Service Location Protocol Administration Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part Number 806-1412–10
Februrary 2000

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227–14(g)(2)(6/87) and FAR 52.227–19(6/87), or DFAR 252.227–7015(b)(6/95) and DFAR 227.7202–3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Adobe PostScript

Please Recycle

Contents

Preface

The *Service Location Protocol System Administration Guide* provides information about configuring and managing the Service Location Protocol (SLP) V2 framework installed in your Solaris operating environment. This book assumes that you have already installed the SunOS™ 5.8 operating system, and you have set up any networking software that you plan to use. The SunOS 5.8 operating system is part of the Solaris product family, which also includes many features, including the Solaris Common Desktop Environment (CDE). The SunOS 5.8 operating system is compliant with AT&T's System V, Release 4 operating system.

Note - The Solaris operating environment runs on two types of hardware, or platforms: SPARC™ and IA. The Solaris operating environment runs on both 64-bit and 32-bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems running the Solaris 8 release. To use this book, you should have 1-2 years of UNIX system administration experience. Attending UNIX® system administration training courses might be helpful.

How This Book Is Organized

Chapter 1 provides an overview of the Service Location Protocol architecture implementation.

Chapter 2 describes planning considerations for the SLP operation specific to your site.

Chapter 3 describes how to configure the various SLP properties using the SLP configuration file.

Chapter 4 describes how to configure the SLP network-related properties for refining the SLP deployment.

Chapter 5 describes how to create and implement SLP scopes and how to tailor service access in your enterprise.

Chapter 6 describes how to deploy directory agents for load balancing and how to tailor service access in the enterprise.

Chapter 7 describes how to incorporate legacy services (network services that pre-date the development and implementation of SLP).

Chapter 8 describes how to configure multihomed hosts effectively in your SLP deployment.

Appendix A provides reference information about the SLP message types.

Appendix B provides a comprehensive list and descriptions of SLP status codes used for communication between agents.

Related Books

For complete information about SLP, refer to the following documents:

- Kempf, James, and Pete St. Pierre. *Service Location Protocol for Enterprise Networks.* Wiley and Son, Inc. (ISBN # 0–47–3158–7)
- *Authentication Management Infrastructure Administration Guide* (part # 805–1139–03)
- Guttman, Erik, Chareles Perkins, John Veizades, and Michael Day. *Service Location Protocol, Version 2,* from the Internet Engineering Task Force (IETF). Available on-line at `http://www.ietf.org/ietf/lid-abstracts.txt.`

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com^{SM} Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

What Typographic Conventions Mean

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	**Meaning**	**Example**
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

TABLE P–1 Typographic Conventions *(continued)*

Typeface or Symbol	Meaning	Example
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. Do *not* save changes yet.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER **1**

Overview of SLP

The Service Location Protocol (SLP) provides a portable, platform-independent framework for the discovery and provisioning of SLP-enabled network services on IP intranets. This chapter provides an overview of the SLP architecture and describes the Solaris SLP implementation.

- "SLP Architecture" on page 11
- "How SLP Is Implemented" on page 15

SLP Architecture

This section describes the fundamental operation of SLP, its constituent parts, and what these parts do for your enterprise. SLP facilitates the following:

- Client application requests for network service location information
- Advertisement of services
- Organization of services and users into logical or functional groups
- Managed recovery from primary server failures

SLP provides all of these services automatically, with little or no configuration. Additionally, you can configure SLP to assist with administration and tune SLP operation, if needed.

For instance, you can enable SLP logging mechanisms to monitor and troubleshoot the SLP operation on your network. You can tune certain properties to synchronize timing on SLP message exchanges between agents. Additionally, you can suppress SLP multicasts to reduce network congestion. Further, you can provision services by placing certain SLP agents, called, *directory agents,* strategically throughout the enterprise by creating groupings of users, called *scopes*, through which you can tailor service provisioning to meet the needs of your enterprise.

Summary of the SLP Design

In SLP, various software-based agents represent user applications and network services. SLP maintains information about the nature and location of enterprise services, so that the information about the SLP entities on the enterprise is automatically updated. Additionally, SLP is capable of advertising services that are not SLP-enabled using proxy registrations (see Chapter 7).

SLP Agents and Processes

The following table describes the agents and processes that make up the SLP framework.

TABLE 1–1 SLP Agents and Processes

Directory Agent	The directory agent (DA) is an optional SLP agent that stores and maintains a cache of service advertisements sent by the service agent (SA). When deployed, the DA resolves user agent (UA) service requests. The DA responds to active solicitations from the SA and UA for directory advertisements. Consequently, the SA and UA discover the DAs and *scopes* (see below) with which they are associated. A DA sends periodic unsolicited advertisements through which UAs and SAs discover the DA within shared scopes.
Service Agent	The service agent (SA) maintains service advertisements for a networked service. If no DA is available, the SA answers multicast service requests from UAs. If a DA is available, the SA registers and, optionally, deregisters services with DAs that support its scopes.
User Agent	The user agent (UA) acts on behalf of the user application and queries for the identity of its corresponding scopes, directory agents, and service advertisements.
Scope	A scope is a grouping of UAs and SAs arranged either administratively, topologically, or in another organizational manner. You can use scopes to tailor how you provision service access across the enterprise (see Chapter 5).

TABLE 1–1 SLP Agents and Processes *(continued)*

SLP daemon (slpd)	In the Solaris implementaion of SLP, `slpd` is a daemon process that acts as either a DA or an SA server. Service processes on a host register their service advertisments with `slpd` rather than maintaining the advertisements individually. These service processes contain an SA client library that communicates with `slpd`, acting as the SA server. `slpd` forwards all registrations and deregistrations to DAs, and times out expired service advertisements. `slpd` also maintains a table of available DAs by performing active and passive DA discovery. Through these mechanisms, DA information is provided to UA clients. UA clients use `slpd` on their host only for DA information. Each SLP host automatically has an `slpd` daemon. You can optionally configure `slpd` as a DA.
Legacy Services	A legacy service is a networked service that is not SLP-enabled. Legacy services can be supported, provided that a proxy registration is created for them. Proxy registrations enable you to register non-SLP-enabled services with SLP. SLP-based clients can then discover legacy services (see Chapter 7).
Service Advertisements	A service advertisement is a URL and a collection of attribute/value list pairs that describe a service. All service advertisements have a lifetime. After the lifetime expires, the service advertisement is no longer valid, unless reregistered.
Service URL	A service URL is used to advertise the network location of services. It contains the service type name, host name, or network address of the system hosting the service, optional port number, and other information necessary to use the service.

The following figure shows the basic agents and processes that implement the SLP architecture.

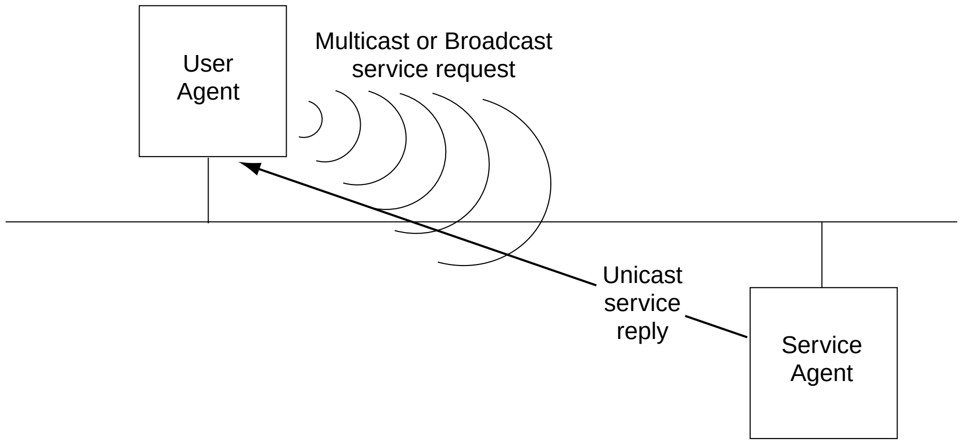

Figure 1–1 SLP Basic Agents and Processes

The figure above also represents a default deployment of SLP in which no configuration has been done. Only two agents are required, the UA and SA. The SLP framework allows the UA to multicast requests for services to the SA. When the SA receives requests for a service that it advertises, it unicasts a reply to the UA containing the service advertisement.

The following figure shows the basic agents and processes that implement the SLP architecture when a DA is deployed in the framework.

