INTERNET DRAFT
Category: Informational
Title: draft-ietf-svrloc-template-conversion-08.txt
Date: July 2000

James Kempf
Sun Microsystems, Inc.
Ryan Moats
Coreon, Inc.
Pete St. Pierre
Sun Microsystems, Inc.

Conversion of LDAP Schemas to and from SLP Templates

Status of this Memo

This document is an individual contribution for consideration by the SrvLoc Working Group of the Internet Engineering Task Force.

Distribution of this memo is unlimited.

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at:

http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at:

http://www.ietf.org/shadow.html.

Abstract

This document describes a procedure for mapping between SLP service advertisments and LDAP descriptions of services. The document covers two aspects of the mapping. One aspect is mapping between SLP service type templates and LDAP directory schema. Because the SLP service type template grammer is relatively simple, mapping from service type templates to LDAP types is straightforward. Mapping in the other direction is straightforward if the attributes are restricted to use just a few of the syntaxes defined in RFC 2252. If arbitrary ASN.1 types occur in the schema, then the mapping is more complex and may even be impossible. The second aspect is representation of service information in an LDAP directory. The recommended representation simplifies interoperability with SLP by allowing SLP directory agents to backend into LDAP directory servers. The resulting system allows service advertisements to propagate easily between SLP and LDAP.

Table of Contents

1.0 Introduction

SLP templates [1] are intended to create a simple encoding of the syntactic and semantic conventions for individual service types, their attributes, and conventions. They can easily be generated, transmitted, read by humans and parsed by programs, as it is a string based syntax with required comments. Directory schemas serve to formalize directory entry structures for use with LDAP [2] These directories serve to store information about many types of entities. Network services are an example of one such entity.

Interoperability between SLP and LDAP is important so clients using one protocol derive benefit from services registered through the other. In addition, LDAP directory servers can serve as the backend for SLP directory agents (DAs) if interoperability is possible In order to facilitate interoperability, this document creates mappings between the SLP template grammar and LDAP directory schema, and establishes some conventions for representing service advertisements in LDAP directories. The goal of the translation is to allow SLPv2 queries (which are syntactically and semantically equivalent to LDAPv3 string queries [7]) to be submitted to an LDAP directory server by an SLP DA backended into LDAP without extensive processing by the DA.

The simple notation and syntactic/semantic attribute capabilities of SLP templates map easily into directory schemas, and are easily converted into directory schemas, even by automated means. The reverse may not be true. If the LDAP schema contains attributes with unrecognized or complex syntaxes, the translation may be difficult or impossible. If, however, the LDAP schema only uses a few of the

common syntaxes defined in RFC 2252 [8], then the translation is more straightforward. In addition, to foster complete bidirectionality, the mapping must follow a very specific representation in its DESC attributes.

This document outlines the correct mappings for SLP templates into the syntactic representation specified for LDAP directory schema by RFC 2252 [8]. This syntax is a subset of the ASN.1/BER described in the X.209 specification [9], and is used by the LDAPv3 [2] directory schema. Likewise, rules and guidelines are proposed to facilitate consistent mapping of ASN.1 based schemas to be translated in the SLP template grammar. Finally, a proposal for a representation of service

advertisements in LDAP directory services is made that facilitates SLP interoperability.

Except when used as elements in the definition of LDAP schemas, 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 [16].

2.0 Mapping SLP Templates to LDAP Schema

We define the following abstract object class as the parent class for all services. Any specific service type is a subclass of this, with its own attributes:

```
( 1.3.6.1.4.1.6252.2.27.6.2.1
  NAME 'slpService'
  DESC 'parent superclass for SLP services'
  ABSTRACT
  SUP top
  MUST  ( template-major-version-number $
          template-minor-version-number $
          description $
          template-url-syntax $
          service-advert-service-type $
          service-advert-scopes )
  MAY   ( service-advert-url-authenticator $
          service-advert-attribute-authenticator ) )
```

The attributes correspond to various parts of the SLP service

template and SLP service advertisement.

SLP service type templates begin with four definitions that set the context of the template:

template-type - This defines the service type of the template. The service type can be a simple service type, like ``service:ftp'', an abstract service type, like ``service:printer'' or a concrete service type, like ``service:printer:lpr''. The type name can additionally include a naming authority, for example ``service:printer.sun:local''. The name that appears in this field omits the ``service:'' prefix.

template-version - A string containing a major and minor version number, separated by a period.

template-description - A block of human readable text describing what the service type does.

template-url-syntax - An ABNF [6] grammer describing the service type specific part of the service URL.

