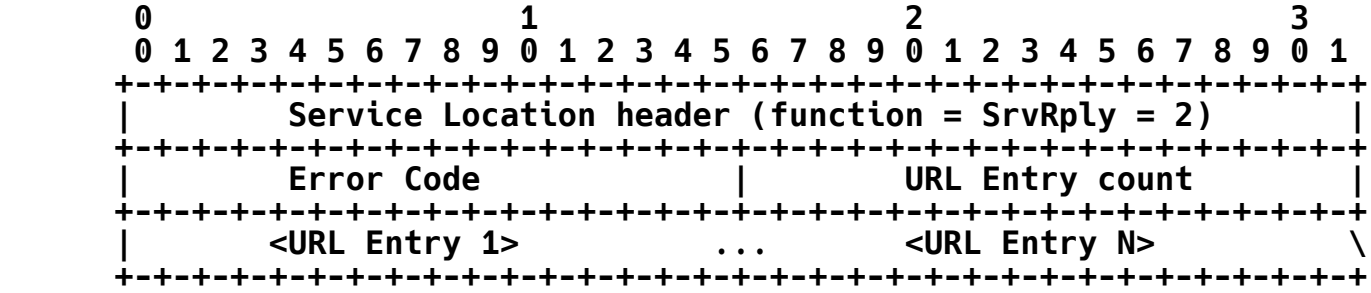
<t>=service:backup <s>=BLDG 32 <p>=(&(q<=3)(speed>=1000))
This returns the backup service which has a queue length less than 3 and a speed greater than 1000. It will return this only for services registered with the BLDG 32 scope.

<t>=service:directory-agent <s>=DEFAULT <p>=
This returns DAAdverts for all DAs in the DEFAULT scope.

DAs are discovered by sending a SrvRqst with the service type set to "service:directory-agent". If a predicate is included in the SrvRqst, the DA SHOULD respond only if the predicate can be satisfied with the DA's attributes. The <scope-list> MUST contain all scopes configured for the UA or SA which is discovering DAs.

The <SLP SPI> string indicates a SLP SPI that the requester has been configured with. If this string is omitted, the responder does not include any Authentication Blocks in its reply. If it is included, the responder MUST return a reply which has an associated authentication block with the SLP SPI in the SrvRqst. If no replies may be returned because the SLP SPI is not supported, the responder returns an AUTHENTICATION_UNKNOWN error.

8.2. Service Reply



The service reply contains zero or more URL entries (see Section 4.3). A service reply with zero URL entries MUST be returned in response to a unicast Service Request, if no matching URLs are present. A service reply with zero URL entries MUST NOT be sent in response to a multicast or broadcast service request (instead, if there was no match found or an error processing the request, the service reply should not be generated at all).

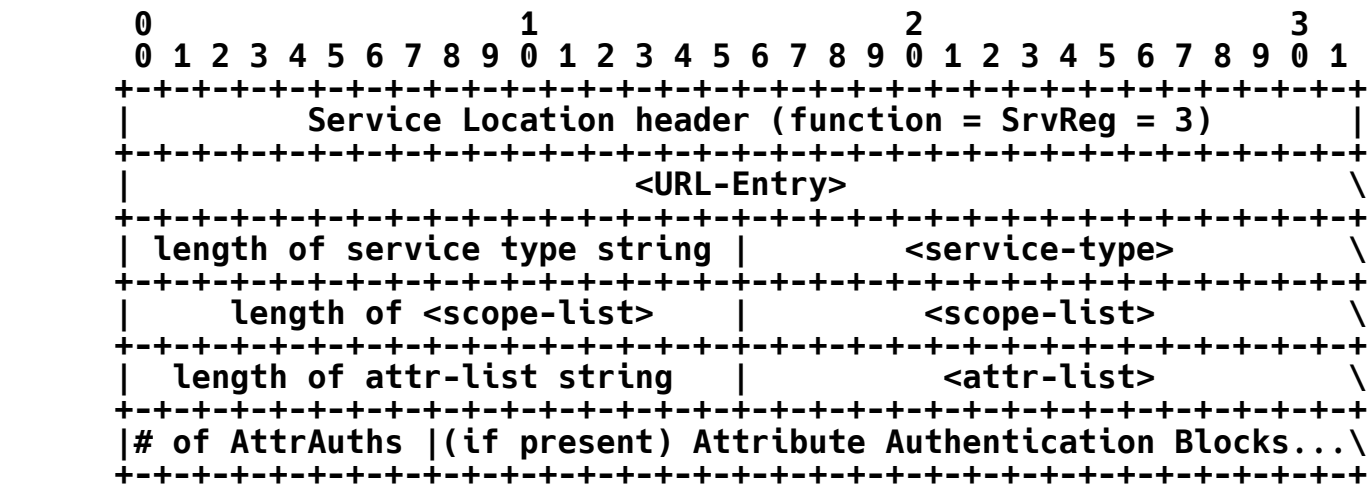
If the reply overflows, the UA MAY simply use the first URL Entry in the list. A URL obtained by SLP may not be cached longer than Lifetime seconds, unless there is a URL Authenticator block present.

In that case, the cache lifetime is indicated by the Timestamp in the URL Authenticator (see Section 9.2).

An authentication block is returned in the URL Entries, including the SLP SPI in the SrvRqst. If no SLP SPI was included in the request, no Authentication Blocks are returned in the reply. URL Authentication Blocks are defined in Section 9.2.1.

If a SrvRply is sent by UDP, a URL Entry MUST NOT be included unless it fits entirely without truncation.

8.3. Service Registration



The <entry> is a URL Entry (see section 4.3). The Lifetime defines how long a DA can cache the registration. SAs SHOULD reregister before this lifetime expires (but SHOULD NOT more often than once per second). The Lifetime MAY be set to any value between 0 and 0xffff (maximum, around 18 hours). Long-lived registrations remain stale longer if the service fails and the SA does not deregister the service.

The <service-type> defines the service type of the URL to be registered, regardless of the scheme of the URL. The <scope-list> MUST contain the names of all scopes configured for the SA, which the DA it is registering with supports. The default value for the <scope-list> is "DEFAULT" (see Section 11).

The SA MUST register consistently with all DAs. If a SA is configured with scopes X and Y and there are three DAs, whose scopes are "X", "Y" and "X,Y" respectively, the SA will register the with all three DAs in their respective scopes. All future updates and deregistrations of the service must be sent to the same set of DAs in

the same scopes the service was initially registered in.

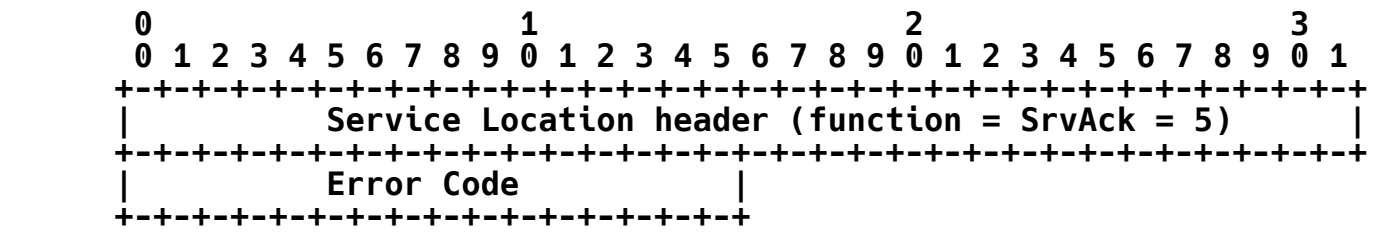
The <attr-list>, if present, specifies the attributes and values to be associated with the URL by the DA (see Section 5).

A SA configured with the ability to sign service registrations MUST sign each of the URLs and Attribute Lists using each of the keys it is configured to use, and the DA it is registering with accepts. (The SA MUST acquire DAAdverts for all DAs it will register with to obtain the DA's SLP SPI list and attributes, as described in Section 8.5). The SA supplies a SLP SPI in each authentication block indicating the SLP SPI configuration required to verify the digital signature. The format of the digital signatures used is defined in section 9.2.1.

Subsequent registrations of previously registered services MUST contain the same list of SLP SPIs as previous ones or else DAs will reject them, replying with an AUTHENTICATION_ABSENT error.