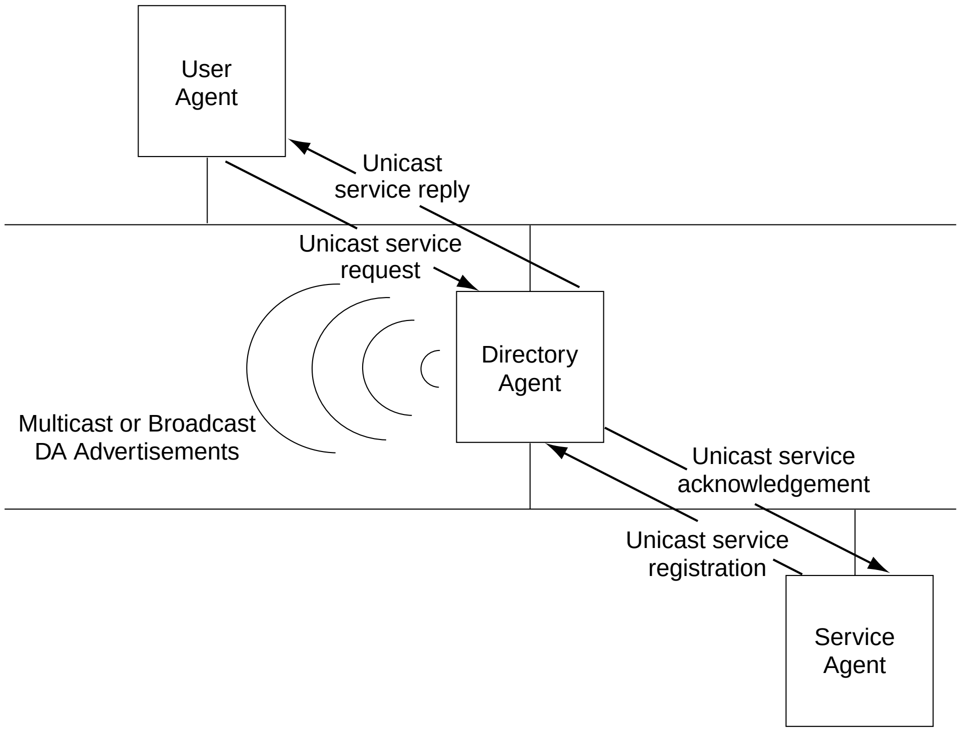
User Agent
Unicast service reply
Unicast service request
Directory Agent
Multicast or Broadcast DA Advertisements
Unicast service acknowledgement
Unicast service registration
Service Agent

Figure 1–2 SLP Architectural Agents and Processes Implemented with a DA

In more complex enterprises, one or more DAs are used. The DA serves as a cache for registered service advertisements. SAs send register messages (`SrvReg`) that list all the services they advertise to DAs and receive acknowledgments in reply (`SrvAck`). The service advertisements are refreshed with the DA, or they expire according to the lifetime set for the advertisement. Once a UA discovers a DA, UAs unicast requests to DAs rather than multicasting requests to SAs.

For more information about Solaris SLP messages, refer to Appendix A.

How SLP Is Implemented

In the Solaris SLP implementation, the SLP SAs, UAs, DAs, SA servers, scopes, and other architectural components (listed in Table 1–1) are mapped partially into `slpd` and partially into application processes. The SLP daemon, `slpd`, organizes certain off-host SLP interactions. The SLP daemon:

- Employs passive and active directory agent discovery for all UAs and SAs on the local host

- Maintains a table of DAs and keeps it current for the use of the UAs and SAs on the local host
- Acts as a proxy SA server for legacy service advertisements (proxy registration)
- Can be configured to act as a DA

For more information about the SLP daemon, see `slpd`(1M).

In addition to `slpd`, the UA and SA client libraries (`libslp.so` for C/C++ and `slp.jar` for Java) provide UA and SA clients with access to the SLP framework. The client libraries:

- Provide communication to register and deregister service advertisements between SA clients and `slpd`
- Provide UA clients with the capability to make UA requests
- Provide communication of DA accessibility information between `slpd` and UA clients

In the following figure, the SLP client library in the Service Provider Program implements SA functionality. The Service Provider Program uses the SLP client library to register services with `slpd`, and to deregister them. The SLP client library in the Service Client Program implements UA functionality. The Service Client Program uses the SLP client library to issue multicast and unicast (to DAs) service requests and to query `slpd` for information on DAs. The `slpd` process takes care of all SA functionality, such as answering multicast requests, registering with DAs, and so on.

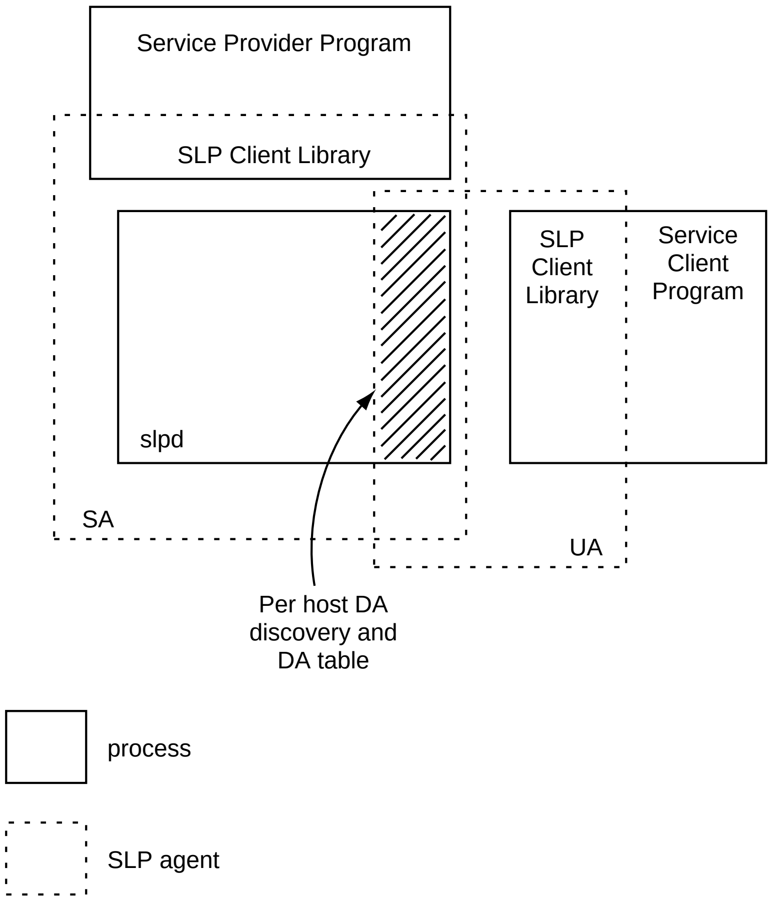

Figure 1–3 SLP Implementation

Enabling SLP

SLP is enabled by running the SLP daemon, `slpd`. The supported interface for starting `slpd` is the `/etc/init.d/slpd` script, which starts the daemon only if the SLP configuration file, `/etc/inet/slp.conf`, exists. The Solaris operating environment includes the file `/etc/inet/slp.conf.example`. Rename this file to `/etc/inet/slp.conf` to enable SLP at boot time.

CHAPTER 2

Planning the Solaris SLP Deployment

This chapter lists SLP planning considerations for your deployment and describes how to use the Solaris `snoop` utility to determine whether any of the SLP properties would benefit from additional configuration.

SLP Configuration Considerations

The Solaris 8 release is shipped with SLP installed and pre-configured with default property settings. If your enterprise operates well with the original SLP settings, the SLP deployment requires virtually no administration. However, in some cases, you might want to alter the default SLP settings to tune SLP network operation or activate various SLP features. For example, with a few configuration changes, you can enable logging for SLP activity exclusively, then check the log file to determine whether additional reconfiguration is needed.

SLP configuration properties reside in the `slp.conf` file located in the `/etc/inet` directory. If you determine that some reconfiguration of the default property settings is needed, after reading the information in this chapter and analyzing the efficiency of SLP activity on the system, then consult the remaining chapters for the appropriate procedures.

Before you alter any of the SLP configuration property settings, consider the following:

- Types of network technologies operating in the enterprise
- Types of traffic the technologies handle smoothly

- Number and types of network services
- Number of users and the services they use
- Locations of users in relation to their most frequently accessed services

Monitoring SLP Activity

The `snoop` utility is a passive administrative tool that provides network traffic information while generating minimal network traffic itself. Invoking `snoop` enables you to watch all activity on your network as it occurs.

The `snoop` utility provides a printout of the actual SLP message traffic. For example, using `snoop` with the `slp` command line argument, the utility displays SLP traces of registrations and deregistrations by which you can gauge the kinds of services being registered and the amount of reregistration activity, which in turn enables you to gauge the network load.

