Network Working Group
Request for Comments: 3082
Category: Experimental

J. Kempf
J. Goldschmidt
Sun Microsystems

March 2001

Notification and Subscription for SLP

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Service Location Protocol (SLP) provides mechanisms whereby service agent clients can advertise and user agent clients can query for services. The design is very much demand-driven, so that user agents only obtain service information when they specifically ask for it. There exists another class of user agent applications, however, that requires notification when a new service appears or disappears. In the RFC 2608 design, these applications are forced to poll the network to catch changes. In this document, we describe a protocol for allowing such clients to be notified when a change occurs, removing the need for polling.

1. Introduction

The Service Location Protocol (SLP) [1] provides a mechanism for service agent (SA) clients to advertise network services and for user agent (UA) clients to find them. The mechanism is demand-driven. UAs obtain service information by actively querying for it, and do not obtain any information unless they do so. While this design satisfies the requirements for most applications, there are some applications that require more timely information about the appearance or disappearance in the services of interest.

Ideally, these applications would like to be notified when a new service comes up or when a service disappears. In order to obtain this information with SLP as described in RFC 2608, such applications must poll the network to periodically refresh their local cache of available service advertisements.

An example of such a client is a desktop GUI that wants to display network service icons as soon as they appear to provide users with an accurate picture of all services available to them.

Because polling is inefficient and wasteful of network and processor resources, we would like to provide these applications a mechanism whereby they can be explicitly notified of changes. In this document, we describe a scalable mechanism allowing UAs to be notified of changes in service availability.

2. 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 [2].

3. Terminology

In this section, we present some additional terminology beyond that in [1] and [3].

Notification - A message sent to an interested agent informing that agent that a service has appeared or disappeared.

Subscription - A request to be informed about changes in service availability for a particular service type and scopes.

4. Design Considerations

The primary design consideration in a notification protocol for SLP is that we would like it to exhibit the same high degree of scalability and robustness that the base SLP protocol exhibits. Notification should work in small networks with only a few SAs, as well as large enterprise networks with thousands of SAs and hundreds of DAs. Small networks should not be required to deploy DAs in order to receive the benefits of notification. We also want to assure that notification in large networks does not cause heavy processing loads to fall on any one particular SLP agent. This requires that the task of notification be distributed rather than centralized, to avoid loading down one agent with doing all the notification work. Finally, we would like the notification scheme to be robust in the face of DA failures, just as the base SLP design is.

An important consideration is that the UA clients obtain notifications of SA events in a timely fashion. If a UA has subscribed to notification for a particular service type, the UA should receive such notification regardless of the state of intervening DAs. SLP is transparent with respect to DAs supporting a

particular scope; that is, a UA can use any DA with a particular scope and expect to get the same service advertisements. Notifications should exhibit the same property. Whether or not a UA receives a notification should not depend on the DA to which they happen to connect. This preserves the DAs' identity as a pure cache.

Another goal is that the notification messages contain enough information about the triggering event that the UA can determine whether or not it is of interest in the large majority of cases without having to issue another SLP request a priori. The UA may, of course, issue an SLP request for related reasons, but it should not have to issue a request to obtain more information on the event that triggered the notification in most cases. This reduces the amount of network traffic related to the event.

In order to simplify implementation, we would like to use similar mechanisms for notification in large and small networks. The mechanisms are not identical, obviously, but we want to avoid having radically different mechanisms that require completely separate implementations. Having similar mechanisms reduces the amount of code in UA and SA clients.

A minor goal is to make use of existing SLP message types and mechanisms wherever possible. This reduces the amount of code necessary to implement the notification mechanism, because much code can be reused between the base SLP and the notification mechanism. In particular, we expect to make use of the SLP extension mechanism in certain cases to support subscription.

5. Notification Design Description

In order to support scalability, we split the design into two parts. A small network design is used when no DAs are present in the network. A large network design is used in networks with DAs. The following subsections describe the two designs.