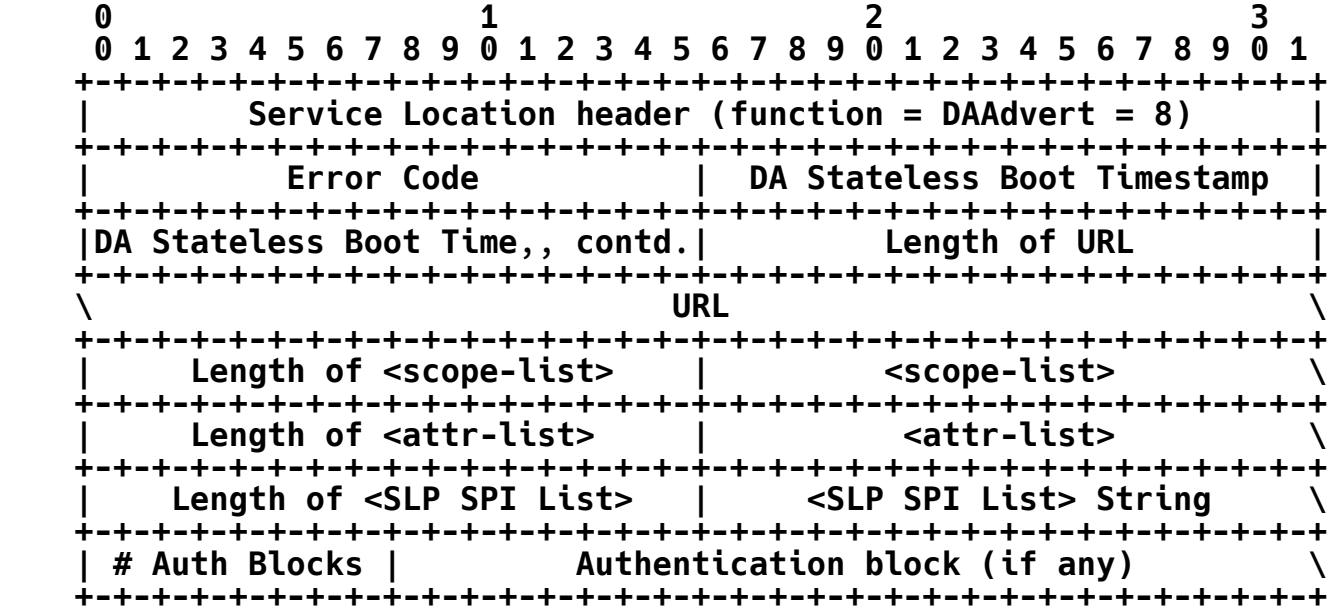
A registration with the FRESH flag set will replace *entirely* any previous registration for the same URL in the same language. If the FRESH flag is not set, the registration is an "incremental" registration (see Section 9.3).

8.4. Service Acknowledgment



A DA returns a SrvAck to an SA after a SrvReg. It carries only a two byte Error Code (see Section 7).

8.5. Directory Agent Advertisement



The Error Code is set to 0 when the DAAdvert is multicast. If the DAAdvert is being returned due to a unicast SrvRqst (ie. a request without the REQUEST MCAST flag set) the DA returns the same errors a SrvRply would.

The <scope-list> of the SrvRqst must either be omitted or include a scope which the DA supports. The DA Stateless Boot Timestamp indicates the state of the DA (see section 12.1).

The DA MAY include a list of its attributes in the DAAdvert. This list SHOULD be kept short, as the DAAdvert must fit into a datagram in order to be multicast.

A potential scaling problem occurs in SLPv2 if SAs choose too low a Lifetime. In this case, an onerous amount of reregistration occurs as more services are deployed. SLPv2 allows DAs to control SAs frequency of registration. A DA MAY reissue a DAAdvert with a new set of attributes at any time, to change the reregistration behavior of SAs. These apply only to subsequent registrations; existing service registrations with the DA retain their registered lifetimes.

If the DAAdvert includes the attribute "min-refresh-interval" it MUST be set to a single Integer value indicating a number of seconds. If this attribute is present SAs MUST NOT refresh any particular service advertisement more frequently than this value. If SrvReg with the FRESH FLAG not set or SrvDereg with a non-empty tag list updating a

particular service are received more often than the value for the DA's advertised "min-refresh-interval" attribute the DA SHOULD reject the message and return a REFRESH_REJECTED error in the SrvAck.

The URL is "service:directory-agent://"<addr> of the DA, where <addr> is the dotted decimal numeric address of the DA. The <scope-list> of the DA MUST NOT be NULL.

The SLP SPI List is the list of SPIs that the DA is capable of verifying. SAs MUST NOT register services with authentication blocks for those SLP SPIs which are not on the list. DAs will reject service registrations which they cannot verify, returning an AUTHENTICATION_UNKNOWN error.

The format of DAAdvert signatures is defined in Section 9.2.1.

The SLP SPI which is used to verify the DAAdvert is included in the Authentication Block. When DAAdverts are multicast, they may have to transmit multiple DAAdvert Authentication Blocks. If the DA is configured to be able to generate signatures for more than one SPI, the DA MUST include one Authentication Block for each SPI. If all these Authentication Blocks do not fit in a single datagram (to multicast or broadcast) the DA MUST send separate DAAdverts so that Authentication Blocks for all the SPIs the DA is capable of generating are sent.

If the DAAdvert is being sent in response to a SrvRqst, the DAAdvert contains only the authentication block with the SLP SPI in the SrvRqst, if the DA is configured to be able to produce digital signatures using that SLP SPI. If the SrvRqst is unicast to the DA (the REQUEST MCAST flag in the header is not set) and an unsupported SLP SPI is included, the DA replies with a DAAdvert with the Error Code set to an AUTHENTICATION_UNKNOWN error.

UAs SHOULD be configured with SLP SPIs that will allow them to verify DA Advertisements. If the UA is configured with SLP SPIs and receives a DAAdvert which fails to be verified using one of them, the UA MUST discard it.

8.6. Service Agent Advertisement

User Agents MUST NOT solicit SA Advertisements if they have been configured to use a particular DA, if they have been configured with a <scope-list> or if DAs have been discovered. UAs solicit SA Advertisements only when they are explicitly configured to use User Selectable scopes (see Section 11.2) in order to discover the scopes that SAs support. This allows UAs without scope configuration to make use of either DAs or SAs without any functional difference

except performance.

A SA MAY be configured with attributes, and SHOULD support the attribute 'service-type' whose value is all the service types of services represented by the SA. SAs MUST NOT respond if the SrvRqst predicate is not satisfied. For example, only SAs offering 'nfs' services SHOULD respond with a SAAdvert to a SrvRqst for service type "service:service-agent" which includes a predicate "(service-type=nfs)".

The SA responds only to multicast SA discovery requests which either include no <scope-list> or a scope which they are configured to use.

The SAAdvert MAY include a list of attributes the SA supports. This attribute list SHOULD be kept short so that the SAAdvert will not exceed the path MTU in size.

The URL is "service:service-agent://"<addr> of the SA, where <addr> is the dotted decimal numeric address of the SA. The <scope-list> of the SA MUST NOT be null.

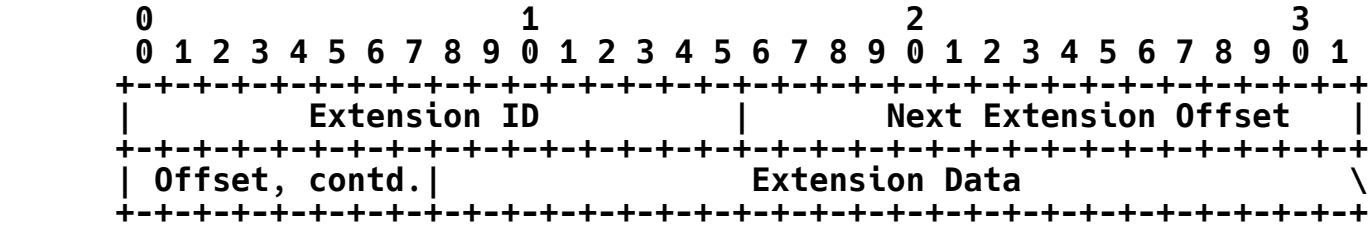
The SAAdvert contains one SAAdvert Authentication block for each SLP SPI the SA can produce Authentication Blocks for. If the UA can verify the Authentication Block of the SAAdvert, and the SAAdvert fails to be verified, the UA MUST discard it.

