RC22098 (W0106-024) June 20, 2001
Computer Science

IBM Research Report

Customization for SLP Service Request and Reply

Weibin Zhao, Henning Schulzrinne
Columbia University

Chatschik Bisdikian, William Jerome
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

INTERNET DRAFT
draft-zhao-slp-customization-00.txt
Expires: December 20, 2001

Weibin Zhao
Henning Schulzrinne
Columbia University
Chatschik Bisdikian
William Jerome
IBM
June 20, 2001

Customization for SLP Service Request and Reply
draft-zhao-slp-customization-00.txt

Status of This Memo

Copyright Notice

Abstract

This document presents a lightweight mechanism for supporting customization in SLP where a UA specifies its customization request in the SrvRqst via an SLP extension, and the DA customizes the SrvRply according to the UA request. Two basic customization operations, sorting and bounding the result set, are provided, and some complex customization requests are supported by composing these two basic operations. Furthermore, customized comparators are enabled to decide the result order. Customizing SLP SrvRqst and SrvRply

messages enhances the basic SLP discovery scenario by also considering user information and preference, and thus it can support value-added services and improve the SLP query efficiency.

1. Introduction

In the Service Location Protocol (SLP [1]), a User Agent (UA) discovers a desired service by specifying its properties (type, scope and attribute predicate) via a Service Request (SrvRqst) message, and a Directory Agent (DA) answers with a Service Reply (SrvRply) message carrying a list of URL entries for the matched services. Although performing discovery based on service properties is sufficient for most applications, there are some applications that also need to incorporate context information (such as location) and user preference (such as order and size of the result set) in the SrvRqst and customize the SrvRply to tailor it to the user request.

Customizing SLP SrvRqst and SrvRply messages can provide several advantages. First, it enhances the basic SLP discovery scenario by also considering user information, and thus it can support value-added services, such as location-based discovery (find a service that is close to the user). Second, bounding the result size (number of URL entries) may be useful when the UA has limited resources or the UA uses a low-bandwidth channel. In some cases, a user may just want to find a few services, not tens or hundreds of them. Third, sorting the result set on some attribute(s) by a DA is more efficient than sorting it by a UA when some ordered result set is needed. For example, if a UA wants to find the available printers in order of speed (pages per minute), using the basic SLP queries, it needs to first send a SrvRqst to get a list of printers, then issue an Attribute Request (AttrRqst) for each printer to get its speed, and finally sort the printer list on speed. The overhead of multiple round message exchange and round-trip delay suggests that a sort at the DA side is more suitable for this discovery request.

In this document, we will present a lightweight mechanism to support customization in SLP where a UA specifies its customization request in the SrvRqst via an SLP extension, and the DA customizes the SrvRply according to the UA request. Two basic customization operations, sorting and bounding the result set, are provided, and some complex customization requests are supported by composing these two basic operations. Furthermore, customized comparators are enabled to decide the result order.

The rest of this document is organized as follows: we first define terminology in Section 2, then present a design overview in Section 3. Section 4 defines the Customization extension and Section 5 defines the Comparator extension. We list constants in Section 6 and

give security considerations in Section 7.

2. Terminology

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 [3].

Customization Request
: A customization request is specified via an SLP extension, which MAY be used in a SrvRqst. It describes the preferred way in which the DA SHOULD customize the result set for the corresponding SrvRply.

Reference-based Comparator
: A reference-based comparator is a customized comparison function associated with an attribute of a service template [2], which MAY be used to compare the attribute value with a reference value. This comparison function receives two arguments (the second one is the reference value), and it returns the difference of the two arguments as a non-negative value. If If two compared values are equal, it returns 0.

3. Design Overview

The processing of a SrvRqst can be viewed as having two stages: matching service properties to obtain a result set (may be empty), then customizing this result set according to the user request. Note that a customization may have effects only if the original result set has more than one URL entries. In other words, if the original result set is empty or has only one URL entry, then any customization will produce no effect, and the customization request can be safely ignored in this case.

3.1. Sorting and Bounding the Result Set

Sorting and bounding the result set are two basic types of customization. The sort operation refers to the natural sorting of alphanumeric characters (sort numbers on value or sort strings on lexicographic order). The bound operation refers to selecting the first N elements (assume bound N) in a list. If the number of elements in the list is less than N, then the bound operation returns the whole list.

We support sort and bound directly, and some complex customizations by composing these two basic operations. For example, maximum speed (or minimum load) can be expressed as a sort followed by a selection

of the first entry, and minimum load in the top-three fast machines can be expressed as a {sort, selection, sort, selection} sequence. The sort can be on one attribute or multiple attributes.

Note that similar customization operations, server side sort [5] and paged result manipulation [6], are supported in LDAP [4]. However, LDAP does not address composing these basic operations.

3.2. Using Customized Comparator

Sometimes the natural sort is not sufficient, a customized comparator is needed to decide the order. One example is the reference-based comparator which compares an attribute value with a reference value, and returns a non-negative value as the difference metric for the comparison. If two compared values are equal, it returns 0.

The reference-based comparator is useful for supporting location-based discovery which needs to compare a service location with a reference location. The difference metric returned by a reference-based comparator is application dependent, i.e., application specific information is needed to decide what is a best match for a reference value. For example, if there is a printer in room 442 and room 458, respectively, which printer is closer to a user in room 449?

A generic interface for the reference-based comparator is as follows.

```
public float comparator(String s1, String s2)
/* compare s1 with s2,
   return a non-negative value as the difference metric
   if s1 and s2 are equal, then return 0
*/
```

4. Customization Extension

This extension is used in the SrvRqst message to specify the customization request on the result set. Figure 1 gives its format.