The `snoop` utility is also useful for observing the traffic flow between hosts in your SLP enterprise. Using `snoop` with the `slp` command line argument, you can gauge and monitor the following types of SLP activity to determine whether any network or agent reconfiguration is needed:

- The number of hosts using a particular DA. With this information, you can determine whether to deploy additional DAs for load balancing.
- Which hosts are using which DAs. With this information, you can determine whether particular hosts should be configured with new or different scopes.
- UA requests are timing out and need to be retransmitted, or the DA requires more than a few seconds to send registration acknowledgments to an SA. You can determine whether that particular DA is overloaded. Also, you can gauge whether to rebalance the network load on the DA by placing additional DAs, or changing the scope configurations.

Using `snoop` with the `–V` (verbose) command line argument, you can obtain registration lifetimes and value of the fresh flag in `SrvReg` to determine whether the number of reregistrations should be reduced. However, if you invoke `snoop` with the verbose option, the volume of text printed out tends to be large. Therefore, you might dump the text to a file and sort through it later.

You can use `snoop` to trace other kinds of SLP traffic, as well. For example:

- Traffic between UA clients and DAs
- Traffic between multicasting UA clients and replying SAs

For more information about `snoop`, refer to the `snoop`(1M) man page.

Tip - Use the netstat command in conjunction with snoop to view traffic and congestion statistics. For more information about netstat, refer to the netstat(1M) man page.

Regulating Display Output

The snoop command provides various filters and options through which you control the focus of the snoop trace and the length of its output. Additionally, you can use snoop slp with any other snoop expression.

When using the snoop utility, you can select either *brief* or *verbose* mode for the output. In *verbose* mode, snoop delivers ongoing, unabbreviated output to your monitor, which provides the following types of information:

- The complete address for the service URL
- All service attributes
- The registration lifetime
- All security parameters and flags, if any

Type the following command to invoke snoop with the slp filter in verbose mode:

```
# snoop slp -v
```

The default setting for snoop is *brief* mode, which delivers ongoing output to your monitor that is truncated to fit one line per SLP message.

Deciding What to Reconfigure

You can use the SLP-enabled snoop utility and SLP logging utilities to decide what needs reconfiguring. For example, you might want to reconfigure certain properties to:

- Accommodate a mix of network media that have varying latencies and bandwidth characteristics
- Recover the enterprise from network failures or unplanned partitioning
- Add DAs to reduce proliferation of SLP multicasts
- Implement new scopes to organize users with their most frequently accessed services

Analyzing a `snoop slp` Trace

The following sample traces illustrate how to start `snoop` with the SLP filter (in the default *brief* mode), and the kinds of traces it might show. This example is invoked using the following command:

```
# snoop slp
```

In this example, when `snoop slp` is invoked on the host *vesuvius*, `slpd` starts on *vesuvius* and initializes and registers the host *umunum* as an echo server. `slpd` on *vesuvius* runs in the default mode as an SA server.

Note - To make it easier to describe the trace results, the lines have been separated and identified with the trace line number.

Trace 1 shows `slpd` on *vesuvius* performing active directory agent discovery by multicasting to the SLP multicast group address in search of directory agents. The message number, (24487), for the active discovery is indicated in square brackets in the trace display:

```
(1) vesuvius -> 239.255.255.253 SLP V@ SrvRqst [24487] service:directory-agent []
```

Trace 2 illustrates the SLP mechanism in which matching pairs of request-reply messages are identified by their matching message numbers:

```
(2) umunum -> vesuvius SLP V2 DAAdvert [24487] service:directory-agent://129
```

Trace 2 indicates that the active discovery request 24487 from trace 1 is answered by `slpd` running as a DA on the host *umunum*. The service URL from *umunum* has been truncated in the trace to fit on a single line. The DA has sent a DA advertisement in reply to the multicast directory agent discovery message, as indicated by the matching message numbers in traces 1 and 2.

Traces 3 and 4 show multicasts from the UAs on *vesuvius* for additional DAs. Because *umunum* has already answered the request, it refrains from responding again, and no other DAs reply.

```
(3) vesuvius -> 239.255.255.253 SLP V2 SrvRqst [24487] service:directory-agent []
(4) vesuvius -> 239.255.255.253 SLP V2 SrvRqst [24487] service:directory-agent []
```

In the next two traces, `slpd` on *vesuvius* forwards registrations made by SA clients to the DA on the host *umunum*. Note that the traffic between the echo server running the SA client and `slpd` on *vesuvius* does not appear in the `snoop` trace because it is performed over the network loopback. Traces 5 and 6 show `slpd` on *vesuvius* forwarding the service registration to the DA on *umunum*, and the reply from *umunum*.

The message exchange shown in lines 5 and 6 of the trace occurs by TCP, as is indicated by the *tcp* appended to the message number in each case.

```
(5) vesuvius -> umunum SLP V2 SrvReg [24488/tcp]service:echo.sun:tcp://vesuvius:
(6) umunum -> vesuvius SLP V2 SrvAck [24488/tcp] ok
```

Trace 5 shows a unicast service registration (`SrvReg`) for an echo server, which is made by *vesuvius* to the DA on *umunum*. Trace 6 shows *umunum* responding to *vesuvius'* `SrvReg` with a service acknowledgment (`SrvAck`), indicating that the registration is successful.

In the next example, the echo server program on *vesuvius* deregisters the echo service advertisement. `slpd` on *vesuvius* forwards the deregistration to the DA on *umunum*.

```
(1) vesuvius -> umunum SLP V2 SrvDereg [24489/tcp] service:echo.sun:tcp://vesuvius:
(2) umunum -> vesuvius SLP V2 SrvAck [24489/tcp] ok
```

The echo server, *umunum*, informs the host, *vesuvius*, that the deregistration was successful by way of a `SrvAck`. The initial exchange between the echo server SA client and `slpd` does not appear. The reply indicates that the deregistration succeeded.

Where to Go From Here

After monitoring the SLP traffic, you can use the information collected from the `snoop` traces to help determine whether any reconfiguration of the SLP defaults is needed. Use the related information in Chapter 3 and Chapter 4 for configuring SLP property settings. For more information about SLP messaging and service registrations, refer to Appendix A.

CHAPTER **3**

Configuring SLP Properties

This chapter describes the contents of the `/etc/inet/slp.conf` file, where you perform all necessary network configuration property tuning. The SLP configuration properties control network and agent characteristics, agent status, protocol behavior, and SLP logging.

- "SLP Configuration File" on page 25
- "When to Use the Configuration Procedure" on page 26
- "Changing Your Configurations" on page 27

SLP Configuration File

Properties of the SLP agent on a host are configured by editing the SLP configuration file, `/etc/inet/slp.conf`. The following list shows examples of agent properties that you can configure:

- Whether `slpd` acts as a DA or SA server
- Network interaction
- Timing of multicast messages
- Whether SLP logging is enabled for SLP network monitoring

Each time you stop and restart the SLP daemon, the daemon obtains its configuration information from this file.

When to Use the Configuration Procedure

Follow the procedure in this chapter under one or more of the following conditions, if:

- The network medium or topology changes
- You are changing settings to compensate for network latencies
- You are reducing congestion on the network
- You are adding agents, or reassigning IP address
- You are activating SLP logging

SLP Configuration File Elements

The `/etc/inet/slp.conf` file defines and activates all SLP activity each time you restart the SLP daemon. The file contains a list of SLP configuration properties with accompanying comments that describe the effect of changing a property of the SLP configuration. The configuration file consists of the following elements:

- Configuration properties
- Comment lines and notations

Configuration Properties

The list of properties are structured as key-value pairs in the file. The key-value pairs consist of the SLP property names and their settings, in this format:

```
<property name>=<value>
```

The key for each property is the property name. The value sets the numeric (distance or time), true/false state, or string value parameters on the property. Property values consist of one of the following data types:

- True/False setting (Boolean)
- Integers
- List of integers
- Strings
- List of strings

SLP behavior is defined by some property or combination of property settings in the `slp.conf` file. For instance, from within the `slp.conf` file, you can configure `slpd` to operate as a DA, or leave it in its default configuration as an SA server.

If you set the `net.slp.isDA` property to True, `slpd` performs the characteristic activities of a DA.

All of the base SLP properties are named in the following format:

```
net.slp.<keyword>
```

Comment Lines and Notations

These are informational lines in the `slp.conf` file that describe the nature and function of the line. Comment lines are optional in the file, but useful for administration. Settings in the configuration file are case insensitive. Use non-ASCII characters for escaping.

Changing Your Configurations

This section provides a procedure to follow whenever you change any of the configuration property settings.

▼ How to Change Your SLP Configuration

1. **Become superuser.**

2. **Type the following command to stop `slpd` and all SLP activity on the host:**

   ```
   # /etc/init.d/slpd stop
   ```

3. **Back up the default `/etc/inet/slp.conf` file before changing any configuration settings.**

4. **Edit the property settings in the `/etc/inet/slp.conf` file, as needed.**

   Refer to “Configuration Properties” on page 26 for information about the SLP property settings, and the succeeding chapters for the appropriate procedures and examples. Refer to `slp.conf`(4) for more information about the `slp.conf` file.