5.1 Small Network Design

In networks without DAs, UAs are notified by an SA when the SA initially appears, and when the SA disappears. This allows UAs to know about the list of service types the SA supports. In small networks, there is no centralized agent available to administer subscriptions for newly appearing SAs. This rules out any kind of subscription design in which a UA subscribes to notifications for a particular service type in particular scopes of interest, because a newly appearing SA can't tell whether or not there are any subscriptions without a centralizing agent to tell it.

As a result, SAs perform notification when they come on line and prior to shutting down regardless of their scope or service type, if they are capable of performing notification. This means that a UA receives notification of all types of changes for all scopes and service types, and consequently must be prepared to filter out those changes in which it is not interested (other scopes, other service types).

The design requires SAs to perform notification by IP multicasting (or broadcasting in IPv4 if multicast is not available) SLP SrvReg or SrvDereg messages using the multicast transmit algorithm described in Section 9.0. The port number for notifications is not the default SLP port, because that port is only accessible to privileged users on some operating systems, but rather the port 1847, as assigned by IANA.

In IPv4, the SA performs multicast on the SLP multicast address (239.255.255.253, default TTL 255) and is administratively scoped in the same manner as SLP [4]. IPv4 UAs interested in notification join the multicast group 239.255.255.253 and listen on port 1847. In IPv6, the multicast is performed to the scoped IPv6 addresses for the service type advertised, as described in [8]. The SA advertises on all addresses up to and including the largest multicast scope that it supports. IPv6 UAs interested in notification join the multicast groups corresponding to the multicast scopes and service type in which they are interested and listen on port 1847. For example, an IPv6 UA that has access to site local scope and is interested in a service type whose hash is 42, calculated according to the algorithm in [8], joins the groups FF01:0:0:0:0:0:10042 through FF05:0:0:0:0:0:10042.

5.2 Large Network Design

In networks with DAs, a DA supporting a particular scope can act as an intermediary for administering UA subscriptions. A subscription consists of a service type and a collection of scopes. A UA interested in being notified about changes in a particular service type attaches the Subscribe extension to a SrvRqst message sent to the DA. The DA obtains multicast group addresses for notification based on the algorithm described in Section 8.0 and puts them into a NotifyAt extension which it attaches to the SrvRply. The UA listens on the group addresses in the reply for notifications.

When a new subscription comes in, existing SAs are informed about the subscription using the following procedure. The DA compares the service type and scopes in the new subscription against a list of existing subscriptions. If no previous subscription has the same service type and scopes, the DA MUST multicast a DAAdvert, using the

multicast transmit algorithm described in Section 9.0, and MUST include the NotifyAt extension with the multicast group addresses for notification. If an existing subscription covers the same service type and scopes as the new subscription, the DA MUST NOT multicast a DAAdvert.

A DA MUST keep track of subscriptions it has arranged as well as subscriptions arranged by other DAs in any scopes with which the DA is configured. To avoid multiple multicast NotifyAt messages, a DA MUST wait a random amount of time, uniformly distributed between 0 and 3 seconds before sending the multicast DAAdvert with NotifyAt. During this period, the DA MUST listen for NotifyAt messages that match the one from the new subscription. If a matching NotifyAt is detected, the DA MUST not multicast.

When a new SA registers with a DA that has existing subscriptions, the new SA is informed of notifications it should perform using the following procedure. If the service type and scopes in the new SA's SrvReg messages match an existing subscription, a NotifyAt containing the multicast addresses for notification MUST be included in the SrvAck. If the SA doesn't support notification, it simply ignores the extension. If the service type and scopes in the new SA's SrvReg do not match any existing subscriptions, the DA MUST NOT include a NotifyAt.