The SLP template-type definition is used as the name of the LDAP object class for the template, a subclass of the ``slpService'' class, together with the ``service'' prefix to indicate that the name is for a service. In the translating service type name, colons and the period separating the naming authority are converted into hyphens. If the template defines an SLP concrete type, the concrete type name is used; the abstract type name is never used. For example, the template for ``service:printer:lpr'' is translated into an LDAP object class called ``service-printer-lpr''. Furthermore, if the type name contains a naming authority, the naming authority name must be included. For example, the service type name ``service:printer.sun:local'' becomes ``service-printer-sun-local''. The LDAP object class is always ``STRUCTURAL''.

The template-version definition is partitioned into two attributes, template-major-version-number and template-minor-version-number. The LDAP definition for these attributes is:

```
( 1.3.6.1.4.1.6252.2.27.6.1.1
  NAME 'template-major-version-number'
  DESC 'The major version number of the service type template'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE
)

( 1.3.6.1.4.1.6252.2.27.6.1.2
  NAME 'template-minor-version-number'
  DESC 'The minor version number of the service type template'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE
)
```

The template-url-syntax definition in the SLP template is described by the following attribute:

```
( 1.3.6.1.4.1.6252.2.27.6.1.3
  NAME 'template-url-syntax'
  DESC 'An ABNF grammar describing the service type
```



```
        specific part of the service URL'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
  SINGLE-VALUE
)
```

The template-description attribute is translated into the X.520 standard attribute ``description'' [3].

We further establish the convention that SLP template characteristics that can't be translated into LDAP are inserted into the DESC field of the object class definition. The items are separated by empty lines (consisting of two "LINE FEED" characters), are preceded by a LINE FEED character, and are tagged at the beginning of the line to indicate what they represent. This allows the template to be

reconstructed from the schema by properly parsing the comments.

The bulk of an SLP template consists of attribute definitions. There are four items in an SLP template attribute definition that need to be mapped into LDAP:

Attribute Name - Since SLPv2 attribute names are defined to be compatible with LDAPv3, SLP attributes map directly into LDAP attributes with no change. Similarly, LDAP attributes map directly to SLP attributes.

Attribute Type - The SLP attribute type is mapped into the LDAP attribute type.

Attribute Flags - The SLP attribute flags are mapped into characteristics of the LDAP attribute definition, or into the DESC field if no equivalent LDAP attribute definition characteristic occurs.

Default and Allowed Values - These must be handled by the client or a DA enabled to handle templates, as in SLP. For reference, however, they should be included in the DESC field of the LDAP attribute definition.

Descriptive Text - The SLP template descriptive text should be mapped into the DESC field.

We discuss mapping of types, flags, default and allowed values, and descriptive text in the subsections below.

OIDs for SLP template conversion schema elements are standardized under the enterprise number of SrvLoc.Org (6252) [18].

For purposes of representing an SLP entry, we also define two standardized LDAP syntaxes and attributes with standardized OIDs.

```
  ( 1.3.6.1.4.1.6252.2.27.6.2.2
    DESC 'SLP Service Type'
  )
```

Defines the syntax for the service type name. The syntax is defined in the BNF for the service URL in RFC 2609 Section 2.1 [1].

```
  ( 1.3.6.1.4.1.6252.2.27.6.2.3
    DESC 'SLP Scope'
  )
```

Defines the syntax for the scope name. The syntax is defined in the BNF for scope names in RFC 2608 Section 6.4.1 [5].

```
  ( 1.3.6.1.4.1.6252.2.27.6.1.4
    NAME 'service-advert-service-type'
    DESC 'The service type of the service advertisement, including the
          "service:" prefix.'
    EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.6252.2.27.6.2.2
    SINGLE-VALUE
  )
```

Defines an attribute for the service type name.

```
  ( 1.3.6.1.4.1.6252.2.27.6.1.5
    NAME 'service-advert-scopes'
    DESC 'A list of scopes for a service advertisement.'
    EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.6252.2.27.6.2.3
  )
```

Defines a multivalued attribute for the scopes.