5. **Save your changes and exit the file.**

6. **Restart `slpd` to activate your changes. Type the following command:**

```
# /etc/init.d/slpd start
```

CHAPTER 4

Configuring Network Properties

SLP includes a collection of configuration properties that control the network interaction characteristics of `slpd` and the client libraries. In most cases, the default configurations of these properties require no change. However, for certain types of network media or network topologies, changes might be required to accommodate varying latencies or bandwidth or to prevent potential problems on your network.

This chapter lists types of scenarios in which you might want to alter the default SLP configuration property settings.

- “Changing Network Configuration Properties” on page 29
- “Modifying DA Advertising and Discovery Frequency” on page 30
- “Accommodating Different Network Media, Topology, or Configuration” on page 33
- “Modifying Timeouts on SLP Discovery Requests” on page 35

Changing Network Configuration Properties

The SLP network properties are designed to control particular aspects of interaction between `slpd`, the client library, and the network. This section lists procedures for altering the following types of SLP networking properties:

- Modifying DA advertising and discovery frequency
- Reducing the interval between SA reregistrations
- Configuring `slpd` and client library networking software for different kinds of networking media and multicast routing configuration

- Modifying the timeouts on SLP discovery requests

In each area, various properties control different aspects of the configuration. The following sections describe how to configure SLP.

Modifying DA Advertising and Discovery Frequency

In some situations, you might need to modify the frequency of DA advertising or DA discovery. DA advertising and discovery are controlled by the following properties:

- `net.slp.DAHeartBeat` - Controls how often a DA multicasts an unsolicited DA advertisement
- `net.slp.passiveDADetection` - Controls whether `slpd` listens for unsolicited DA advertisements
- `net.slp.DAActiveDiscoveryInterval` - Controls how often `slpd` periodically performs active DA discovery for new DAs

The reasons why you might want to modify the default DA advertising and discovery configuration are:

- If you want all SAs and UAs to obtain their DA configuration information statically rather than through discovery, you can configure the properties to disable DA discovery.
- If your network is subject to frequent partitions, or UA and SA clients are accessing a DA over a dial-up network, you can configure the frequency of passive advertisements and periodic active discovery to either increase or reduce the period.
- If network congestion is high, you might need to limit multicast.

Limiting UAs and SAs to Statically Configured DAs

In some cases, it might be necessary to limit UAs and SAs to obtain only their DA addresses from the static configuration information in the `slp.conf` file. You can accomplish this by disabling passive and active DA discovery in `slpd`.

Disable passive discovery by setting the `net.slp.passiveDADetection` property to `False`. This causes `slpd` to ignore unsolicited DA advertisements.

Disable initial and periodic active discovery by setting the `net.slp.DAActiveDiscoveryInterval` to `-1`. This causes `slpd` to refrain from performing initial active DA discovery, and from periodically polling for new DAs.

Perform these configurations only on those hosts executing the UAs and SAs that are restricted to using their static configurations. As a consequence of these configuration actions, `slpd` obtains DA information only from the `net.slp.DAAddresses` property in the `slp.conf` file.

Modifying DA Advertising and Discovery Timing

For certain combinations of network media and the overall network topology, you might need to modify the timing of passive DA advertisements and periodic active discovery requests. This section provides these examples:

- If your network is subject to frequent partitions, and SAs tend to remain in operation for long time intervals, you can increase the frequency of the passive DA advertisement frequency.
- If you have a DA on one side of a dial-up connection and UAs or SAs that use the DA on the other side, you can decrease both the DA heartbeat frequency and the active discovery frequency, to decrease the number of times the dial-up line is activated.

Configuring the DA Heartbeat for Frequent Partitions

SAs are required to register with all DAs that support their scopes. A DA can appear after `slpd` has performed active discovery. If the DA supports `slpd` scopes, then `slpd` registers all advertisements on its host with the DA, in order for the set of service advertisements in the scopes to be consistent.

One way `slpd` discovers DAs is by the initial unsolicited advertisement a DA sends out when it boots. `slpd` uses the periodic unsolicited advertisement (the heartbeat) to determine whether a DA is still active, and removes the DAs it uses and offers to UAs if a heartbeat fails to appear.

Finally, when a DA undergoes a controlled shutdown, it transmits a special DA advertisement that informs listening SA services that it will be out of service. `slpd` also uses this advertisement to remove inactive DAs from the cache.

If your network is subject to frequent partitions and SAs are long-lived, `slpd` can remove DAs from its cache during the partitions if heartbeat advertisements are not received. By decreasing the heartbeat time, you can decrease the delay before a deactivated DA is restored to the cache after the partition is repaired.

The `net.slp.DAHeartBeat` property controls how often a DA multicasts an unsolicited DA advertisement. By default, the DA heartbeat period is set to 3 hours (10800 seconds). The following entry decreases the DA heartbeat value to one hour:

```
net.slp.DAHeartBeat=3600
```

Configuring DA Discovery for Dial-up Networks

If the UAs or SAs are separated from the DA by a dial-up network, you can configure DA discovery to reduce or eliminate the number of discovery requests and DA advertisements. Dial-up networks usually incur a charge when they are activated, so minimizing extraneous calls can reduce the cost of using the dial-up network.

You can disable DA discovery completely using the method described in "Limiting UAs and SAs to Statically Configured DAs" on page 30. If DA discovery is completely disabled, the `net.slp.DAAddresses` property must be set in `slp.conf` on the hosts executing UAs and SAs so that they access the correct DA.

You can reduce unsolicited DA advertisements and active discovery by increasing the DA heartbeat period and the active discovery period. In the following example, both the active discovery interval and the DA heartbeat are set to about 18 hours. The first line shows the setting for the host executing the DA:

```
net.slp.DAHeartBeat=65535
```

On the hosts executing the UAs and SAs, the active discovery interval is configured as:

```
net.slp.DAActiveDiscoveryInterval=65535
```

Both property values are measured in seconds.

Network Congestion Is High

If network congestion is high, you might want to limit the amount of multicast. If DAs have not already been deployed in the network, deploying DAs can drastically cut back on the amount of SLP-related multicast.

However, even after DAs are deployed, multicast is still necessary for DA discovery. You can reduce the amount of multicast necessary for DA discovery using the method described in "Configuring DA Discovery for Dial-up Networks" on page 32. You can completely eliminate multicast for DA discovery using the method described in "Limiting UAs and SAs to Statically Configured DAs" on page 30.

Accommodating Different Network Media, Topology, or Configuration

Depending on the underlying network media and configuration in your network, you can tune SLP performance by modifying one of the following parameters:

- `net.slp.DAAttributes` - Configures the `minimum refresh interval` attribute to specify the minimum refresh interval the DA will accept for advertisements.
- `net.slp.multicastTTL` - Configures the time to live of multicast packets.
- `net.slp.MTU` - Configures the maximum UDP and multicast/broadcast packet size.
- `net.slp.isBroadcastOnly` - Configures broadcast or multicast for non-DA-based service discovery and DA discovery.

This section describes how to use these properties to tune SLP performance.

Reducing the Interval Between SA Reregistrations

SAs need to periodically refresh their service advertisements before the lifetime expires. If a DA is handling an extremely heavy load from many UAs and SAs, frequent refreshes can cause the DA to become overloaded. If the DA becomes overloaded, UA requests start timing out and are dropped. Timing out of UA requests has many possible causes, so before assuming that DA overloading is the problem, you should use a `snoop` trace to check the lifetimes of service advertisements registered with a service registration. If the lifetimes are short and reregistrations are occurring often, then timeouts are probably due to too-frequent reregistrations. A service registration is a reregistration if the fresh flag is not set. See Appendix A for more information on service registration messages.

The following example sets the `minimum refresh interval` for SAs to one hour:

```
net.slp.DAAttributes=(min-refresh-interval=3600)
```

The `net.slp.DAAttributes` property has units of seconds. The default minimum reregistration period is zero (0), so that an SA is free to reregister whenever it chooses.

Configuring the Multicast Time to Live Property

The multicast time to live (`net.slp.multicastTTL` property) determines the range over which a multicast packet is propagated in your intranet. The multicast TTL is configured by setting the `net.slp.multicastTTL` property to an integer between 1 and 255. The default value of the multicast TTL is 255, which means, theoretically, that the packet routing is unrestricted. However, a TTL of 255 causes a multicast packet to penetrate the intranet to the border routers on the edge of your administrative domain. Correct configuration of multicast on border routers is required to prevent multicast packets from leaking into the Internet's multicast backbone, or to your ISP.