The DA itself MUST also perform notification, according to the multicast transmit algorithm, when a service advertisement times out. Time-out of a service advertisement results in the DA multicasting a SrvDereg for the deregistered URL. This allows interested UAs to be informed of the service advertisement's demise even if the SA has disappeared without deregistering. A DA MUST NOT perform notification when it receives a SrvReg from an SA, however, that is the job of the SA.

As in small networks, notification is performed primarily by SAs. If an SA receives a DAAdvert or SrvAck with a NotifyAt extension and the following conditions are met:

1. The SA supports notification.

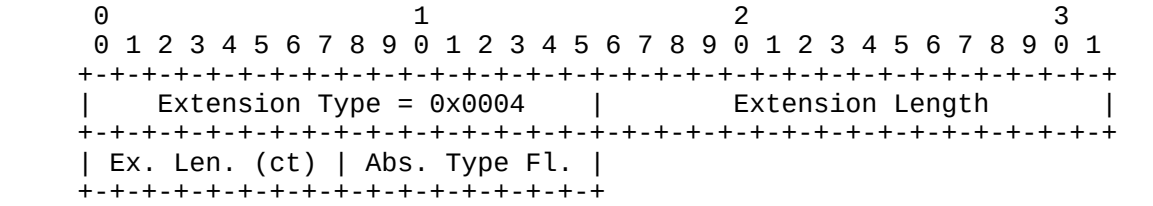
2. The SA's service type matches the service type in the NotifyAt extension.

3. The SA's scopes match one of the scopes of the NotifyAt extension.

then the SA saves the multicast addresses that correspond to the scopes and service types it supports. The SA MUST perform notification immediately after the SA has performed the SrvReg or SrvDereg with the DA. An SA that has detected a DA in its scopes MUST NOT multicast any notifications unless it receives a NotifyAt extension in a SrvAck with service type and scopes matching the SA's service type and scopes.

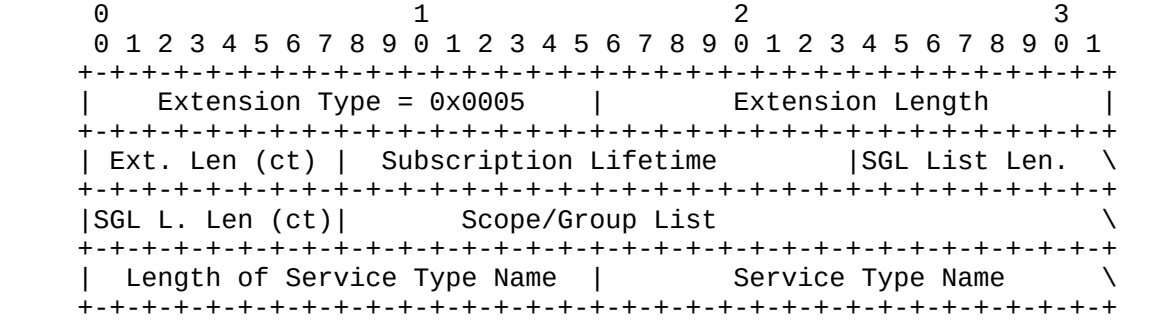
6. Subscribe Extension

The Subscribe extension has the following format:



The scope list and service type of the extension are taken from the accompanying SrvRqst. The abstract type flag indicates whether the UA is interested in hearing from all SAs advertising concrete instances of an abstract type [3], and is only of interest if the service type in the SrvRqst is a concrete type. If the flag is 1, the UA is interested in hearing from all SAs advertising concrete types having the same abstract type as the type of the SrvRqst. If the flag is 0, the UA is only interested in hearing from SAs supporting the particular concrete type in the SrvRqst. If the service type in the accompanying SrvRqst is not a concrete type, the flag is ignored.