Searches for abstract types can be made with an LDAP query that wildcards the concrete type. For example, a search for all service advertisements of the printer abstract type can be made with the

following query:

(service-advert-service-type=service:printer:*)

SLP specifies that service URLs and attribute lists can be accompanied by a structured authenticator consisting of a digital signature and information necessary to verify the signature. A syntax and two standardized SLP attributes are defined for this purpose:

(1.3.6.1.4.1.6252.2.27.6.2.3 DESC 'SLP Authenticator')

The syntax of an SLP authenticator is the bytes of the authenticator in network byte order, see RFC 2608, Section 9.2 [5].

```
( 1.3.6.1.4.1.6252.2.27.6.1.6
  NAME 'service-advert-url-authenticator'
  DESC 'The authenticator for the URL, null if none.'
  SYNTAX 1.3.6.1.4.1.6252.2.27.6.2.3
  SINGLE-VALUE
)
```

This attribute contains the SLP URL authenticator, as defined in RFC 2608, Section 9.2 [5].

```
( 1.3.6.1.4.1.6252.2.27.6.1.7
  NAME 'service-advert-attribute-authenticator'
  DESC 'The authenticator for the attribute list, null if none.'
  SYNTAX 1.3.6.1.4.1.6252.2.27.6.2.3
  SINGLE_VALUE
)
```

This attribute contains the SLP attribute authenticator, as defined in RFC 2608, Section 9.2 [5].

2.1 Mapping from SLP Attribute Types to LDAP Attribute Types

We define the mapping from SLP attribute types to LDAP as follows:

SLP Type	ASN.1 Type	LDAP Type
Integer	INTEGER	INTEGER

String	DirectoryString	Directory String
Boolean	BOOLEAN	Boolean
Opaque	OCTET STRING	Octet String
Keyword	(N/A)	IA5 String

The following subsections discuss further details of the mapping.

2.1.1 Integer

SLP integers compare as integers when performing a query. LDAP integers behave similarly. Consequently, the mapping from the SLP integer type to LDAP is INTEGER, with the integerMatch matching rule.

2.1.2 String

SLP strings are encoded as described in the SLP protocol specification [5]. All value strings are considered case insensitive for matching operations. SLP strings are not null terminated and are encoded in UTF-8.

SLP strings are mapped to the LDAP Directory String type. The Directory String type exactly matches the SLP string type, i.e. it is a non-null terminated UTF-8 string. The caseIgnoreMatch equality rule, caseIgnoreOrderingMatch ordering rule, and caseIgnoreSubstringsMatch substring rule are used for comparing string attribute values.

2.1.3 Boolean

Boolean attributes may have one of two possible values. In SLP, these values are represented as strings, TRUE and FALSE. In SLP's string encoding of a boolean value, case does not matter.

The SLP Boolean type maps directly into an LDAP BOOLEAN. The caseIgnoreMatch rule is used for equality matching.

2.1.4 Opaque

SLP attribute values of type Opaque are represented as OCTET STRING in LDAP, and the octetStringMatch matching rule is used to compare them.

2.2 Keyword Attributes

SLP service type templates allow the definition of keyword

attributes. Keyword attributes are attributes whose only characteristic is their presence. Keyword attributes have no flag

information, nor any default or allowed values (since, by definition, they have no values).

ASN.1 has no concept of keyword attributes. Keyword attributes are translated into a ``May'' clause in the ASN.1 class definition for the service type. If the keyword attribute is present, then its value is of no consequence, but for consistency we make it simply the NUL character, ``\00''.

2.3 Template Flags

SLP template flags can be handled as described in the following subsections.

2.3.1 Multi-valued

Multi-valued attributes are defined in an SLP template using the one value. All values for a given attribute must be of the same type.

LDAP attribute definitions require that a single valued attribute include the SINGLE-VALUE tag if the attribute is single valued. Otherwise, the attribute is assumed to be multivalued by default.

2.3.2 Optional

SLP uses the 'O' flag to indicate an attribute may or may not be present. These optional attributes are defined using the "May" clause in the ASN.1 definition class definition for the service type. All other attributes must be defined as a "Must".

2.3.3 Literal

ASN.1 does not have a mechanism to indicate that the values of an attribute may not be translated from one language to another, since ASN.1 schema are not typically translated. This flag is dropped when translating a template, but presence of the flag should be noted in the DESC field. It should be placed on a separate line and tagged with ``Literal:'' so the template can be reconstructed from the schema.

2.3.4 Explicit Matching

The SLP template syntax uses a flag of 'X' to indicate that an attribute must be present in order for the query to be properly satisfied. There is no provision for requiring that particular attributes be in a query. Consequently, this flag is dropped when translating a template, but presence of the flag should be noted in the DESC field. It should be placed on a separate line and tagged

with ``Explicit:'' so the template can be reconstructed from the schema.