9. Optional Features

The features described in this section are not mandatory. Some are useful for interactive use of SLP (where a user rather than a program will select services, using a browsing interface for example) and for scalability of SLP to larger networks.

9.1. Service Location Protocol Extensions

The format of a Service Location Extension is:



Extension IDs are assigned in the following way:

0x0000-0x3FFF Standardized. Optional to implement. Ignore if unrecognized.
0x4000-0x7FFF Standardized. Mandatory to implement. A UA or SA which receives this option in a reply and does not understand it MUST silently discard the reply. A DA or SA which receives this option in a request and does not understand it MUST return an OPTION_NOT_UNDERSTOOD error.
0x8000-0x8FFF For private use (not standardized). Optional to implement. Ignore if unrecognized.
0x9000-0xFFFF Reserved.

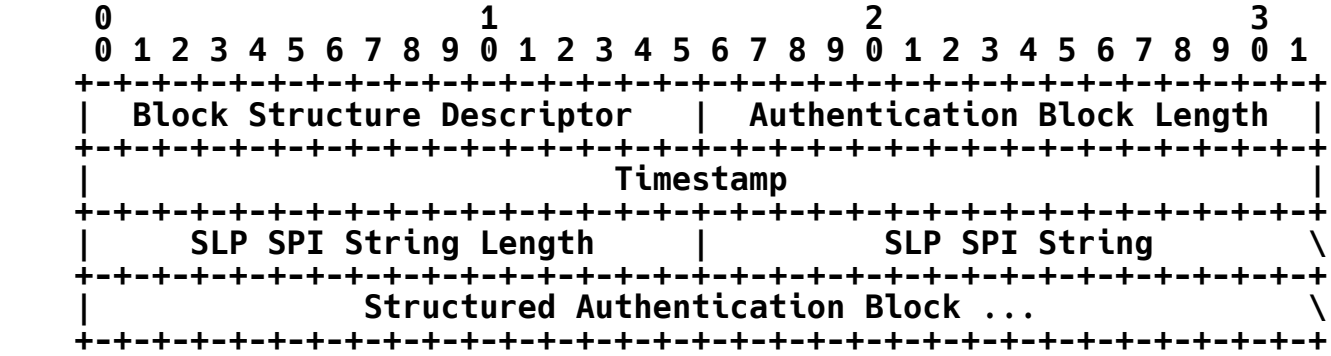
The three byte offset to next extension indicates the position of the next extension as offset from the beginning of the SLP message.

The offset value is 0 if there are no extensions following the current extension.

If the offset is 0, the length of the current Extension Data is determined by subtracting total length of the SLP message as given in the SLP message header minus the offset of the current extension.

Extensions defined in this document are in Section D. See section 15 for procedures that are required when specifying new SLP extensions.

9.2. Authentication Blocks



Authentication blocks are returned with certain SLP messages to verify that the contents have not been modified, and have been transmitted by an authorized agent. The authentication data (contained in the Structured Authentication Block) is typically case sensitive. Even though SLP registration data (e.g., attribute values) are typically are not case sensitive, the case of the registration data has to be preserved by the registering DA so that UAs will be able to verify the data used for calculating digital signature data.

The Block Structure Descriptor (BSD) identifies the format of the Authenticator which follows. BSDs 0x0000-0x7FFF will be maintained by IANA. BSDs 0x8000-0x8FFF are for private use.

The Authentication Block Length is the length of the entire block, starting with the BSD.

The Timestamp is the time that the authenticator expires (to prevent replay attacks.) The Timestamp is a 32-bit unsigned fixed-point number of seconds relative to 0h on 1 January 1970. SAs use this value to indicate when the validity of the digital signature expires. This Timestamp will wrap back to 0 in the year 2106. Once the value of the Timestamp wraps, the time at which the Timestamp is relative to resets. For example, after 06h28 and 16 seconds 5 February 2106, all Timestamp values will be relative to that epoch date.

The SLP Security Parameters Index (SPI) string identifies the key length, algorithm parameters and keying material to be used by agents to verify the signature data in the Structured Authentication Block. The SLP SPI string has the same grammar as the <scope-val> defined in Section 6.4.1.

Reserved characters in SLP SPI strings must be escaped using the same convention as used throughout SLPv2.

SLP SPIs deployed in a site MUST be unique. An SLP SPI used for BSD=0x0002 must not be the same as used for some other BSD.

All SLP agents MUST implement DSA [20] (BSD=0x0002). SAs MUST register services with DSA authentication blocks, and they MAY register them with other authentication blocks using other algorithms. SAs MUST use DSA authentication blocks in SrvDeReg messages and DAs MUST use DSA authentication blocks in unsolicited DAAdverts.

9.2.1. SLP Message Authentication Rules

The sections below define how to calculate the value to apply to the algorithm identified by the BSD value. The components listed are used as if they were a contiguous single byte aligned buffer in the order given.

URL
    16-bit Length of SLP SPI String, SLP SPI String.
    16-bit Length of URL, URL,
    32-bit Timestamp.

Attribute List
    16-bit Length of SLP SPI String, SLP SPI String,
    16-bit length of <attr-list>, <attr-list>,
    32-bit Timestamp.

DAAdvert
    16-bit Length of SLP SPI String, SLP SPI String,
    32-bit DA Stateless Boot Timestamp,
    16-bit Length of URL, URL,
    16-bit Length of <attr-list>, <attr-list>,
    16-bit Length of DA's <scope-list>, DA's <scope-list>,
    16-bit Length of DA's <SLP SPI List>, DA's <SLP SPI List>,
    32-bit Timestamp.

    The first SLP SPI is the SLP SPI in the Authentication Block. This SLP SPI indicates the keying material and other parameters to use to verify the DAAdvert. The SLP SPI List is the list of SLP SPIs the DA itself supports, and is able to verify.

SAAdvert
    16-bit Length of SLP SPI String, SLP SPI String,
    16-bit Length of URL, URL,
    16-bit Length of <attr-list>, <attr-list>,
    16-bit Length of <scope-list>, <scope-list>,
    32-bit Timestamp.

9.2.2 DSA with SHA-1 in Authentication Blocks

BSD=0x0002 is defined to be DSA with SHA-1. The signature calculation is defined by [20]. The signature format conforms to that in the X.509 v3 certificate:

1. The signature algorithm identifier (an OID)
2. The signature value (an octet string)
3. The certificate path.

All data is represented in ASN.1 encoding:

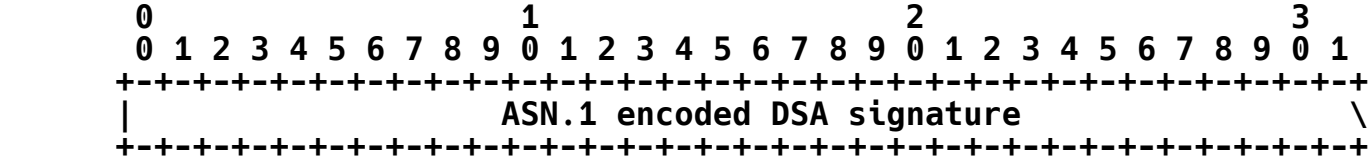
```
id-dsa-with-sha1 ID  ::=  {
                 iso(1) member-body(2) us(840) x9-57 (10040)
                 x9cm(4) 3 }
```

i.e., the ASN.1 encoding of 1.2.840.10040.4.3 followed immediately by:

```
Dss-Sig-Value  ::=  SEQUENCE  {
                 r       INTEGER,
                 s       INTEGER  }
```

i.e., the binary ASN.1 encoding of r and s computed using DSA and SHA-1. This is followed by a certificate path, as defined by X.509 [10], [2], [3], [4], [5].

Authentication Blocks for BSD=0x0002 have the following format. In the future, BSDs may be assigned which have different formats.



9.3. Incremental Service Registration

Incremental registrations update attribute values for a previously registered service. Incremental service registrations are useful when only a single attribute has changed, for instance. In an incremental registration, the FRESH flag in the SrvReg header is NOT set.

The new registration's attributes replace the previous registration's, but do not affect 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 an incremental 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.