Multicast TTL scoping is similar to standard IP TTL, with the exception that a TTL comparison is made. Each interface on a router that is multicast-enabled is assigned a TTL value. When a multicast packet arrives, the router compares the TTL of the packet with the TTL of the interface. If the TTL of the packet is greater than or equal to the TTL of the interface, the packet TTL is reduced by one, as is the case with the standard IP TTL. If the TTL becomes zero, the packet is discarded. Therefore, using TTL scoping for SLP multicast requires that your routers be properly configured so that packets are limited to a particular subsection of your intranet.

You can reduce the range of multicast propagation by reducing the TTL value. If the TTL value is 1, then the packet is restricted to the subnet. If it is 32, the packet is restricted to the site. Unfortunately, the term *site* is not defined by RFC 1075, where multicast TTLs are discussed. Values above 32 refer to theoretical routing on the Internet, and should not be used, but values below 32 can be used to restrict multicast to a set of accessible subnets, provided the routers are properly configured with TTLs.

Configuring the Packet Size

The default packet size for SLP is 1400 bytes, and this should be sufficient for most local area networks. For wireless networks or wide area networks, you can reduce the packet size to avoid message fragmentation and reduce network traffic. For local area networks that have larger packets, increasing the packet size can improve performance. You can determine whether the packet size needs to be reduced by checking the minimum packet size for your network. If the network medium has a smaller packet size, setting the `net.slp.MTU` property is necessary.

You can also increase the packet size if your network medium has larger packets. However, unless the service advertisements from SAs or queries from UAs are frequently overflowing the default packet size, configuration should not be necessary. You can determine whether the default packet size is overflowing frequently by using `snoop` to check whether UA requests are overflowing often, and rolling over to use TCP instead of UDP.

The `net.slp.MTU` property measures the complete IP packet size, including the link layer header, the IP header, and the UDP or TCP header, as well as the SLP message.

Configuring Broadcast Instead of Multicast

SLP is designed to use multicast for service discovery in the absence of DAs and for DA discovery. For various reasons, some networks do not deploy multicast routing. If your network does not deploy multicast routing, you can configure SLP to use broadcast by setting the `net.slp.isBroadcastOnly` property to `True`.

Unlike multicast, broadcast packets do not propagate across subnets by default. For this reason, service discovery without DAs in a non-multicast network works only on a single subnet. In addition, special considerations are required when deploying DAs and scopes in networks where broadcast is used. A DA on a multihomed host can bridge service discovery between multiple subnets with multicast disabled. See Chapter 8 for more information on deploying DAs on multihomed hosts.

Modifying Timeouts on SLP Discovery Requests

There are two general situations where timeouts on SLP discovery requests might need to be modified:

- If the SLP agents are separated by multiple subnets, dial-up lines, or other WANs, the network latency can be high enough that the default timeouts are insufficient for a request or registration to complete. Conversely, if your network is low latency, performance can be improved by decreasing the timeouts.
- In a network with much traffic or a high collision rate, the maximum period that SAs and UAs need to wait before sending a message might be insufficient to assure collision-free transactions.

Changing Default Timeouts

High network latency can cause UAs and SAs to time out in making their requests and registrations before a response returns. Latency can be a problem if a UA is separated from an SA, or both a UA and SA are separated from a DA, either by multiple subnets, a dial-up line, or a WAN. You can determine whether latency is a problem by checking whether UAs and SAs are timing out on requests and registrations and their SLP requests are failing, or by using `ping` to measure the actual latency.

The sets of configuration properties that control timeouts:

- The `net.slp.multicastTimeouts`, `net.slp.DADiscoveryTimeouts`, and `net.slp.datagramTimeouts`. These properties control the list of timeouts used for repeated multicast and unicast UDP message transmissions, before a message transmission is abandoned.

- The `net.slp.multicastMaximumWait`. This property controls the maximum amount of time a multicast message is retransmitted before it is abandoned.

If frequent timeouts are occurring during multicast service discovery or DA discovery, the `net.slp.multicastMaximumWait` property should be increased from its default value of 15000 milliseconds (15 seconds). Increasing the maximum wait period allows more time for requests on high latency networks to complete. After this property is increased, you should also modify the `net.slp.multicastTimeouts` and `net.slp.DADiscoveryTimeouts` lists so that the sum of the integer timeout values in the lists equals the `net.slp.multicastMaximumWait` value.

As an example, suppose you have determined that multicast requests require 20 seconds (20000 milliseconds) instead of 15 seconds to complete. The following property configurations change the maximum wait time and the list of multicast timeouts:

```
net.slp.multicastMaximumWait=20000
net.slp.multicastTimeouts=2000,5000,6000,7000
net.slp.DADiscoveryTimeouts=3000,3000,6000,8000
```

If your network has low latency, you can reduce the `net.slp.multicastMaximumWait` configuration value and the values in the two multicast timeout lists to reduce waiting time.

A similar procedure works for the timeouts involved in sending a UDP datagram to a DA, except the upper bound on the DA timeout is determined by the sum of the elements in the `net.slp.datagramTimeouts` list and not a separate property. A UDP datagram is repeatedly sent to a DA until a response is received or the timeout bound is reached. If frequent timeouts occur, the values in the list can be increased.

For example, suppose, as above, you need to increase the datagram timeout bound to 20000 milliseconds to avoid frequent timeouts. The following configuration will achieve that:

```
net.slp.datagramTimeouts=2000,5000,6000,7000
```

In high performance networks, you can perform this procedure in reverse to reduce the timeout bound for multicast and unicast UDP datagram transmission. This will reduce the amount of latency in satisfying SLP requests.

Configuring the Random Wait Bound

In networks with heavy traffic or a high collision rate, communication with a DA might be affected. When collision rates are high, the sending agent must retransmit the UDP datagram. You can determine if retransmission is occurring by using `snoop` to monitor traffic in a network of hosts running `slpd` as an SA server and a host running `slpd` as a DA. If multiple service registration messages for the same service from the host running `slpd` as an SA server are appearing in the `snoop` trace, then you might have a problem with collisions.

Collisions can be a particular problem at boot time. When a DA is first coming up, it sends out unsolicited advertisements and the SAs respond with their registrations. SLP requires the SAs to wait for a random amount of time after receiving a DA advertisement before responding. The random wait bound is uniformly distributed with a maximum value controlled by the `net.slp.randomWaitBound`. The default random wait bound is 1000 milliseconds (1 second).

You can lengthen the maximum wait by configuring the property. For example:

```
net.slp.randomWaitBound=5000
```

This example lengthens the random wait to 5000 milliseconds (5 seconds).

Lengthening the random wait bound causes longer delay in registration with the DA, so SAs can complete registrations with newly discovered DAs more slowly, thereby avoiding collisions and timeouts.

CHAPTER **5**

Configuring Scopes

SLP enables you to provision services that depend on logical, physical, or administrative grouping of users by creating scopes to administer their access to service advertisements.

Deploying Scopes

Use the `net.slp.useScopes` property to create scopes. For example, in the `/etc/inet/slp.conf` file on a host, add a new scope, called `newscope`, as shown:

```
net.slp.useScopes=newscope
```

To help you understand the scope concept, imagine that your organization has an alcove of networked office machines, such as printers and fax machines, and that they reside at the end of the south hall on the second floor of Building 6. These office machines might be provided for all users on the second floor, or their use might be restricted to members of a certain department. Scopes provide a way to provision access to the service advertisements for these machines.

If the office machines are dedicated to the marketing department, then the marketing department hosts and the printers in proximity to that cluster can be configured with a scope named `mktg`. Other printers, belonging to other departments, can be configured with different scope names.

In another scenario, the departments might be dispersed. For instance, the mechanical engineering and the CAD/CAM departments might be split between floors 1 and 2. However, you can provide the floor 2 machines for the hosts on both floors by assigning them to the same scope. You can deploy scopes in any manner that operates well with your network and users.

Note - UAs having a particular scope are not prevented from actually using services advertised in other scopes. Configuring scopes controls only which service advertisements a UA sees. It is up to the service itself to enforce any access control restrictions.

When to Configure Scopes

SLP can function adequately without any scope configuration whatsoever. In the Solaris operating environment, the default scope for SLP is `default`. If no scopes are configured, `default` is the scope of all SLP messages.

Configuration of scopes is suggested if any of the following apply:

1. The organizations you support want to restrict service advertisement access to their own members.
2. The physical layout of the organization you support suggests that services in a certain area be accessed by particular users.
3. Some other reason exists for partitioning the service advertisements that users are allowed to see.

An example of the first case was cited in "Deploying Scopes" on page 39. An example of the second case is a situation in which an organization is spread between two buildings, and you want users in a building to access local services in that building. Users in Building 1 can be configured with the `B1` scope, while users in Building 2 can be configured with the `B2` scope.

