Network Working Group
Request for Comments: 3421
Category: Experimental

W. Zhao
H. Schulzrinne
Columbia University
E. Guttman
Sun Microsystems
C. Bisdikian
W. Jerome
IBM

November 2002

Select and Sort Extensions for the Service Location Protocol (SLP)

Status of this Memo

Copyright Notice

Abstract

This document defines two extensions (Select and Sort) for the Service Location Protocol (SLP). These extensions allow a User Agent (UA) to request that the Uniform Resource Locator (URL) entries in a Service Reply (SrvRply) be limited to the specified number, or be sorted according to the specified sort key list. Using these two extensions together can facilitate discovering the best match, such as finding a service that has the maximum speed or the minimum load.

1. Introduction

This document defines two extensions (Select and Sort) for SLP [RFC2608]. These extensions allow a UA to request that the URL entries in a SrvRply be limited to the specified number, or be sorted according to the specified sort key list. A Directory Agent (DA) or Service Agent (SA) that supports the Select and/or Sort extensions MUST include the attribute keyword "select-enabled" and/or "sort-enabled" in its advertisement (DAAdvert or SAAdvert). A UA SHOULD check these attributes of the contacting DA/SA before it uses the Select and/or Sort extensions to query the DA/SA.

Using the Select extension, a UA can opt for finding just a few (not necessarily all) matching services, which is useful if the UA uses a low-bandwidth channel. Using the Sort extension, a UA can ask the DA/SA to sort matching URL entries, which helps the UA to choose a service from multiple candidates. Sorting by the server is more efficient than sorting by the client since for sorting purposes, the former does not need to pass the attributes of matching URLs to the client. Furthermore, using the Select and Sort extensions together can facilitate discovering the best match, such as finding a service that has the maximum speed or the minimum load, or has a speed closest to a reference value.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted according to in BCP 14, RFC 2119 [RFC2119].

2. Select Extension

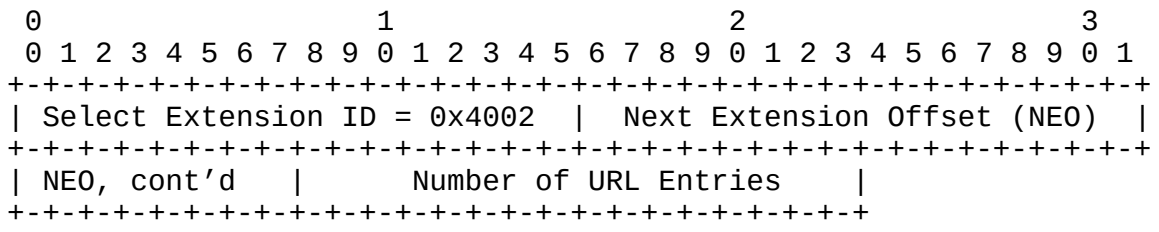

Figure 1. Select Extension

The format of the Select extension is shown in Figure 1. A UA uses this extension in a Service Request (SrvRqst) to limit the maximum number (say, n) of URL entries to be returned. When a DA/SA receives a SrvRqst with a Select extension, it MUST use a Select extension in the corresponding SrvRply to indicate the total number (say, m) of matching URL entries if the DA/SA supports this extension, otherwise the DA/SA MUST set the error code in the corresponding SrvRply to OPTION_NOT_UNDERSTOOD [RFC2608]. If n < m, then only the first n matching URL entries are returned, else all m matching URL entries are returned. As a special case, a UA may set n to zero to obtain the number of matching URL entries without retrieving the entries themselves.

We denote a Select extension as Select(number). Thus, Select(3) means that the corresponding SrvRply can have at most three URL entries.

3. Sort Extension

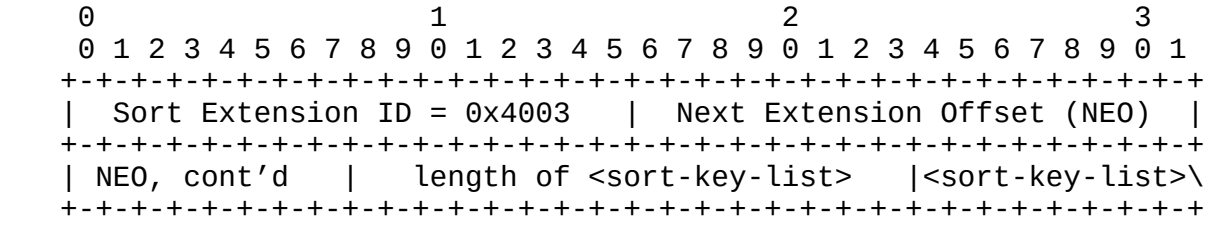

Figure 2. Sort Extension

The format of the Sort extension is shown in Figure 2. A UA uses this extension in a SrvRqst to request the URL entries in the corresponding SrvRply be sorted according to the sort-key-list. The sort-key-list is defined using Augmented Backus-Naur Form (ABNF) [RFC2234] as follows:

```
sort-key-list  = sort-key / sort-key "," sort-key-list
sort-key       = key-name ":" type ":" ordering [":" ref-value]
key-name       = attr-tag from Section 5 of RFC 2608
type           = "s" / "i"
                 ; "s" for string type
                 ; "i" for integer type
ordering       = "+" / "-"
                 ; "+" for increasing order
                 ; "-" for decreasing order
ref-value      = intval from Section 5 of RFC 2608
```

Each sort-key in the sort-key-list has a key-name, a type specifier, an ordering specifier, and an optional reference value. The key-name MUST be a valid attribute name, and its type is explicitly specified. Although SLP has five attribute types (integer, string, boolean, opaque and keyword), we only consider integer sort and string sort since keyword attributes (they have no values) never need to be sorted, and boolean and opaque attributes can be sorted as strings if needed. The integer sort uses the integerOrderingMatch rule defined in X.520 [X520], whereas the string sort is performed based on lexical ordering. Strings are compared using the rules defined in Section 6.4 of RFC 2608.