7. NotifyAt Extension



The service type name is in the same format as in the SrvRqst. The scope/group list is a list of scope names and multicast group addresses. The following ABNF [5] syntax describes the list:

```
sglist          = sgitem / sgitem "," sglist
sgitem          = scope-name ":" ip-addr
ip-addr         = ipv4-number | ipv6-number
scope-name      =  ; See RFC 2608 for the format of scope names.
ipv4-number     =  1*3DIGIT 3("." 1*3DIGIT)
ipv6-number     = ;See RFC 2373 [9] Section 2.2
```

An example of a scope/group list for IPv4 is:

```
eng:239.255.255.42,corp:239.255.255.43
```

An example of a scope/group listfor IPv6 is:

```
eng:FF02:0:0:0:0:0:1:1042,corp:FF03:0:0:0:0:0:1:1042
```

The scope/group list gives the multicast addresses to use for notifications involving the service type for the given scopes.

The service type name can be a simple type name, an abstract type name, or a concrete type name. If the name is an abstract type name, all SAs advertising the abstract type MUST notify. If the name is a concrete or simple type name, ONLY those SAs advertising the simple or concrete type MUST notify, others MUST NOT notify. A DA that receives a subscription for a concrete type with the abstract type flag set, MUST include the abstract type name in all the NotifyAt messages it sends. If the DA receives a subscription for a concrete type with the abstract type flag not set, the DA MUST NOT include the abstract type, but rather MUST include the concrete type name.

There are three cases in which an agent may receive a NotifyAt extension: in a SrvRply returned to a UA, in a multicast DAAdvert, and in a SrvAck returned to an SA. The three subsections below describe the response in each of these cases.

7.1 NotifyAt received with SrvRply

When a UA sends a SrvRqst with a Subscribe extension, the DA responds with a SrvRply including a NotifyAt. The DA MUST NOT unicast a NotifyAt to a UA with any other message and MUST NOT send a NotifyAt unless a SrvRqst with a Subscribe extension was received.

The UA responds by setting up a multicast listener to the group addresses included in the extension on the SLP notification port 1847. The UA MAY also want to note the expiration lifetime of the

subscription assigned by the DA, and reissue a subscription before the lifetime expires.

7.2 NotifyAt received with Multicast DAAdvert

The DA multicasts a NotifyAt with a DAAdvert using the multicast transmit algorithm when a UA has requested notification and the scopes and service type in the subscription were not previously seen. This message informs existing SAs having the service type and scopes in the announcement that they should multicast notifications when they shut down.

A receiving SA participating in notification responds by noting the multicast address if the service type and scopes match. When the SA is about to go down, the SA MUST first unicast a SrvDereg without attribute tag list to its DAs (as per standard SLP), then it MUST multicast the same SrvDereg message according to the multicast transmit algorithm. The SA MUST cease performing notification when the subscription lifetime expires, unless a subsequent NotifyAt is received prolonging the subscription.

A UA that is performing passive DA detection will naturally also receive the extension, but the UA SHOULD ignore the extension.

7.3 NotifyAt received with SrvAck

An SA can receive a NotifyAt with a SrvAck when it first comes up and registers itself with a DA. If the DA has any subscriptions from UAs for the service type and scopes represented by the SA, it MUST return a NotifyAt with the SrvAck.

The SA upon receiving the NotifyAt immediately multicasts the same SrvReg it sent to the DA, according to the multicast transmit algorithm. The SA MUST only perform the multicast algorithm once, even if it registers with more than one DA and receives the NotifyAt in reply from more than one. Prior to its demise and after deregistering with a DA, the SA MUST notify with the same SrvDereg, as described in Section 7.2.

8. Multicast Address Allocation

Enterprise networks that allow SLP notification SHOULD deploy the Multicast Address Allocation Architecture (MAAA) including administratively scoped multicast and Multicast Address Dynamic Client Allocation Protocol (MADCAP) [6].

If it is not possible to obtain a multicast address for use in SLP notifications, the SLP multicast address is used.