Configuring Scopes

Scopes are configured by setting the `net.slp.useScopes` property in the `slp.conf` file to a comma-separated list of scope names. Construct scope names using the following grammatical guidelines:

- Any alphanumeric characters, upper or lower-case
- Any punctuation characters (except for: ’’, \, *!*, <, =, >, and ~).

- Spaces that are considered part of the name
- Non-ASCII characters

  You use a backslash to escape non-ASCII characters. For example, UTF-8 encoding uses `0xc3a9` hex code to represent the letter *e* with the French *aigue* accent. If the platform does not support UTF-8, you use the UTF-8 hex code as the escape sequence `\c3\a9`.

▼ How to Configure Scopes

This procedures assumes (as an example) that you are creating scopes `eng` and `mktg` in `bldg6`. In this case, you are configuring multiple scopes.

1. **Become superuser.**

2. **Edit the** `/etc/inet/slp.conf` **file and change the** `net.slp.useScopes` **line to:**

```
net.slp.useScopes=eng,mktg,bldg6
```

3. **Save the file and exit.**

Considerations When Configuring Scopes

By modifying the `net.slp.useScopes` property in the `slpd.conf` file, you configure scopes for all agents on the host. If the host is running any SAs or is acting as a DA, you must configure this property if you want to configure the SAs or DA into scopes other than `default`. If only UAs are running on the machine and the UAs should discover SAs and DAs supporting scopes other than `default`, you do not need to configure the property unless you want to restrict the scopes the UAs use. If the property is not configured, UAs will automatically discover available DAs and scopes through `slpd`, which uses active and passive DA discovery to find DAs, or through SA discovery if no DAs are running. On the other hand, if the property is configured, UAs will use only the configured scopes and not discard them.

If you decide to configure scopes, you should consider keeping the `default` scope on the list of configured scopes unless you are sure that all SAs in the network have scopes configured. If any SAs are left unconfigured, UAs with configured scopes will be unable to find them, because the unconfigured SAs automatically have scope `default`, but the UAs have the configured scopes.

If you also decide to configure DAs by setting the `net.slp.DAAddresses` property, be sure that the scopes supported by the configured DAs are the same as the scopes that you have configured with the `net.slp.useScopes` property. If this is not the case, `slpd` prints an error message when it is restarted.

CHAPTER **6**

Deploying DAs

This chapter describes the strategic deployment of DAs in a network running SLP.

SLP functions adequately with only the base agents, UA and SA, and without any deployed DAs or configured scopes (all agents automatically have the `default` scope). However, DAs serve as caches for service advertisements, and they are useful for reducing multicast. This capability enables SLP to accommodate larger networks.

Why Deploy an SLP DA?

The primary reason to deploy DAs is to reduce the amount of multicast traffic involved in service discovery. In a large network with many UAs and SAs, the amount of multicast traffic involved in service discovery can become so large that network performance degrades. By deploying one or more DAs, UAs must unicast to DAs for service and SAs must register with DAs using unicast. The only SLP-registered multicast in a network with DAs is for active and passive DA discovery.

SAs register automatically with any DAs they discover within a set of common scopes, rather than taking multicast service requests. Multicast requests in scopes that are not supported by the DA are still answered directly by the SA, however.

Service requests from UAs are unicast to DAs rather than multicast onto the network when a DA is deployed within the UA's scopes. Consequently, DAs within the UA's scopes reduce multicast. By eliminating multicast for normal UA requests, delays and timeouts are eliminated.

DAs act as a focal point for SA and UA activity. Deploying one or several DAs for a collection of scopes provides a centralized point for monitoring SLP activity. By turning on DA logging, it is easier to monitor registrations and requests than to check the logs from multiple SAs scattered around the network. You can deploy any number of DAs for a particular scope or scopes, depending on the need to balance the load.

In networks without multicast routing enabled, you can configure SLP to use broadcast. However, broadcast is very inefficient, because it requires each host to process the message. Broadcast also does not normally propagate across routers. As a result, in a network without multicast, DAs can be deployed on multihomed hosts to bridge SLP advertisements between the subnets. See Chapter 4 for more information on how to deploy SLP on networks without multicast enabled.

Finally, the Solaris SLPv2 DA supports interoperability with SLPv1. SLPv1 interoperability is enabled by default in the Solaris DA. If your network contains SLPv1 devices, like printers, or you need to interoperate with Novell Netware 5, which uses SLPv1 for service discovery, you should deploy a DA. Without a DA, the Solaris SLP UAs are unable to find SLPv1 advertised services.

When to Deploy DAs

Deploy DAs on your enterprise if any of the following conditions are true:

- Multicast SLP traffic exceeds 1% of the bandwidth on your network, as measured by `snoop`.
- UA clients experience long delays or timeouts during multicast service requests.
- You would like to centralize monitoring of SLP service advertisements for particular scopes on one or several hosts.
- Your network does not have multicast enabled and consists of multiple subnets that must share services.
- Your network employs devices that support earlier versions of SLP (SLPv1) or you would like SLP service discovery to interoperate with Novell Netware 5.

Deploying DAs

In Solaris SLP, you deploy a DA by changing the default setting for the `net.slp.isDA` property in the host's `slp.conf` file, then stopping and restarting `slpd`. This causes `slpd` to start as a DA. You can assign only one DA per host.

▼ How to Deploy a DA

1. **Become superuser.**

2. **Edit the `/etc/inet/slp.conf` file and set the `net.slp.isDA` property to true.**

   ```
   net.slp.isDA=True
   ```

3. **Save the file and exit.**

4. **Restart `slpd` to deploy it as a DA.**

   ```
   # /etc/init.d/slpd start
   ```

Where to Place DAs

This section provides suggestions for where to place DAs in different situations.

1. When multicast routing is not enabled and DAs are required to bridge service discovery between subnets

   In this case, a DA must be placed on a host with interfaces and all subnets that share services. The `net.slp.interfaces` configuration property does *not* need to be set, unless IP packets are not routed among the interfaces. See Chapter 8 for more information on configuring the `net.slp.interfaces` property.

2. When DAs are deployed for scalability, the primary consideration is optimizing agent access

   UAs typically make many requests for services to DAs. An SA registers with the DA once, and can refresh the advertisement at periodic, but not frequent, intervals. As a result, UA access to DAs is far more frequent than SA access. The number of service advertisements is also usually smaller than the number of requests. Consequently, most DA deployments are more efficient if the deployment is optimized for UA access.

3. Situating DAs so that they are topologically close to UAs on the network to optimize UA access

   Placing UAs topologically close to their DAs reduces the amount of routing delay for answering SLP requests. Naturally, you must configure the DA with a scope shared by both the UA and SA clients.

Placing Multiple DAs for Load Balancing

You can deploy multiple DAs for the same collection of scopes as a means of load balancing. Deployment of multiple DAs is suggested in any of the following circumstances:

- UA requests to a DA are timing out, or are returning with the `DA_BUSY_NOW` error.
- The DA log shows that many SLP requests are being dropped.
- The network of users sharing services in the scopes spans a number of buildings or physical sites.

You can do a `snoop` trace of SLP traffic to determine how many UA requests return with the `DA_BUSY_NOW` error. If the number of UA requests returned is high, which is likely if a single DA is deployed for all users, UAs in buildings physically and topologically distant from the DA can exhibit slow response or excessive timeouts. You might want to deploy a DA in each building to improve response for UA clients within the building.

Links connecting buildings are often slower than the local area networks within the buildings. If your network spans multiple buildings or physical sites, set the `net.slp.DAAddresses` property in the `/etc/inet/slp.conf` file to a list of specific host names or addresses so that the UAs access only the DAs you specify.

If a particular DA is using large amounts of host memory for service registrations, reduce the number of SA registrations by reducing the number of scopes the DA supports. You can split a scope having many registrations into two and support one of the new scopes by deploying another DA on another host.

CHAPTER 7

Incorporating Legacy Services

Servers that can use SLP to advertise contain an internal SA that advertises their service and attributes to either the UA directly, or to the DA, if any. Legacy services are network services that pre-date the development and implementation of SLP. For example, Solaris services, such as the line printer daemon (`lpsched`), NFS file service, and NIS/NIS+ name service, do not contain internal SAs for SLP. This chapter describes when and how to advertise legacy services.

- "When to Advertise Legacy Services" on page 47
- "How to Advertise Legacy Services" on page 47
- "Considerations When Advertising Legacy Services" on page 51

When to Advertise Legacy Services

Legacy service advertising might be required if applications with SLP UAs are deployed in your network and one of the following is true:

- Hardware devices without SLP built in, such as printers, are available on your network; applications with SLP UAs need to be able to find them.
- Software services without SLP built in, such as databases, are available on your network; applications with SLP UAs need to be able to find them.