Incremental registrations MUST NOT be performed for services registered with Authentication Blocks. These must be registered with ALL attributes, with the FRESH flag in the SrvReg header set. DAs which receive such registration messages return an AUTHENTICATION_FAILED error.

If the FRESH flag is not set and the DA does not have a prior registration for the service, the incremental registration fails with error code INVALID_UPDATE.

The SA MUST use the same <scope-list> in an update message as was used in the prior registration. If this is not done, the DA returns a SCOPE_NOT_SUPPORTED error. In order to change the scope of a service advertisement it MUST be deregistered first and reregistered with a new <scope-list>.

The SA MUST use the same <service-type> in an update message as was used in a prior registration of the same URL. If this is not done, the DA returns an INVALID_UPDATE error.

9.4. Tag Lists

Tag lists are used in SrvDeReg and AttrReq messages. The syntax of a <tag-list> item is:

```
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
      grammar in Section 5).
```

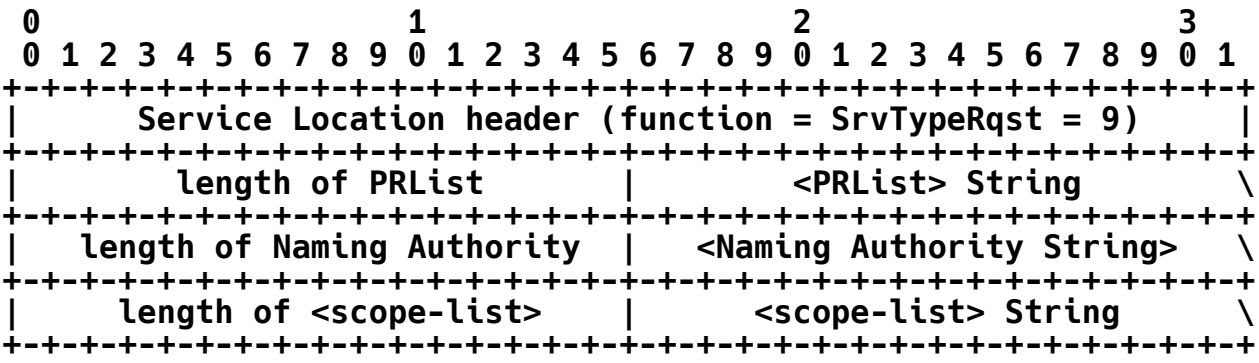
Wild card characters in a <tag-list> item match arbitrary sequences of characters. For instance "*bob*" matches "some bob I know", "bigbob", "bobby" and "bob".

10. Optional SLP Messages

The additional requests provide features for user interaction and for efficient updating of service advertisements with dynamic attributes.

10.1. Service Type Request

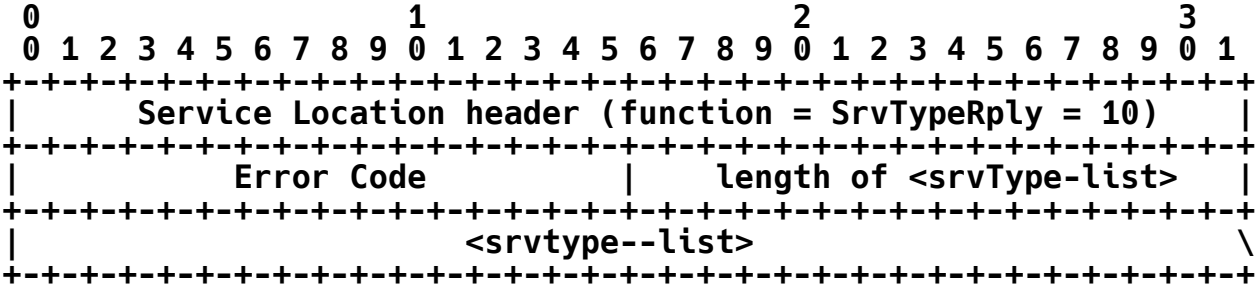
The Service Type Request (SrvTypeRqst) allows a UA to discover all types of service on a network. This is useful for general purpose service browsers.



The <PRList> list and <scope-list> are interpreted as in Section 8.1.

The Naming Authority string, if present in the request, will limit the reply to Service Type strings with the specified Naming Authority. If the Naming Authority string is absent, the IANA registered service types will be returned. If the length of the Naming Authority is set to 0xFFFF, the Naming Authority string is omitted and ALL Service Types are returned, regardless of Naming Authority.

10.2. Service Type Reply

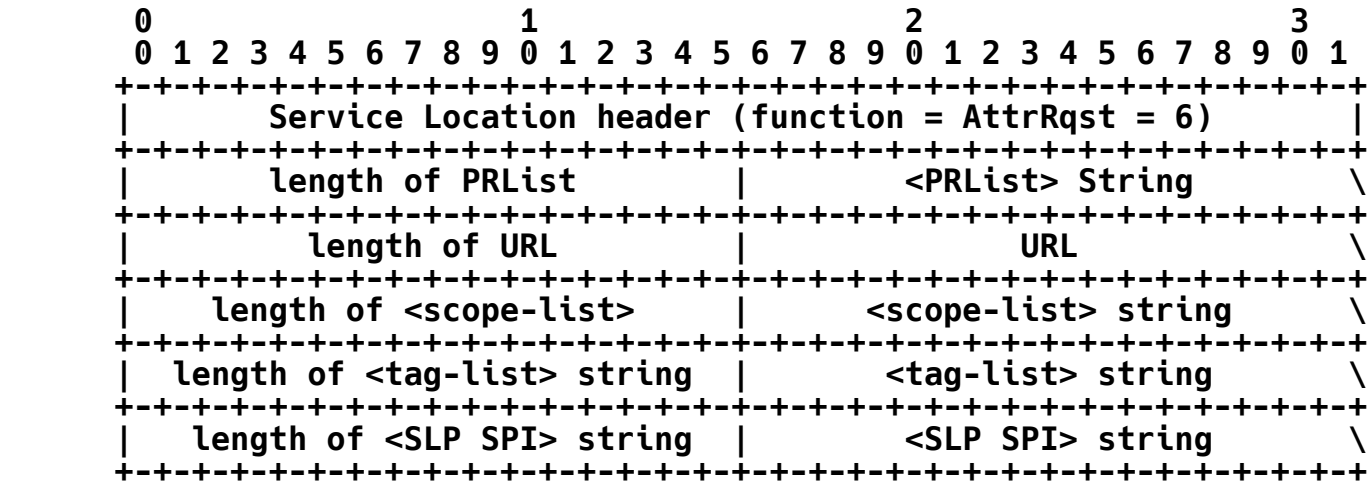


The service-type Strings (as described in Section 4.1) are provided in <srvtype-list>, which is a <string-list>.

If a service type has a Naming Authority other than IANA it MUST be returned following the service type string and a `.' character. Service types with the IANA Naming Authority do not include a Naming Authority string.

10.3. Attribute Request

The Attribute Request (AttrRqst) allows a UA to discover attributes of a given service (by supplying its URL) or for an entire service type. The latter feature allows the UA to construct a query for an available service by selecting desired features. The UA may request that all attributes are returned, or only a subset of them.



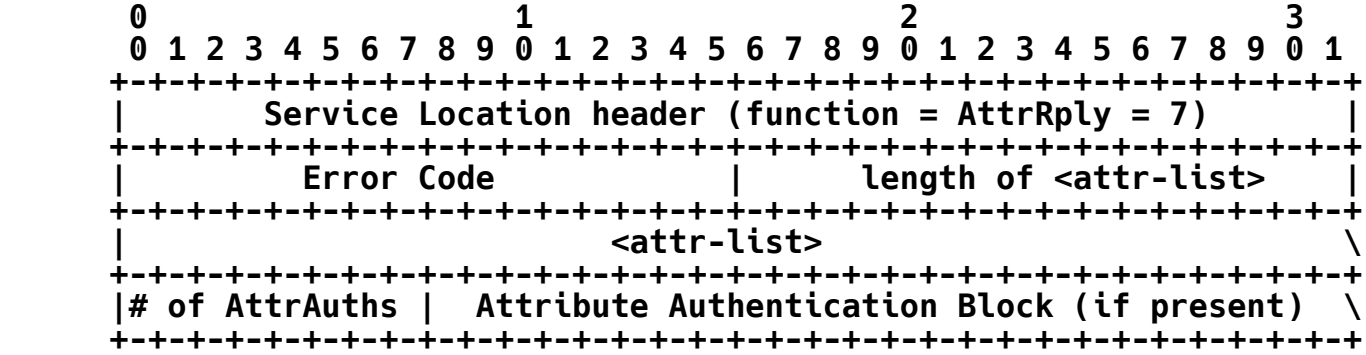
The <PRList>, <scope-list> and <SLP SPI> string are interpreted as in Section 8.1.