2.4 Default and Allowed Value Lists

The SLP template grammar provides the capability to define default and allowed values for an attribute. The SLP protocol does not enforce these restrictions on registered attributes, however. The default and allowed values may be used by client side applications, or alternatively it may also be used by DAs to initialize registrations having no attributes and to limit attribute values to the template allowed values.

LDAP servers also do not support default and allowed values on attributes. Therefore, enforcement of default and allowed values in SLP templates is left up to the clients or a DA, if the DA is backending into LDAP. The default and allowed values should be included in the DESC field. The comments should be placed on separate lines and labeled with the ``Default:'' and ``Allowed:'' tags to allow reconstruction of the template.

2.5 Descriptive Text

The descriptive text associated with an attribute definition should be included in the DESC field. It should start on a separate line and begin with the ``Description:'' tag.

2.6 Generating LDAP Attribute OIDs

LDAP attributes require an OID. In general, there is no a priori way that an algorithm can be defined for generating OIDs, because it will depend on the conventions used by the organization developing the

template. In some cases, an organization's procedure for generating OIDs may be regular enough that a template developer can algorithmically generate OIDs off of an assigned root. Whatever means is used, the template developer should assure that unique OIDs are assigned to each SLP attribute that is translated into an LDAP attribute.

2.7 Example

The template included below is a hypothetical abstract printer service template, similar to that described in [10].

```
template-type = printer

template-version = 0.0
```



```
template-description =
The printer service template describes the attributes
supported by network printing devices.  Devices may be
either directly connected to a network, or connected to a
printer spooler that understands the a network queuing
protocol such as IPP, lpr or the Salutation  Architecture.

template-url-syntax =
;The URL syntax is specific to the printing protocol being
;employed

description = STRING
# This attribute is a free form string that can contain any
# site-specific descriptive information about this printer.

printer-security-mechanisms-supported = STRING L M
none
# This attribute indicates the security mechanisms supported
tls, ssl, http-basic, http-digest, none

printer-operator = STRING O L M
# A person, or persons responsible for maintaining a
# printer on a day-to-day basis, including such tasks
# as filling empty media trays, emptying full output
# trays, replacing toner cartridges, clearing simple
```

```
# paper jams, etc.

printer-location-address = STRING O
# Physical/Postal address for this device.  Useful for
# nailing down a group of printers in a very large corporate
# network.  For example: 960 Main Street, San Jose, CA 95130

printer-priority-queue = BOOLEAN O
FALSE
# TRUE indicates this printer or print queue is a priority
# queuing device.

printer-number-up = INTEGER O
1
# This job attribute specifies the number of source
# page-images to impose upon a single side of an instance
# of a selected medium.
1, 2, 4

printer-paper-output = STRING M L O
standard
# This attribute describes the mode in which pages output
# are arranged.
```



```
standard, noncollated sort, collated sort, stack, unknown
```

We assume that the concrete type ``service:printer:lpr'' for printers that speak the LPR protocol [4] has the following template definition:

```
template-type = printer:lpr

template-version = 0.0

template-description =
The printer:lpr service template describes the attributes
supported by network printing devices that speak the
LPR protocol. No new attributes are included.

template-url-syntax = queue
```

```
  queue = ;The queue name, see RFC 1179.
```

The LDAP class definition for the ``service:printer:lpr'' concrete service type is translated as follows:

```
  ( ---place the assigned OID here---
    NAME  'service-printer-lpr'
    DESC  'Description: The printer:lpr service template
                describes the attributes supported by network printing
                devices that speak the LPR protocol. No new attributes
                are included.

           URL Syntax: queue
                queue = ;The queue name, see RFC 1179.'
    SUP   slpService
    MUST  ( description $ security-mechanisms-supported $
    labeledURI)
    MAY   ( operator $ location-address $ priority-queue $
            number-up $ paper-output)
  )
```