How to Advertise Legacy Services

You can advertise legacy services in the following ways:

- Modify the service to incorporate an SLP SA.
- Write a small program that functions as an SLP SA to advertise the service.
- Use proxy advertising to have `slpd` advertise the service.

Modifying the Service

If the service is a software server and the source code is available, you can modify the source code to incorporate an SLP SA into the server. The C and Java APIs for SLP are relatively straightforward to use (see the man pages for the C API and Javadoc pages for the Java API). If the service is a hardware device, the manufacturer might have an updated PROM that incorporates SLP. Contact the device manufacturer for more information.

Writing an SLP SA to Advertise the Service

If the source code or an updated PROM with SLP is not available, you can write a small SLP SA that advertises the service. The SA can function as a small daemon that is started from the same shell script as the service itself. Similarly, the service control shell script can also stop the SA. Again, the SLP APIs for C and Java provide programmatic access to SLP.

Using SLP Proxy Registration

Solaris `slpd` supports advertising of legacy services using a proxy registration file. A proxy registration file is a list of service advertisements in a portable format.

A service advertisement consists of a service URL line, an optional scope line, and a series of attribute definitions. `slpd` reads proxy advertisements from the file, registers the advertisements, and maintains them exactly like an SA client would. The following listing shows an example of a proxy registration file. Line numbers have been added for description purposes and are not part of the file.

```
1  #Advertise legacy printer.
2
3  service:lpr://bizserver/mainspool,en,65535
4  scope=eng,corp
5  make-model=Laserwriter II
6  location-description=B16-2345
7  color-supported=monochromatic
8  fonts-supported=Courier,Times,Helvetica 9 10
```

(continued)

(Continuation)

```
9
10 #Advertise FTP server
11
12 ftp://archive/usr/src/public,en,65535,src-server
13 content=Source code for projects
14
```

In this example, a legacy printer supporting the LPR protocol and an FTP server are being advertised.

Lines 1 and 10	Comment lines begin with a cross-hatch symbol (#) and do not affect the file's operation. Everything up through the end of a comment line is ignored.
Blank lines at 2, 9, and 14	Delimit the advertisements
Line 3 and 12	Service URLs
Line 4	Scope designation
Lines 5-8 and 13	Attribute definitions

Service URL

A service URL line has three required fields and one optional field, separated by commas:

- **First field** – This field contains the service URL to be advertised. The service URL can be either a generic URL or a service: URL. See RFC 2609 for the specification of how to form a service: URL.
- **Second field** – This field designates the language of the advertisement. In the previous example, the field designated English, *en*. The language is an RFC 1766 language tag.
- **Third field** – This field establishes the lifetime of the registration, measured in seconds. The lifetime is restricted to an unsigned 16 bit-integer. If the lifetime is less than the maximum, 65535, `slpd` times out the advertisement like any other advertisement. If the lifetime is 65535, `slpd` refreshes the advertisement periodically, and the lifetime is considered permanent, until `slpd` exits.

Note - As a practical matter, nonpermanent proxy registrations are of limited usefulness, because once `slpd` removes the advertisement, it is no longer accessible to UAs. Both advertisements in the example are permanent.

- **Service type field (optional)** – If used, this field defines the service type. If the service URL is a generic URL, then it is possible to change the service type under which the URL is advertised. In the previous example of a proxy registration file, line 12 contains a generic FTP URL, and the optional type field causes it to be advertised under the service type name *src-server*. The `service:` prefix is not added by default to the type name.

Scope Designation

After the URL line, an optional scope line can appear. The scope line consists of the token `scope` followed by an equal sign, followed by a comma-separated list of scope names defined by the `net.slp.useScopes` configuration property. Only scopes that are part of the list of configured scopes for the host should appear in the scope list. If the scope line does not appear, the registration is made in all scopes with which `slpd` is configured. The scope line must appear immediately after the URL line; otherwise, it is recognized as an attribute.

Attribute Definitions

After the optional scope line, the bulk of the service advertisement contains attribute/value list pair lines. Each pair consists of the attribute tag, followed by an equal sign, followed by a comma-separated list of attribute values, or a single value if the attribute is single valued. In the previous example of a proxy registration file, line 8 illustrates a value list with multiple values; all other value lists have single values. The format for the attribute names and values is the same as on-the-wire SLP messages.

For a description of the format see RFC 2608. The proxy registration file supports the same convention for escaping non-ASCII characters as the configuration file. For more information about the proxy registration file format, see RFC 2614.

Enabling Proxy Registration

The first step in enabling proxy registration is to create a proxy registration file. You can locate the file anywhere on the host's file system, or anywhere else on the network that is accessible by HTTP. Use the `net.slp.serializedRegURL` property in the `slp.conf` file in the `/etc/inet` directory to locate it (see `slp.conf`(4)).

When creating proxy registrations, you should check whether a service type template exists for the service. A service type template is a description of the service URL and attributes for a service type. Templates enable UAs and SAs to interoperate, because they define what constitutes a service advertisement for a particular service type. If a service type template exists, you should use the template to construct the proxy registration. See RFC 2609 for more information on service type templates.

If a service type template is not available for the service, then you should select a collection of attributes that precisely describe the service. In addition, you should use a naming authority, other than the default, for the advertisement, because the default naming authority is only allowed for service types that have been standardized. See RFC 2609 for more information on naming authorities.

For example, suppose a company called *BizApp* has a local software defect database that they would like to advertise. They make a URL with the service type `service:bugdb.bizapp` and advertise that. In this case, the naming authority is `bizapp`.

When the proxy registration file is complete, you need to configure `slpd` to read it. Edit the `slp.conf` file and add the following line:

```
net.slp.serializedRegURL=<proxy registration file URL>
```

You do not add the file name here. Instead, you add a URL giving the location of the file. For example, if the serialized registration file is named `slp.reg`, and it is in the local `/net/inet` directory, you configure the following the URL property:

```
net.slp.serializedRegURL=file:/etc/inet/slp.reg
```

Considerations When Advertising Legacy Services

Generally, modifying the source code to add SLP is preferable to either writing a stand-alone SA or using proxy registration. By modifying the source code, it is possible to add service-specific features and to closely track service availability. If the source code is unavailable, writing a stand-alone SA is preferable to using proxy registration. A stand-alone SA allows the SA to track service availability by being integrated into the service start/stop procedure. Proxy advertising is generally the third choice, when no source code is available and writing a stand-alone SA is impractical.

Proxy advertisements are maintained only as long as `slpd` that reads the proxy registration file is running. If an advertisement times out or `slpd` is halted, the proxy advertisement is no longer available. Conversely, there is no direct connection between the proxy advertisement and the service itself.

If the service is brought down, `slpd` must be brought down, the serialized registration file edited to comment out or remove the proxy advertisement, and `slpd` brought up again. You must follow the same procedure when the service is restarted or reinstalled. The lack of connection between the proxy advertisement and the service is a major drawback of proxy advertisements, in comparison with the two other techniques of legacy advertising.

CHAPTER **8**

Configuring SLP on Multihomed Hosts

Server machines can sometimes have more than one network interface card, if the server is acting as a host on multiple IP subnets. In the majority of cases, such multihomed hosts also act as routers; IP packets, including multicast packets, are routed between the interfaces. In some cases, however, routing between interfaces might be disabled. This chapter describes how to configure SLP for those cases.

- “Overview” on page 53
- “When to Configure for Nonrouted, Multiple Network Interfaces” on page 54
- “How to Configure for Nonrouted, Multiple Network Interfaces” on page 54
- “Considerations When Configuring for Nonrouted, Multiple Network Interfaces” on page 57

Overview

Without configuration, `slpd` listens for multicast and for UDP/TCP unicast on the default network interface. If unicast and multicast routing is turned on between interfaces on a multihomed machine, no additional configuration is needed, because multicast packets that arrive at another interface are properly routed to the default. As a result, multicast requests for DA or other service advertisements arrive at `slpd`. If routing is not turned on for some reason, then configuration is required.

When to Configure for Nonrouted, Multiple Network Interfaces

The most likely cases where configuration might be required on multihomed machines are:

- Unicast routing is enabled between the interfaces and multicast routing is disabled.
- Unicast routing and multicast routing are both disabled between the interfaces.

When multicast routing is disabled between interfaces, it is usually because multicast has not been deployed in the network. In that case, broadcast is normally used for nonDA-based service discovery and DA discovery on the individual subnets. Broadcast is configured by setting the `net.slp.isBroadcastOnly` property to true.