The URL field can take two forms. It can simply be a Service Type (see Section 4.1), such as "http" or "service:tftp". In this case, 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 alternatively be a full URL, such as "service:printer:lpr://igore.wco.ftp.com:515/draft" or "nfs://max.net/znoo". In this, only the registered attributes for the specified URL are returned.

The <tag-list> field is a <string-list> of attribute tags, as defined in Section 9.4 which indicates the attributes to return in the AttrRply. If <tag-list> is omitted, all attributes are returned. <tag-list> MUST be omitted and a full URL MUST be included when attributes when a SLP SPI List string is included, otherwise the DA will reply with an AUTHENTICATION_FAILED error.

10.4. Attribute Reply



The format of the <attr-list> and the Authentication Block is as specified for SrvReg (see Section 9.2.1).

Attribute replies SHOULD be returned with the original case of the string registration intact, as they are likely to be human readable. In the case where the AttrRqst was by service type, all attributes defined for the service type, and all their values are returned.

Although white space is folded for string matching, attribute tags and values MUST be returned with their original white space preserved.

Only one copy of each attribute tag or String value should be returned, arbitrarily choosing one version (with respect to upper and lower case and white space internal to the strings): Duplicate attributes and values SHOULD be removed. An arbitrary version of the string value and tag name is chosen for the merge. For example: "(A=a a,b)" merged with "(a=A A,B)" may yield "(a=a a,B)".

10.5. Attribute Request/Reply Examples

Suppose that printer services have been registered as follows:

```
Registered Service:
  URL        = service:printer:lpr://igore.wco.ftp.com/draft
  scope-list = Development
  Lang. Tag  = en
  Attributes = (Name=Igore),(Description=For developers only),
               (Protocol=LPR),(location-description=12th floor),
               (Operator=James Dornan \3cdornan@monster\3e),
               (media-size=na-letter),(resolution=res-600),x-OK

  URL        = service:printer:lpr://igore.wco.ftp.com/draft
  scope-list = Development
```

```
  Lang. Tag  = de
  Attributes = (Name=Igore),(Description=Nur fuer Entwickler),
               (Protocol=LPR),(location-description=13te Etage),
               (Operator=James Dornan \3cdornan@monster\3e),
               (media-size=na-letter),(resolution=res-600),x-OK

  URL        = service:printer:http://not.wco.ftp.com/cgi-bin/pub-prn
  scope-list = Development
  Lang. Tag  = en
  Attributes = (Name=Not),(Description=Experimental IPP printer),
               (Protocol=http),(location-description=QA bench),
               (media-size=na-letter),(resolution=other),x-BUSY
```

Notice the first printer, "Igore" is registered in both English and German. The `<' and `>' characters in the Operator attribute value which are part of the Email address had to be escaped, as they are reserved characters for values.

Attribute tags are not translated, though attribute values may be, see [13].

The attribute Request:

```
  URL        = service:printer:lpr://igore.wco.ftp.com/draft
  scope-list = Development
  Lang. Tag  = de
  tag-list   = resolution,loc*
```

receives the Attribute Reply:

```
  (location-description=13te Etage),(resolution=res-600)
```

The attribute Request:

```
  URL        = service:printer
  scope-list = Development
  Lang. Tag  = en
  tag-list   = x-*,resolution,protocol
```

receives an Attribute Reply containing:

```
  (protocols=http,LPR),(resolution=res-600,other),x-OK,x-BUSY
```

The first request is by service instance and returns the requested values, in German. The second request is by abstract service type (see Section 4) and returns values from both "Igore" and "Not".

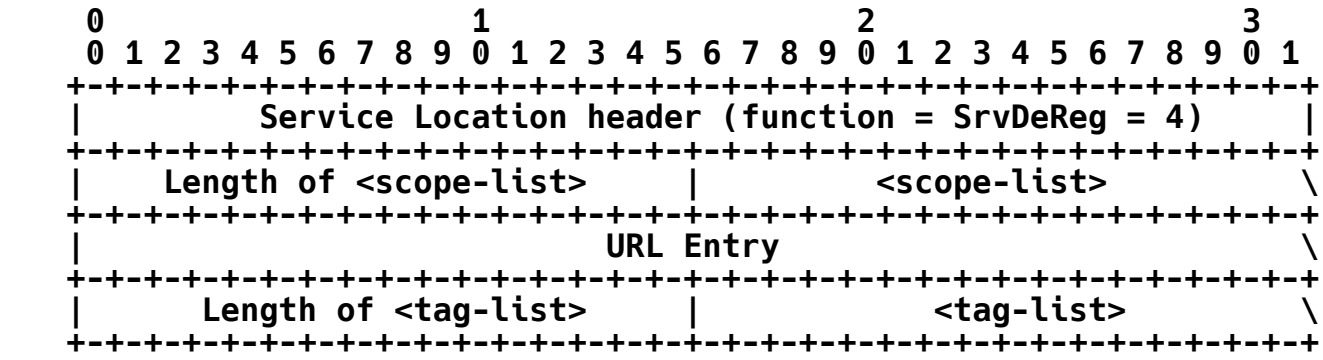
An attribute Authentication Block is returned if an authentication block with the SLP SPI in the AttrRqst can be returned. Note that the <attr-list> returned from a DA with an Authentication Block MUST be identical to the <attr-list> registered by a SA, in order for the authentication verification calculations to be possible.

A SA or DA only returns an Attribute Authentication Block if the AttrRqst included a full URL in the request and no tag list.

If an SLP SPI is specified in a unicast request (the REQUEST MCAST flag in the header is not set) and the SA or DA cannot return an Authentication Block with that SLP SPI, an AUTHENTICATION_UNKNOWN error is returned. The # of Attr Auths field is set to 0 if there no Authentication Block is included, or 1 if an Authentication Block follows.

10.6. Service Deregistration

A DA deletes a service registration when its Lifetime expires. Services SHOULD be deregistered when they are no longer available, rather than leaving the registrations to time out.



The <scope-list> is a <string-list> (see section 2.1).

The SA MUST retry if there is no response from the DA, see Section 12.3. The DA acknowledges a SrvDeReg with a SrvAck. Once the SA receives an acknowledgment indicating success, the service and/or attributes are no longer advertised by the DA. The DA deregisters the service or service attributes from every scope specified in the SrvDeReg which it was previously registered in.

The SA MUST deregister all services with the same scope list used to register the service with a DA. If this is not done in the SrvDeReg message, the DA returns a SCOPE_NOT_SUPPORTED error. The Lifetime field in the URL Entry is ignored for the purposes of the SrvDeReg.

The <tag-list> is a <string-list> of attribute tags to deregister as defined in Section 9.4. If no <tag-list> is present, the SrvDeReg deregisters the service in all languages it has been registered in. If the <tag-list> is present, the SrvDeReg deregisters the attributes whose tags are listed in the tag spec. Services registered with Authentication Blocks MUST NOT include a <tag-list> in a SrvDeReg message: A DA will respond with an AUTHENTICATION_FAILED error in this case.

If the service to be deregistered was registered with an authentication block or blocks, a URL authentication block for each of the SLP SPIs registered must be included in the SrvDeReg. Otherwise, the DA returns an AUTHENTICATION_ABSENT error. If the message fails to be verified by the DA, an AUTHENTICATION_FAILED error is returned by the DA.

11. Scopes

Scopes are sets of services. The primary use of Scopes is to provide the ability to create administrative groupings of services. A set of services may be assigned a scope by network administrators. A client seeking services is configured to use one or more scopes. The user will only discover those services which have been configured for him or her to use. By configuring UAs and SAs with scopes, administrators may provision services. Scopes strings are case insensitive. The default SCOPE string is "DEFAULT".

Scopes are the primary means an administrator has to scale SLP deployments to larger networks. When DAs with NON-DEFAULT scopes are present on the network, further gains can be had by configuring UAs and SAs to have a predefined non-default scope. These agents can then perform DA discovery and make requests using their scope. This will limit the number of replies.