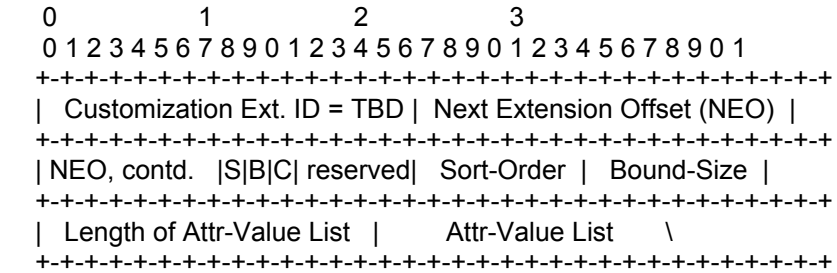

Figure 1. Customization Extension

The customization request is specified using three basic operations: sorting (S) the result set, bounding (B) the result size and calling (C) the customized comparator. These actions can be used individually or combined in some way. An operation is selected by setting its corresponding bit to 1. The attribute-value list is needed only by the sort (S) and call (C) operations: sort is based on all specified attributes, and call is applied to attributes that have a value. If a customization only has the bound (B) operation, then the length of the attribute-value list is zero.

S: sort the result set on the specified attributes in the Sort-Order.

The Sort-Order is given in a bit-field where 0 is used for increasing and 1 is used for decreasing sorting order. The Sort-Order field can specify at most 8 attribute sorting orders corresponding to their positions in the attribute list. For example, 0x00 means that all attributes are sorted in increasing order, and 0xA0 means the first and the third attribute are sorted in decreasing order.

B: bound the result size to the Bound-Size.

C: call the customized comparator(s) for those attributes that have a reference value. Note that an attribute can have at most one customized comparator.

When multiple attributes are present in the attribute list, they are in order of highest to lowest sort key precedence.

When multiple Customization extensions are present in a SrvRqst message, they are processed in sequence.

4.1. Examples

We use a tuple (action-flag, sort-order, bound-size, attribute-list) to represent a Customization extension. A customization that needs multiple Customization extensions is illustrated as multiple tuples.

Example 1. sort in decreasing speed:
(sort, 0x80, NA, {speed})

Example 2. bound to three URLs
(bound, NA, 3, NA)

Example 3. one minimum load:
(sort & bound, 0x00, 1, {load})

Example 4. top three fast speed:
(sort & bound, 0x80, 3, {speed})

Example 5. one minimum load in the top three fast speed:
(sort & bound, 0x80, 3, {speed}) and
(sort & bound, 0x00, 1, {load})

Example 6. sort in decreasing speed and increasing load:
(sort, 0x80, NA, {speed,load})

Example 7. the nearest service:
(sort & bound & call, 0x00, 1, {location})

4.2. Client-Server Interaction

A UA MAY use the Customization extension in a unicast SrvRqst sent to a DA to specify its customization request. However, a UA SHOULD NOT use this extension in a multicast SrvRqst since each SA will answer a SrvRply individually, and no customization can be made by SAs.

For a SrvRqst that has the Customization extension, a DA MUST return an OPTION_NOT_UNDERSTOOD [1] error if the DA does not support the Customization extension or it cannot perform the requested customization.

5. Comparator Extension

This extension is used in the SrvReg message to specify a customized comparison function for an attribute of a service template. Figure 2 gives its format.

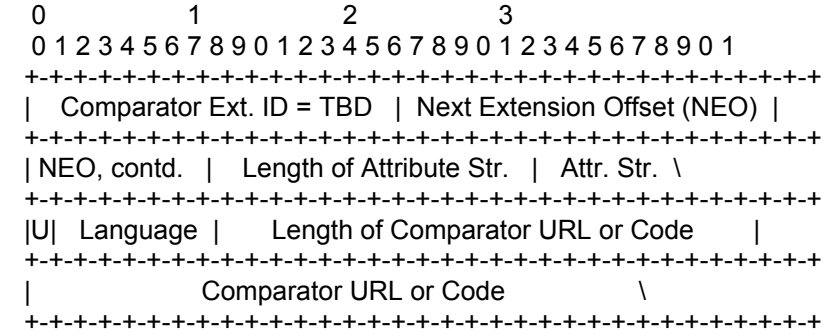

Figure 2. Comparator Extension

If the U bit is set to 1, then the comparator is specified via a URL, otherwise its code is given in this extension. The Language field specifies a platform independent programming language (such as Java and script language) in which the comparator is written. Two proposed languages are Java (1) and Perl (2).

5.1. Client-Server Interaction

For a SrvReg that has the Comparator extension, a DA MUST return an OPTION_NOT_UNDERSTOOD error if the DA does not support the Comparator extension, cannot download the program code from the specified URL, or does not understand the programming language.

6. Constants

Customization Extension ID	TBD	(Section 4)
Comparator Extension ID	TBD	(Section 5)

7. Security Considerations

Before accepting a customized comparator, a DA SHOULD verify the program. As a registered comparator MAY crash, a DA SHOULD handle failure properly so that a failed comparator will not crash the whole system.

8. References

[1] E. Guttman, C. Perkins, J. Veizades and M. Day, "Service location protocol, version 2", RFC 2608, June 1999.

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

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

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

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

[6] C. Weider, A. Herron, A. Anantha and T. Howes, "LDAP Control Extension for Simple Paged Results Manipulation", RFC 2696, September, 1999.

9. Authors' Addresses

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

Email: {zwb,hgs}@cs.columbia.edu

Chatschik Bisdikian
William F. Jerome
IBM T. J. Watson Research Center
P.O.Box 218
Yorktown Heights, NY 10598-0218
Email: {bisdik,wfj}@us.ibm.com

10. Full Copyright Statement