If the MAAA infrastructure is deployed, DAs and SAs obtain their scope configuration from MADCAP, because the SLP scopes are the same as the MADCAP scopes. Each SLP scope MUST correspond to a multicast scope name, in the sense of [6]. In such a case, a DA allocates, using MADCAP, a new multicast group address for each new service type/scope pair to which a UA subscribes. The allocation is made by MADCAP from the multicast address range for the scope. In this way, only those UAs interested in the service type and scopes in the subscription receive the multicast notification. The DA sets up the lease on the multicast address to correspond with the duration of the subscription. If the MADCAP server runs out of addresses, the SLP multicast group is used as a last resort.

For example, if the multicast scope has an address range of 239.1.0.0 through 239.1.255.255, the notification group address for service type X in scope A could be 239.1.0.42 and for service type Y in scope B could be 239.1.42.42.

9. Multicast Transmit Algorithm

The DA and SAs use a multicast transmit algorithm similar to that used for discovering services in SLP, described in RFC 2608 [1], except the agent performing the notification doesn't wait for replies. The agent performing the notification transmits a notification message repeatedly over a period of 15 seconds, backing off exponentially on the duration of the time interval between the multicasts. The rationale for this algorithm is to limit the duration and scope of the multicast announcement while still repeating the announcement enough times to increase the probability that one message gets through.

For an SA, a notification message is either a SrvReg or a SrvDereg message, depending on whether the SA is registering a new service or deregistering a service. When a new service is registered, the SrvReg message MUST have the fresh bit set in the SLP header. The entire list of attributes for the service SHOULD be included. The SrvDereg message MUST NOT include an attribute tag list. Notifications MUST NOT be transmitted at any other time, to minimize multicast traffic.

Since a SrvReg could contain attribute lists of arbitrary length, the message could potentially overflow the packet MTU for UDP. If an attribute list causes a packet MTU overflow, the SA MUST set the overflow bit in the SLP header. The attribute list in the notification message MUST be formatted so that a UA can use the attributes even if an overflow occurs. If a UA needs more attributes than are transmitted in the notification message, it can contact the SA (if no DA is present) or the DA for the attributes it needs.

A DA multicasts a DAAdvert when a subscription comes in containing a service type and scopes that do not match any on the DA's list of known subscriptions. The same algorithm MUST be used. If the combination of the DA attributes and the NotifyAt message cause the DAAdvert to overflow a UDP packet, DA attributes MUST be truncated to allow the NotifyAt to fit and the overflow bit MUST be set in the header. An SA knows that the purpose of the message is to inform it of a new subscription rather than for passive advertisement, because of the extension, and it can therefore ignore the DA attribute list field if the overflow bit is set in the header. A DA also transmits a SrvDereg message when a service advertisement is deregistered due to timeout, following the same rules as for an SA.

10.0 DA Disappearance

Robustness to DA failure is an important goal of the design. When a DA disappears due to unforeseen circumstances, subscription information from UAs is lost. UAs continue to get notifications from existing SAs. However, new SAs will not be informed of the subscription unless other DAs also have the subscription information. Because a UA may not discover a new DA until it tries to perform an active request, the UA could potentially miss the appearance of new services. For this reason, UAs that are concerned about receiving notification of absolutely every service that appears SHOULD issue subscriptions to every newly discovered DA that supports the scopes it supports. Similarly, if a DA disappears through controlled shutdown, a UA performing passive discovery can detect the shutdown and reissue the subscription to an alternate DA.

On the SA side, when a DA goes down, existing SAs continue to notify until the subscription expires. Before ceasing to notify, an SA MUST determine whether the DA is still active and, if not, verify with another DA whether the subscription has been extended. If no other DA is available, the SA MUST ignore the subscription expiration time and continue notifying until a new DA is discovered. When a new DA is discovered the SA must send a new SrvReg to the DA, according to RFC 2608 [1]. The replying SrvAck contains a NotifyAt extension if the UA has renewed its subscription with the DA. If the SrvAck does not contain a NotifyAt message the SA MUST continue to notify until the subscription expires. If a UA is interested in continuing the notification, it renews the subscription with the new DA prior to the expiration of the old one, and so the SA is informed to continue notifying.