11.1. Scope Rules

SLP messages which fail to contain a scope that the receiving Agent is configured to use are dropped (if the request was multicast) or a SCOPE_NOT_SUPPORTED error is returned (if the request was unicast). Every SrvRqst (except for DA and SA discovery requests), SrvReg, AttrRqst, SrvTypeRqst, DAAdvert, and SAAdvert message MUST include a <scope-list>.

A UA MUST unicast its SLP messages to a DA which supports the desired scope, in preference to multicasting a request to SAs. A UA MAY multicast the request if no DA is available in the scope it is configured to use.

11.2. Administrative and User Selectable Scopes

All requests and services are scoped. The two exceptions are SrvRqsts for "service:directory-agent" and "service:service-agent". These MAY have a zero-length <scope-list> when used to enable the user to make scope selections. In this case UAs obtain their scope list from DAAdverts (or if DAs are not available, from SAAdverts.)

Otherwise, if SAs and UAs are to use any scope other than the default (i.e., "DEFAULT"), the UAs and SAs are configured with lists of scopes to use by system administrators, perhaps automatically by way of DHCP option 78 or 79 [21]. Such administrative scoping allows services to be provisioned, so that users will only see services they are intended to see.

User configurable scopes allow a user to discover any service, but require them to do their own selection of scope. This is similar to the way AppleTalk [12] and SMB [19] networking allow user selection of AppleTalk Zone or workgroups.

Note that the two configuration choices are not compatible. One model allows administrators control over service provision. The other delegates this to users (who may not be prepared to do any configuration of their system).

12. Directory Agents

DAs cache service location and attribute information. They exist to enhance the performance and scalability of SLP. Multiple DAs provide further scalability and robustness of operation, since they can each store service information for the same SAs, in case one of the DAs fails.

A DA provides a centralized store for service information. This is useful in a network with several subnets or with many SLP Agents. The DA address can be dynamically configured with UAs and SAs using DHCP, or by using static configuration.

SAs configured to use DAs with DHCP or static configuration MUST unicast a SrvRqst to the DA, when the SA is initialized. The SrvRqst omits the scope list and sets the service type of the request to "service:directory-agent". The DA will return a DAAdvert with its attributes, SLP SPI list, and other parameters which are essential for proper SA to DA communication.

Passive detection of DAs by SAs enables services to be advertised consistently among DAs of the same scope. Advertisements expire if not renewed, leaving only transient stale registrations in DAs, even

in the case of a failure of a SA.

A single DA can support many UAs. UAs send the same requests to DAs that they would send to SAs and expect the same results. DAs reduce the load on SAs, making simpler implementations of SAs possible.

UAs MUST be prepared for the possibility that the service information they obtain from DAs is stale.

12.1. Directory Agent Rules

When DAs are present, each SA MUST register its services with DAs that support one or more of its scope(s).

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

DAs MUST flush service advertisements once their lifetime expires or their URL Authentication Block "Timestamp" of expiration is past.

DAAdverts MUST include DA Stateless Boot Timestamp, in the same format as the Authentication Block (see Section 9.2). The Timestamp in the Authentication Block indicates the time at which all previous registrations were lost (i.e., the last stateless reboot). The Timestamp is set to 0 in a DAAdvert to notify UAs and SAs that the DA is going down. DAs MUST NOT use equal or lesser Boot Timestamps to previous ones, if they go down and restart without service registration state. This would mislead SAs to not reregister with the DA.

DAs which receive a multicast SrvRqst for the service type "service:directory-agent" MUST silently discard it if the <scope-list> is (a) not omitted and (b) does not include a scope they are configured to use. Otherwise the DA MUST respond with a DAAdvert.

DAs MUST respond to AttrRqst and SrvTypeRqst messages (these are OPTIONAL only for SAs, not DAs.)

12.2. Directory Agent Discovery

UAs can discover DAs using static configuration, DHCP options 78 and 79, or by multicasting (or broadcasting) Service Requests using the convergence algorithm in Section 6.3.

See Section 6 regarding unsolicited DAAdverts. Section 12.2.2 describes how SAs may reduce the number of times they must reregister with DAs in response to unsolicited DAAdverts.

DAs MUST send unsolicited DAAdverts once per CONFIG_DA_BEAT. An unsolicited DAAdvert has an XID of 0. SAs MUST listen for DAAdverts, passively, as described in Section 8.5. UAs MAY do this. If they do not listen for unsolicited DAAdverts, however, they will not discover DAs as they become available. UAs SHOULD, in this case, do periodic active DA discovery, see Section 6.

A URL with the scheme "service:directory-agent" indicates the DA's location as defined in Section 8.5. For example: "service:directory-agent://foobawooba.org".

The following sections suggest timing algorithms which enhance the scalability of SLP.

12.2.1. Active DA Discovery

After a UA or SA restarts, its 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 8.1. DA Discovery requests MUST include a Previous Responder List. SrvRqsts for Active DA Discovery SHOULD NOT be sent more than once per CONFIG_DA_FIND seconds.

After discovering a new DA, a SA MUST wait a random time between 0 and CONFIG_REG_ACTIVE seconds before registering their services.

12.2.2. Passive DA Advertising

A DA MUST multicast (or broadcast) an unsolicited DAAdvert every CONFIG_DA_BEAT seconds. CONFIG_DA_BEAT SHOULD be specified to prevent DAAdverts from using more than 1% of the available bandwidth.

All UAs and SAs which receive the unsolicited DAAdvert SHOULD examine its DA stateless Boot Timestamp. If it is set to 0, the DA is going down and no further messages should be sent to it.

If a SA detects a DA it has never encountered (with a nonzero timestamp,) the SA must register with it. SAs MUST examine the DAAdvert's timestamp to determine if the DA has had a stateless reboot since the SA last registered with it. If so it registers with the DA. SAs MUST wait a random interval between 0 and CONFIG_REG_PASSIVE before beginning DA registration.

12.3. Reliable Unicast to DAs and SAs

If a DA or SA fails to respond to a unicast UDP message in CONFIG_RETRY seconds, the message should be retried. The wait interval for each subsequent retransmission MUST exponentially increase, doubling each time. If a DA or SA fails to respond after CONFIG_RETRY_MAX seconds, the sender should consider the receiver to have gone down. The UA should use a different DA. If no such DA responds, DA discovery should be used to find a new DA. If no DA is available, multicast requests to SAs are used.

12.4. DA Scope Configuration

By default, DAs are configured with the "DEFAULT" scope. Administrators may add other configured scopes, in order to support UAs and SAs in non default scopes. The default configuration MUST NOT be removed from the DA unless:

- There are other DAs which support the "DEFAULT" scope, or

- All UAs and SAs have been configured with non-default scopes.

Non-default scopes can be phased-in as the SLP deployment grows. Default scopes should be phased out only when the non-default scopes are universally configured.

If a DA and SA are coresident on a host (quite possibly implemented by the same process), configuration of the host is considerably simplified if the SA supports only scopes also supported by the DA. That is, the SA SHOULD NOT advertise services in any scopes which are not supported by the coresident DA. This means that incoming requests can be answered by a single data store; the SA and DA registrations do not need to be kept separately.

12.5. DAs and Authentication Blocks

DAs are not configured to sign service registrations or attribute lists. They simply cache services registered by Service Agents. DAs MUST NOT accept registrations including authentication blocks for SLP SPIs which it is not configured with, see Section 8.5.

A DA protects registrations which are made with authentication blocks using SLP SPIs it is configured to use. If a service S is registered, a subsequent registration (which will replace the adertisement) or a deregistration (which will remove it) MUST include an Authentication Block with the corresponding SLP SPI, see Section 8.3 and Section 10.6.

Example:

A DA is configured to be able to verify Authentication Blocks with SLP SPIs "X,Y", that is X and Y.

An SA registers a service with an Authentication Block with SPI "Z". The DA stores the registration, but discards the Authentication Block. If a UA requests a service with an SLP SPI string "Z", the DA will respond with an AUTHENTICATION_UNKNOWN error.