The attribute definitions are translated as follows:

```
  ( ---place the assigned OID here---
    NAME 'printer-security-mechanisms-supported'
    DESC 'Description: This attribute indicates the security mechanisms
          supported.

          Default: value
```



```
          Allowed: tls, ssl, http-basic, http-digest, none

          Literal:'
    EQUALITY caseIgnoreMatch
    ORDERING caseIgnoreOrderingMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  )

  ( ---place the assigned OID here---
    NAME 'printer-operator'
```

```
      DESC 'Description: A person, or persons responsible for
            maintaining a printer on a day-to-day basis, including
            such tasks as filling empty media trays, emptying full
            output trays, replacing toner cartridges, clearing simple
            paper jams, etc.

            Literal:'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    )

    ( --place the assigned OID here---
      NAME 'printer-location-address'
      DESC 'Description Physical/Postal address for this device.
            Useful for nailing down a group of printers in a very
            large corporate network.  For example: 960 Main Street,
            San Jose, CA 95130.'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
      SINGLE-VALUE
    )

    ( ---place the assigned OID here---
      NAME 'printer-priority-queue'
      DESC 'Description: TRUE indicates this printer or print
           queue is a priority queuing device.'
      EQUALITY booleanMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
      SINGLE-VALUE
    )

    ( ---place the assigned OID here---
      NAME 'printer-number-up'
```



```
      DESC 'Description: This job attribute specifies the number
            of source page-images to impose upon a single side of
            an instance of a selected medium. This attribute is
            INTEGER.
```

```
            Default: 1

            Allowed: 1, 2, 3, 4'
      EQUALITY integerMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
      SINGLE-VALUE
    )

    ( ---place the assigned OID here---
      NAME 'printer-paper-output'
      DESC 'Description: This attribute describes the mode in
            which pages output are arranged. Default value is
            standard.

            Default: standard

            Allowed: standard, noncollated sort, collated sort,
              stack, unknown.
            Literal:'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    )
```

3.0 Attribute Name Conflicts

LDAP has a flat name space, and attribute names and OIDs must be unique in a directory server. In order to avoid name conflicts in the translation of SLP templates to LDAP schemas, template developers may want to consider prepending the name of the service type to the attribute. Postprocessing attribute names to make them unique when translated is not possible, because it would require the DA to rewrite queries before submitting them to the directory server. In addition, developers should use standard LDAP attributes when such attributes are available.

In the above example template, the abstract type name ``printer'' is prepended to attributes to avoid conflicts. The standard ``description'' attribute defined by X.520 [3] is used to translate the template description attribute.

4.0 Mapping from Schema to Templates

The reverse mapping from LDAP schema to SLP service type templates requires dealing with both LDAP and ASN.1 data types. RFC 2252 defines 33 attribute syntaxes that should be supported by LDAP directory servers. These syntaxes are defined using BNF for strings or using ASN.1 for binary valued attributes defined by X.520.

Mapping of the LDAP data types into SLP template types is fairly straightforward, but mapping arbitrary ASN.1 data types is somewhat more complicated and requires encoding the ASN.1 data type into a string. To a certain extent, this masks the ASN.1 data type because it becomes impossible to distinguish between a native string having content equivalent to an encoded ASN.1 string. However, inclusion of the ASN.1 data type in the comment provides additional information should a reverse transformation from SLP to ASN.1 be required.

The following subsections deal with both LDAP and ASN.1 attribute data type mappings.

4.1 Mapping LDAP Attribute Syntaxes to SLP Attribute Types

The following table contains the mappings for LDAP syntaxes to SLP data types:

LDAP Type	SLP Type
ACI Item	NA
Access Point	NA
Attribute Type Description	NA
Audio	Opaque
Binary	ASN.1 escape
Bit String	String
Boolean	Boolean
Certificate	Opaque
Certificate List	Opaque
Certificate Pair	Opaque
Country String	String
DN	String
Data Quality Syntax	NA
Delivery Method	NA
Directory String	String
DIT Content Rule Description	NA
DIT Structure Rule Description	NA
DL Submit Permission	NA
DSA Quality Syntax	NA
Enhanced Guide	NA
Facsimile Telephone Number	String
Fax	Opaque

Generalized Time	String

Guide	NA
IA5 String	String
INTEGER	Integer
JPEG	Opaque
LDAP Syntax Description	NA
LDAP Schema Definition	NA
LDAP Schema Description	NA
Master and Shadow Access Points	NA
Matching Rule Description	NA
Matching Rule Use Description	NA
Mail Preference	NA
MHS OR Address	String
Modify Rights	NA
Name and Optional UID	NA
Name Form Description	NA
Numeric String	String
Object Class Description	NA
Octet String	Opaque
OID	String
Other Mailbox	String
Postal Address	String
Protocol Information	NA
Presentation Address	String
Printable String	String
Substring Assertion	NA
Subtree Specification	NA
Supplier Information	NA
Supplier or Consumer	NA
Supplier And Consumer	NA
Supported Algorithm	NA
DSE Type	NA
Telephone Number	String
Teletex Terminal Identifier	String
Telex Number	String
UTC Time	String

4.2 Mapping ASN.1 Types to SLP Types

ASN.1 employs a much richer set of data types than provided by SLP. The table below show the mapping of selected ASN.1 data type to their

nearest SLP equivalent. Because of the complexity and flexibility of ASN.1, a complete list cannot be provided.