Only integer keys may have a reference value, causing the sort to be based on the distance to the reference value. A reference-based sort, such as "X:i:+:12", requires the following two steps:

Step 1. For each matching service, if its attribute X has a value of x, then use |x-12| as its metric.

Step 2. Use the metrics obtained in Step 1 to sort attribute X for matching services.

The SLP sort rules are adapted from the Lightweight Directory Access Protocol (LDAP) sort rules defined in RFC 2891 [RFC2891]. Note that sort in SLP is a best effort, no sort error will be returned from a DA/SA to a UA.

(1) The sort-key-list is in order of highest to lowest sort key precedence (Section 1.1 of RFC 2891).

(2) Each attribute SHOULD only occur in the sort-key-list once (Section 1.1 of RFC 2891). If an attribute is included in the sort-key-list multiple times, only its first occurrence is considered, all other occurrences are ignored.

(3) For a multi-valued attribute, the least value in each entry SHOULD be used in sort (Section 2.2 of RFC 2891).

(4) An entry missing one or more of the sort keys is treated as having NULLs for those missing keys (Section 2.2 of RFC 2891).

(5) NULL is considered as a larger value than all other valid values (Section 2.2 of RFC 2891).

(6) As the attribute type in SLP is not enforced, an attribute may have inconsistent values. For the purpose of sorting, inconsistent values may exist only when an attribute is sorted as integer. Inconsistent values SHOULD be treated as NULLs.

When a DA/SA receives a SrvRqst with a Sort extension, it MUST set the error code in the corresponding SrvRply to OPTION_NOT_UNDERSTOOD [RFC2608] if the DA/SA does not support the Sort extension or cannot perform the requested sort. The DA/SA sets the error code in the corresponding SrvRply to zero if it has successfully processed the SrvRqst and performed the requested sort.

We denote a Sort extension as Sort(sort-key-list). The following examples illustrate how to use the Sort extension.

o Integer sort on speed (decreasing order).

Sort(speed:i:-)

[Note] "i" means integer sort, and "-" means decreasing order.

- Integer sort on load (increasing order) and speed (decreasing order).

  ```
  Sort(load:i:+,speed:i:-)
  ```

  [Note] "+" means increasing order.

- String sort on model (increasing order).

  ```
  Sort(model:s:+)
  ```

  [Note] "s" means string sort.

- Integer sort on speed (increasing order), based on a reference value 12.

  ```
  Sort(speed:i:+:12)
  ```

  [Note] For example, if we have four matching services, with the "speed" attribute as 8 (URL1), 10 (URL2), 12 (URL3), and 15 (URL4), then the sorted URL list will be "URL3,URL2,URL4,URL1" (based on the metric ordering of |12-12| < |12-10| < |12-15| < |12-8|).

4. Using the Select and Sort Extensions Together

In addition to being used individually, the Select and Sort extensions can be used together to facilitate discovering the best match, such as finding a service with the maximum speed. When these two extensions appear in the same SrvRqst message, they MUST be processed in the order of their presence. We show some examples next.

- Find the service with the minimum load

  ```
  Sort(load:i:+)
  Select(1)
  ```

- Find the three fastest services

  ```
  Sort(speed:i:-)
  Select(3)
  ```

- Find the service with the minimum load among the three fastest

  ```
  Sort(speed:i:-)
  Select(3)
  Sort(load:i:+)
  Select(1)
  ```

o Find the service that has a speed closest to 12

```
Sort(speed:i:+:12)
Select(1)
```

5. IANA Considerations

The Select and Sort extension IDs, 0x4002 and 0x4003, described in Section 2 and Section 3, respectively, have been assigned by IANA out of the SLP extension space (RFC 2608, Section 9.1) reserved for "mandatory to implement" extensions (i.e., the 0x4000-0x7FFF range).

6. Security Considerations

There are no new security issues beyond those described in RFC 2608.

7. Acknowledgments

Ira McDonald provided good suggestions.

8. Normative References

[RFC2608] Guttman, E., Perkins, C., Veizades, J. and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[RFC2119] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

9. Non-normative References

[X520] International Telephone Union, "The Directory: Selected Attribute Types", X.520, 1997.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2891] Howes, T., Wahl, M. and A. Anantha, "LDAP Control Extension for Server Side Sorting of Search Results", RFC 2891, August 2000.

10. Authors' Addresses

Weibin Zhao
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027-7003

EMail: zwb@cs.columbia.edu

Henning Schulzrinne
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027-7003

EMail: hgs@cs.columbia.edu

Erik Guttman
Sun Microsystems
Eichhoelzelstr. 7
74915 Waibstadt
Germany

EMail: Erik.Guttman@sun.com

Chatschik Bisdikian
IBM Corp.
Thomas J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA

EMail: bisdik@us.ibm.com

William F. Jerome
IBM Corp.
Thomas J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA

EMail: wfj@us.ibm.com

11. Full Copyright Statement

Acknowledgement

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