An SA registers a service S with Authentication Blocks including SLP SPIs "X" and "Y". If a UA requests a service with an SLP SPI string "X" the DA will be able to return S (if the service type, language, scope and predicate of the SrvRqst match S) The DA will also return the Authentication Block with SLP SPI set to "X". If the DA receives a subsequent SrvDeReg for S (which will remove the advertisement) or a subsequent SrvReg for S (which will replace it), the message must include two URL Authentication Blocks, one each for SPIs "X" and "Y". If either of these were absent, the DA would return an AUTHENTICATION_ABSENT error.

13. Protocol Timing Defaults

Interval name	Section	Default Value	Meaning
CONFIG_MC_MAX	6.3	15 seconds	Max time to wait for a complete multicast query response (all values.)
CONFIG_START_WAIT	12.2.1	3 seconds	Wait to perform DA discovery on reboot.
CONFIG_RETRY	12.3	2 seconds	Wait interval before initial retransmission of multicast or unicast requests.
CONFIG_RETRY_MAX	12.3	15 seconds	Give up on unicast request retransmission.
CONFIG_DA_BEAT	12.2.2	3 hours	DA Heartbeat, so that SAs passively detect new DAs.
CONFIG_DA_FIND	12.3	900 seconds	Minimum interval to wait before repeating Active DA discovery.
CONFIG_REG_PASSIVE	12.2	1-3 seconds	Wait to register services on passive DA discovery.
CONFIG_REG_ACTIVE	8.3	1-3 seconds	Wait to register services on active DA discovery.
CONFIG_CLOSE_CONN	6.2	5 minutes	DAs and SAs close idle connections.

14. Optional Configuration

Broadcast Only
Any SLP agent SHOULD be configurable to use broadcast only. See Sections 6.1 and 12.2.

Predefined DA
A UA or SA SHOULD be configurable to use a predefined DA.

No DA Discovery
The UA or SA SHOULD be configurable to ONLY use predefined and DHCP-configured DAs and perform no active or passive DA discovery.

Multicast TTL
The default multicast TTL is 255. Agents SHOULD be configurable to use other values. A lower value will focus the multicast convergence algorithm on smaller subnetworks, decreasing the number of responses and increases the performance of service location. This may result in UAs obtaining different results for the identical requests depending on where they are connected to the network.

Timing Values
Time values other than the default MAY be configurable. See Section 13.

Scopes
A UA MAY be configurable to support User Selectable scopes by omitting all predefined scopes. See Section 11.2. A UA or SA MUST be configurable to use specific scopes by default. Additionally, a UA or SA MUST be configurable to use specific scopes for requests for and registrations of specific service types. The scope or scopes of a DA MUST be configurable. The default value for a DA is to have the scope "DEFAULT" if not otherwise configured.

DHCP Configuration
DHCP options 78 and 79 may be used to configure SLP. If DA locations are configured using DHCP, these SHOULD be used in preference to DAs discovered actively or passively. One or more of the scopes configured using DHCP MUST be used in requests. The entire configured <scope-list> MUST be used in registration and DA configuration messages.

Service Template
UAs and SAs MAY be configured by using Service Templates. Besides simplifying the specification of attribute values, this also allows them to enforce the inclusion of 'required' attributes in SrvRqst, SrvReg and SrvDeReg messages. DAs MAY be configured with templates to allow them to WARN UAs and SAs in these cases. See Section 10.4.

SLP SPI for service discovery
Agents SHOULD be configurable to support SLP SPIs using the following parameters: BSD=2 (DSA with SHA-1) and a public key identified by the SLP SPI String. In the future, when a Public Key Infrastructure exists, SLP Agents may be able to obtain public keys and cryptographic parameters corresponding to the names used in SLP SPI Strings.

Note that if the SLP SPI string chosen is identical to a scope string, it is effectively the same as a Protected Scope in SLPv1. Namely, every SA advertising in that scope would be configured with the same Private Key. Every DA and UA of that scope would be configured with the appropriate Public Key to verify signatures produced by those SAs. This is a convenient way to configure SLP deployments in the absence of a Public Key Infrastructure. Currently, it would be too difficult to manage the keying of UAs and DAs if each SA had its own key.

SLP SPI for Directory Agent discovery
Agents SHOULD be configurable to support SLP SPIs as above, to be used when discovering DAs. This SPI SHOULD be sent in SrvRqsts to discover DAs and be used to verify multicast DAAdvert messages.

SA and DA Private Key
SAs and DAs which can generate digital signatures require a Private Key and a corresponding SLP SPI indentifier to include in the Authentication Block. The SLP SPI identifies the Public Key to use to verify the digital signature in the Authentication Block.

15. IANA Considerations

SLP includes four sets of identifiers which may be registered with IANA. The policies for these registrations (See [18]) are noted in each case.

The Block Structure Descriptor (BSD) identifies the format of the Authenticator which follows. BSDs 0x8000-0x8FFF are for Private Use.

Further Block Structured Descriptor (BSD) values, from the range 0x0003-0x7FFF may be standardized in the future by submitting a document which describes:

- The data format of the Structured Authenticator block.
- Which cryptographic algorithm to use (including a reference to a technical specification of the algorithm.)
- The format of any keying material required for preconfiguring UAs, DAs and SAs. Also include any considerations regarding key distribution.
- Security considerations to alert others to the strengths and weaknesses of the approach.

The IANA will assign Cryptographic BSD numbers on the basis of IETF Consenus.

New function-IDs, in the range 12-255, may be standardized by the method of IETF Consensus.

New SLP Extensions with types in the range 2-65535 may be registered following review by a Designated Expert.

New error numbers in the range 15-65535 are assigned on the basis of a Standards Action.

Protocol elements used with Service Location Protocol may also require IANA registration actions. SLP is used in conjunction with "service:" URLs and Service Templates [13]. These are standardized by review of a Designated Expert and a mailing list (See [13].)

16. Internationalization Considerations

SLP messages support the use of multiple languages by providing a Language Tag field in the common message header (see Section 8).

Services MAY be registered in multiple languages. This provides attributes so that users with different language skills may select services interactively.

Attribute tags are not translated. Attribute values may be translated unless the Service Template [13] defines the attribute values to be 'literal'.

A service which is registered in multiple languages may be queried in multiple languages. The language of the SrvRqst or AttrRqst is used to satisfy the request. If the requested language is not supported, a LANGUAGE_NOT_SUPPORTED error is returned. SrvRply and AttrRply messages are always in the same language of the request.

A DA or SA MAY be configured with translations of Service Templates [13] for the same service type. This will allow the DA or SA to translate a request (say in Italian) to the language of the service advertisement (say in English) and then translate the reply back to Italian. Similarly, a UA MAY use templates to translate outgoing requests and incoming replies.

The dialect field in the Language Tag MAY be used: 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.

17. Security Considerations

SLP provides for authentication of service URLs and service attributes. This provides UAs and DAs with knowledge of the integrity of service URLs and attributes included in SLP messages. The only systems which can generate digital signatures are those which have been configured by administrators in advance. Agents which verify signed data may assume it is 'trustworthy' inasmuch as administrators have ensured the cryptographic keying of SAs and DAs reflects 'trustworthiness.'

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.

If Agents are not configured to generate Authentication Blocks and Agents are not configured to verify them, an adversary might easily use this protocol to advertise services on servers controlled by the adversary and thereby gain access to users' private information. Further, an adversary using this protocol will find it much easier to engage in selective denial of service attacks. Sites that are in potentially hostile environments (e.g., are directly connected to the Internet) should consider the advantages of distributing keys associated with SLP SPIs prior to deploying the sensitive directory agents or service agents.

SLP 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. SLP 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 Authentication Blocks if cryptographic information and SLP SPIs can be preconfigured with the end systems before they use SLP.

SLPv2 enables a number of security policies with the mechanisms it includes. A SLPv2 UA could, for instance, reject any SLP message which did not carry an authentication block which it could verify. This is not the only policy which is possible to implement.

A. Appendix: Changes to the Service Location Protocol from v1 to v2

SLP version 2 (SLPv2) corrects race conditions present in SLPv1 [22]. In addition, authentication has been reworked to provide more flexibility and protection (especially for DA Advertisements). SLPv2 also changes the formats and definition of many flags and values and reduces the number of 'required features.' SLPv2 clarifies and changes the use of 'Scopes', eliminating support for 'unscoped directory agents' and 'unscoped requests'. SLPv2 uses LDAPv3 compatible string encodings of attributes and search filters. Other changes (such as Language and Character set handling) adopt practices recommended by the Internet Engineering Steering Group.