How to Configure for Nonrouted, Multiple Network Interfaces

The most likely configuration steps that might be required on multihomed machines with routing disabled are:

- Configuring the `net.slp.interfaces` property
- Arranging proxy service advertisements so that UAs on subnets get service URLs with reachable addresses or host names
- Placing DAs and configuring scopes to assure reachability between UAs and SAs on the subnets

Configuring the `net.slp.interfaces` Property

If the `net.slp.interfaces` property is set, `slpd` listens for unicast and multicast/ broadcast SLP requests on the interfaces listed in the property, rather than on the default interface. For example, suppose a server with three network cards and multicast routing turned off is connected to three subnets. And suppose the three network interfaces have IP addresses `192.147.142.42`, `192.147.143.42`, and `192.147.144.42` and that the subnet mask is `255.255.255.0`. The following property setting causes `slpd` to listen on all three interfaces for unicast and multicast/broadcast messaging:

```
net.slp.interfaces=192.147.142.42,192.147.143.42,192.147.144.42
```

You can set the `net.slp.interfaces` property to either IP addresses in dotted decimal form, as above, or to resolvable host names.

Usually, you set the `net.slp.interfaces` property in conjunction with enabling broadcast by setting the `net.slp.isBroadcastOnly` property, because multicast has not been deployed in the network. However, if multicast has been deployed, but is not being routed on this particular multihomed host, it is possible for a multicast request to arrive at `slpd` from more than one interface. This can happen because another multihomed host or router connecting the subnets served by the interfaces routes the packets.

When this happens, the SA server or UA sending the request gets two responses from `slpd` on the multihomed host. These responses are filtered out by the client libraries, so the client should not see them. They are visible on the `snoop` trace, however.

Be careful when configuring the `net.slp.interfaces` property if unicast routing is turned off. Without unicast routing, any services advertised by SA clients on the multihomed host might not be reachable from all the subnets, unless the SA clients do the following:

- Advertise one service URL for each individual subnet
- Assure that requests from a particular subnet are answered with a reachable URL

The SA client library makes no effort to assure that reachable URLs are advertised. Therefore, it is up to the service program, which might or might not handle a multihomed host with no routing, to assure that reachable URLs are advertised.

Before deploying a service on a multihomed host with unicast routing disabled, you should use `snoop` to determine whether the service handles requests from multiple subnets correctly. Furthermore, if you are planning on deploying a DA on the multihomed host, see "DA Placement and Scope Name Assignment" on page 56.

Proxy Advertising on Multihomed Hosts

If a host with multiple interfaces advertises services using `slpd` and proxy registration, the service URLs advertised by `slpd` must contain reachable host names or addresses. If unicast routing is enabled between the interfaces, then hosts on all subnets are able to reach hosts on the others, and proxy registrations can be made for a service on any subnet. If, however, unicast routing is disabled, then service clients on one subnet cannot reach services on another through the multihomed host (although they might be able to reach them through another router).

For example, suppose the host with default host name `bigguy` has three interface cards on three different unrouted subnets. The host names on these subnets are `bigguy`, with IP address `192.147.142.42`, `bigguy1`, with IP address `192.147.143.42`, and `bigguy2`, with IP address `192.147.144.42`. Now suppose that a legacy printer, `oldprinter`, is connected to the `143` subnet. If the URL:

```
service:printing:lpr://oldprinter/queue1
```

is proxy-advertised on all interfaces by configuring `net.slp.interfaces` to listen on all interfaces, machines on the `142` and `144` subnets will receive the URL in response to service requests, though they will be unable to access the service.

The solution to this problem is to perform the proxy advertisement with `slpd` running on a machine connected the `143` subnet only, rather than on the multihomed host. Only hosts on the `143` subnet are able to obtain the advertisement in response to a service request.

DA Placement and Scope Name Assignment

The placement of DAs and assignment of scope names in a network with a multihomed host, in which routing is disabled but the `net.slp.interfaces` property is configured, must be done carefully to assure that clients obtain accessible services. Again, if unicast routing is enabled between the interfaces on a multihomed machine, no special DA and scope configuration is necessary, because advertisements cached with the DA identify services that are accessible from any of the subnets. However, if unicast routing is disabled, poor placement of DAs can result in problems.

To see what can go wrong in the previous example, consider what would happen if `bigguy` runs a DA, and clients on all subnets have the same scopes. SAs on the `143` subnet register their service advertisements with the DA. UAs on the `144` subnet can obtain those service advertisements, even though hosts on the `143` subnet are unreachable.

One solution to this problem is to run a DA on each subnet and not on the multihomed host. In this case, the `net.slp.interfaces` property on the multihomed hosts should be configured with a single interface host name or address, or it should be left unconfigured, forcing the default interface to be used. A drawback of this solution is that multihomed hosts are often large machines that could better handle the computational load of a DA.

Another solution is to run a DA on the multihomed host, but configure scopes such that the SAs and UAs on each subnet have a different scope. For example, in the above case, scopes could be configured so that UAs and SAs on the `142` net have scope `scope142`, UAs and SAs on the `143` net have scope `scope143`, and UAs and SAs on the `144` net have scope `scope144`. You can configure the `net.slp.interfaces` property on `bigguy` with the three interfaces, so that the DA serves three scopes on the three subnets.

Considerations When Configuring for Nonrouted, Multiple Network Interfaces

If multicast routing is turned off in the network, but unicast routing between interfaces on a multihomed host is enabled, configuring the `net.slp.interfaces` property enables a DA on the multihomed host to bridge service advertisements between the subnets. Because unicast is routed between the interfaces, hosts on a subnet different from the subnet on which the service is located can contact the service when they receive the service URL. Without the DA, SA servers on a particular subnet receive only broadcasts made on the same subnet, so they cannot locate services off of their subnet.

The most common case requiring configuration of `net.slp.interfaces` is when multicast is not deployed in the network and broadcast is used instead. Other cases require careful thought and planning to avoid unnecessary duplicate responses or unreachable services.

APPENDIX **A**

SLP Message Types

This appendix describes the SLP message types and includes their abbreviation and function code.

SLP Message Types

TABLE A–1 SLP Message Types

Message Type	**Abbreviation**	**Function Code**	**Description**
Service Request	`SrvRqst`	1	Issued by a UA to find services or by a UA or SA server during active DA discovery
Service Reply	`SrvRply`	2	The DA or SA response to a service request
Service Registration	`SrvReg`	3	Enables SAs to register new advertisements, to update existing advertisements with new and changed attributes, and to refresh URL lifetimes
Service Deregistration	`SrvDereg`	4	Used by the SA to deregister its advertisements when the service they represent is no longer available

TABLE A–1 SLP Message Types *(continued)*

Message Type	**Abbreviation**	**Function Code**	**Description**
Acknowledgment	`SrvAck`	5	The DA response to an SA's service request or service deregistration message
Attribute Request	`AttrRqst`	6	Made either by URL or by service type to request a list of attributes
Attribute Reply	`AttrRply`	7	Used to return the list of attributes
DA Advertisement	`DAAdvert`	8	The DA response to multicast service requests
Service Type Request	`SrvTypeRqst`	9	Used to inquire about registered service types that have a particular naming authority and are in a particular set of scopes
Service Type Reply	`SrvTypeRply`	10	The message returned in response to the service type request
SA Advertisement	`SAAdvert`	11	UAs employ the `SAAdvert` to discover SAs and their scopes in networks where no DAs are deployed

APPENDIX **B**

SLP Status Codes

This appendix lists and describes the status codes. Status codes indicate that either the request in question is received (code 0), or the receiver is busy. `slpd` returns status codes for unicast messages only.

SLP Status Codes

TABLE B–1 SLP Status Codes

Status Type	**Status Code**	**Description**
No Error	0	Request was processed without error
LANGUAGE_NOT_SUPPORTED	1	For an AttrRqst or SrvRqst, there is data for the service type in the scope, but not in the language indicated.
PARSE_ERROR	2	The message fails to obey SLP syntax.
INVALID_REGISTRATION	3	The SrvReg has problems—for example, a zero lifetime or an omitted language tag

TABLE B–1 SLP Status Codes *(continued)*

Status Type	Status Code	Description
SCOPE_NOT_SUPPORTED	4	The SLP message did not include a scope in its scope list that is supported by the SA or DA that answered the request.
AUTHENTICATION_UNKNOWN	5	The DA or SA received a request for an unsupported SLP SPI.
AUTHENTICATION_ABSENT	6	The UA or DA expected URL and attribute authentication in the SrvReg and did not receive it.
AUTHENTICATION_FAILED	7	The UA or DA detected an authentication error in an Authentication block.
VER_NOT_SUPPORTED	9	Unsupported version number in message
INTERNAL_ERROR	10	An unknown error occurred in the DA or SA. For example, the operating system ran out of file space.
DA_BUSY_NOW	11	The UA or SA should retry, using exponential back off. The DA is busy processing other messages.
OPTION_NOT_UNDERSTOOD	12	The DA or SA received an unknown option from the mandatory range.
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.

Index