Note that this procedure still does not inform SAs that come up between the time a newly booted DA comes up and the time the UA has renewed its subscription with the newly booted DA. If this situation is of concern, multiple DAs can be used to assure that all subscriptions are covered when a DA goes down.

11. Network Administration Considerations

In SLP networks with DAs as described in RFC 2608, the only multicast is the SrvRqst for DAAdverts performed during active DA discovery, and unsolicited DAAdverts sent periodically by the DA for passive discovery. There is no multicast involved in UA queries or SA registrations. This allows network administrators to set up DAs for a particular collection of IP subnets and confine all service discovery traffic to unicast between the SA and UA clients and the DA. Administratively scoped multicast can additionally be used to limit the extent of active DA discovery and passive DA advertising. The amount of multicast involved is not high and DHCP DA and scope configuration can be used to limit which DAs a particular UA or SA client sees, or to inhibit multicast entirely so that UAs and SAs only use configured DAs.

With notification, however, multicast traffic involving events in SAs becomes available. Because DAs request multicast addresses based on scope and service type, the multicast associated with particular events should only propagate to those subnets in which UAs and SAs of the same scope are interacting. Routers should be configured with administrative multicast scoping to limit multicast. If DAs are not deployed (or the MAAA is not deployed), however, the amount of multicast on the SLP multicast address when notifications are being used could quickly become very large. Therefore, it is crucial that DAs supporting notification be deployed in large networks where UA clients are interested in notification.

12. Security Considerations

The SrvReg and SrvDereg messages contain authentication blocks for all SLP SPIs supported by the DAs with which the SA registers. Since these SPIs are necessarily the same as those that UAs can verify, a UA receiving a multicast notification is in a position to verify the notification. It does so by selecting the authentication block or blocks that it can verify. If authentication fails, either due to lack of an authentication block, or lack of the proper SPI, the UA simply discards the notification. In a network without DAs, the SPIs of the UA and SA must also match.

13. IANA Considerations

The SLP Notification services use the IANA-assigned port number of 1847. The SLP extension identifiers assigned by IANA are 0x0004 for Subscribe and 0x0005 for NotifyAt.

14. Acknowledgements

The authors would like to thank Charles Perkins, of Nokia, and Erik Guttman and Jonathan Wood, of Sun Microsystems, for their stimulating discussion and suggestions during the initial phases of the subscription/notification design. We would also like to thank Erik for his intense scrutiny of the specification during the later phases. His comments were instrumental in refining the design. Shivaun Albright, of HP, motivated simplification of the protocol to focus on initial registration and deregistration only. Vaishali Mithbaokar implemented the simplified protocol.

15. References

[1] Guttman, E., Perkins, C., Veizades, J. and M. Day, "Service Location Protocol", RFC 2608, July 1999.

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

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

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

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

[6] Hanna, S., Patel,B. and M. Shah, "Multicast Address Dynamic Client Allocation Protocol (MADCAP)", RFC 2730, December 1999.

[7] http://www.isi.edu/in-notes/iana/assignments/multicast-addresses

[8] Guttman, E., "Service Location Protocol Modifications for IPv6", Work in Progress.

[9] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2375, July 1997.

16. Author's Addresses

James Kempf
Sun Microsystems
UMPK15-214
901 San Antonio Rd.
Palo Alto, CA 94040
USA

Phone: +1 650 786 5890
EMail: james.kempf@sun.com

Jason Goldschmidt
Sun Microsystems
UMPK17-202
901 San Antonio Rd.
Palo Alto, CA 94040
USA

Phone: +1 650 786 3502
EMail: jason.goldschmidt@sun.com

Full Copyright Statement

Acknowledgement

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