Effort has been made to make SLPv2 operate the same whether DAs are present or not. For this reason, a new message (the SAAdvert) has been added. This allows UAs to discover scope information in the absence of administrative configuration and DAs. This was not possible in SLPv1.

SLPv2 is incompatible in some respects with SLPv1. If a DA which supports both SLPv1 and SLPv2 with the same scope is present, services advertised by SAs using either version of the protocol will be available to both SLPv1 and SLPv2 UAs. SLPv1 DAs SHOULD be phased out and replace with SLPv2 DAs which support both versions of the protocol.

SLPv1 allows services to be advertised and requested without a scope. Further, DAs can be configured without a scope. This is incompatible with SLPv2 and presents scalability problems. To facilitate this forward migration, SLPv1 agents MUST use scopes for all registrations and requests. SLPv1 DAs MUST be configured with a scope list. This constitutes a revision of RFC 2165 [22].

B. Appendix: Service Discovery by Type: Minimal SLPv2 Features

Service Agents may advertise services without attributes. This will enable only discovery of services by type. Service types discovered this way will have a Service Template [13] defined which specifies explicitly that no attributes are associated with the service advertisement. Service types associated with Service Templates which specify attributes MUST NOT be advertised by SAs which do not support attributes.

While discovery of service by service type is a subset of the features possible using SLPv2 this form of discovery is consistent with the current generation of products that allow simple browsing of all services in a 'zone' or 'workgroup' by type. In some cases, attribute discovery, security and feature negotiation is handled by

application layer protocols - all that is required is the basic discovery of services that support a certain service.

UAs requesting only service of that service type would only need to support service type and scope fields of the Service Request. UAs would still perform DA discovery and unicast SLPv2 SrvRqst messages to DAs in their scope once they were discovered instead of multicasting them.

SAs would also perform DA discovery and use a SLPv2 SrvReg to register all their advertised services with SLPv2 DAs in their scope. These advertisements would needless to say contain no attribute string.

These minimal SAs could ignore the Language Tag in requests since SrvRqst messages would contain no attributes, hence no strings would be internationalized. Further, any non-null predicate string would fail to match a service advertisement with no attributes, so these SAs would not have to parse and interpret search filters. Overflow will never occur in SrvRqst, SrvRply or SrvReg messages so TCP message handling would not have to be implemented. Finally, all AttrRqst messages could be dropped by the SA, since no attributes are supported.

C. Appendix: DAAdverts with arbitrary URLs

Using Active DA Discovery, a SrvRqst with its service type field set to "service:directory-agent". DAs will respond with a DAAdvert containing a URL with the "service:directory-agent:" scheme. This is the same DAAdvert that such a DA would multicast in unsolicited DA advertisements.

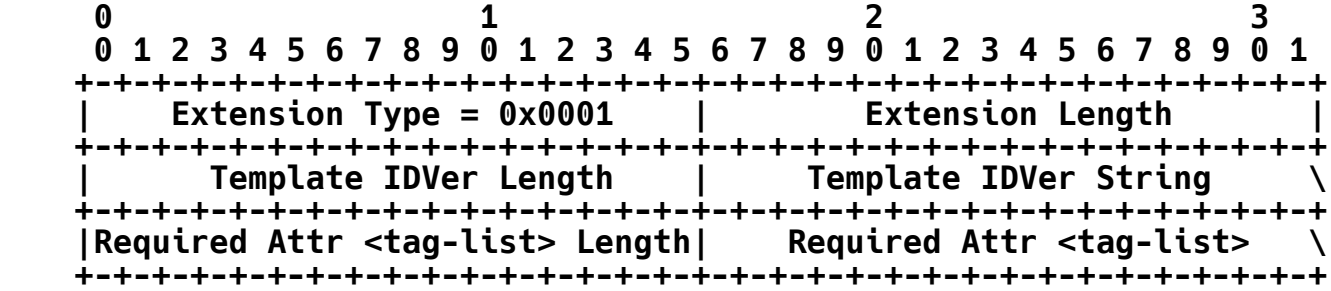
A UA or SA which receives an unsolicited DAAdvert MUST examine the URL to determine if it has a recognized scheme. If the UA or SA does not recognize the DAAdvert's URL scheme, the DAAdvert is silently discarded. This document specifies only how to use URLs with the "service:directory-agent:" scheme.

This provides the possibility for forward compatibility with future versions of SLP and enables other services to advertise their ability to serve as a clearinghouse for service location information.

For example, if LDAPv3 [15] is used for service registration and discovery by a set of end systems, they could interpret a LDAP URL [16] to passively discover the LDAP server to use for this purpose. This document does not specify how this is done: SLPv2 agents without further support would simply discard this DAAdvert.

D. Appendix: SLP Protocol Extensions

D.1. Required Attribute Missing Option



Required attributes and the format of the IDVer string are defined by [13].

If a SA or DA receives a SrvRqst or a SrvReg which fails to include a Required Attribute for the requested Service Type (according to the Service Template), it MAY return the Required Attribute Extension in addition to the reply corresponding to the message. The sender SHOULD reissue the message with a search filter including the attributes listed in the returned Required Attribute Extension. Similarly, the Required Attribute Extension may be returned in response to a SrvDereg message that contains a required attribute tag.

The Template IDVer String is the name and version number string of the Service Template which defines the given attribute as required. It SHOULD be included, but can be omitted if a given SA or DA has been individually configured to have 'required attributes.'

The Required Attribute <tag-list> MUST NOT include wild cards.

E. Acknowledgments

This document incorporates ideas from work on several discovery protocols, including RDP by Perkins and Harjono, and PDS by Michael Day. We are grateful for contributions by Ye Gu and Peter Ford. John Veizades was instrumental in the standardization of the Service Location Protocol. Implementors at Novell, Axis Communications and Sun Microsystems have contributed significantly to make this a much clearer and more consistent document.

F. References

[1] Port numbers, July 1997.
ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers.

[2] ISO/IEC JTC1/SC 21. Certificate Extensions. Draft Amendment DAM 4 to ISO/IEC 9594-2, December 1996.

[3] ISO/IEC JTC1/SC 21. Certificate Extensions. Draft Amendment DAM 2 to ISO/IEC 9594-6, December 1996.

[4] ISO/IEC JTC1/SC 21. Certificate Extensions. Draft Amendment DAM 1 to ISO/IEC 9594-7, December 1996.

[5] ISO/IEC JTC1/SC 21. Certificate Extensions. Draft Amendment DAM 1 to ISO/IEC 9594-8, December 1996.

[6] Unicode Technical Report #8. The Unicode Standard, version 2.1. Technical report, The Unicode Consortium, 1998.

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

[8] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[9] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

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

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

[13] Guttman, E., Perkins, C. and J. Kempf, "Service Templates and service: Schemes", RFC 2609, June 1999.

[14] Howes, T., "The String Representation of LDAP Search Filters", RFC 2254, December 1997.

[15] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[16] Howes, T. and M. Smith, "The LDAP URL Format", RFC 2255, December 1997.

[17] Meyer, D., "Administratively Scoped IP Multicast", RFC 2365, July 1998.

[18] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs, BCP 26, RFC 2434, October 1998.

[19] Microsoft Networks. SMB File Sharing Protocol Extensions 3.0, Document Version 1.09, November 1989.

[20] National Institute of Standards and Technology. Digital signature standard. Technical Report NIST FIPS PUB 186, U.S. Department of Commerce, May 1994.

[21] Perkins, C. and E. Guttman, "DHCP Options for Service Location Protocol", RFC 2610, June 1999.

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

[23] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

G. Authors' Addresses

Erik Guttman
Sun Microsystems
Bahnstr. 2
74915 Waibstadt
Germany

Phone: +49 7263 911 701
EMail: Erik.Guttman@sun.com

Charles Perkins
Sun Microsystems
901 San Antonio Road
Palo Alto, CA 94040
USA

Phone: +1 650 786 6464
EMail: cperkins@sun.com

John Veizades
@Home Network
425 Broadway
Redwood City, CA 94043
USA

Phone: +1 650 569 5243
EMail: veizades@home.net

Michael Day
Vinca Corporation.
1201 North 800 East
Orem, Utah 84097 USA

Phone: +1 801 376-5083
EMail: mday@vinca.com

H. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.