As sample of some ASN.1 encodings and their mappings to SLP:

ASN.1 type	SLP type
INTEGER	Integer

BOOLEAN	Boolean
ENUMERATED	String
OBJECT IDENTIFIER	String
OCTET STRING	Opaque
REAL	String

Data types that do not map directly to SLP data types should be defined as either a String, or as Opaque. ASN.1 types that may only contain valid characters for Strings, as defined in X.680 [9] should be encoded as strings. ASN.1 types such as GraphicString that change their character set encoding in part way through a value should not be encoded as strings, however, If such types are required, the SLP Opaque type should be used. In either case, the first line of the help text is used to indicate the original ASN.1 data type.

The following subsections describe how to convert from the ASN.1 BER [9] to the SLP template for the different types in the table above.

4.2.1 Integer

Both SLP templates and ASN.1 support Integers, so there is a one to one mapping between an SLP Integer attribute and an ASN.1 Integer attribute. Details on the encoding of integers is summarized in the SLP template to ASN.1 section above.

4.2.2 Boolean

Boolean values are supported by both SLP and ASN.1, though on wire encodings differ. X.680 [9] specifies zero and non-zero encoding for booleans, where SLP encodes booleans using the strings TRUE and FALSE. In general, most LDAP servers will use the LDAP Boolean type (which is a string), so again the ASN.1 type should be recorded in the comment or it will be lost.

4.2.3 Enumerated

SLP templates support the concept of enumerations through the listing of allowed values in the attribute definition. These enumerations are not strictly binding on clients or DAs, but they are similar to the ASN.1 definition of enumerations. BER encodes the ASN.1 enumeration by passing the number of the element's position in the enumeration. This requires both sides to have knowledge of the specific enumeration prior to decoding an enumeration's value. SLP provides no specific support for transmitting enumerations. They are simply String types. Information on the ASN.1 type and ASN.1 encoding of the enumeration values is recorded in the comment.

Example:


```
color-supported = STRING   M
none
# ASN.1: Enumeration.
# ASN.1 Mapping: none = 0, highlight = 1, three color = 2, four color = 4,
#   monochromatic = 5
#This attribute specifies whether the Printer supports
# color and, if so, what type.
none,highlight,three color,four color,monochromatic
```

4.2.4 Object Identifier

Object identifiers(OIDs) are commonly used in the ASN.1 world to identify object and attributes. OIDs are a numerical representation of an element's place in the naming hierarchy. Each element at a particular level of a hierarchy has a unique number assigned within that level of the hierarchy. A sample OID would be the naming tree for SNMP MIBs: iso(1) org(3) dod(6) internet(1) mgmt(2) mib(1) would be written as the string ``1.3.6.1.2.1''.

Because this representation reduces down to a string of dot separated numbers, this maps easily to the SLP String type. The help text for this element should indicate it is an ASN.1 OID

```
  identifier = STRING
```

```
  # ASN.1: OID
  # The object identifier for this SNMP agent.
```

4.2.5 Octet String

An ASN.1 octet string should be mapped to an Opaque in an SLP template. An octet string is a sequence of bytes, whereas an Opaque is a a string that encodes a sequence of bytes. Again, the ASN.1 type is lost unless recorded in the comment.

4.2.6 Real

There is no direct mapping between floating point numbers and any SLP data types. Attributes having the ASN.1 type of Real are mapped to SLP type String. Comments are added to the attribute help text indicating the value was originally an ASN.1 real. For example:

```
  weight = STRING
  # ASN.1: Real
  # The objects weight in pounds.
```


4.3 Example ASN.1 Schema

The following is an example schema for an exported filesystem. The section presents it as in ASN.1 and the following section shows the SLP template translation. The template translation does not capture the actual attribute format for the Set type, that would be done in the LDAP client software making the translation. Note that even though the class definition does not conform with the previously defined conventions for SLP classes, the schema can still be translated into an SLP template. The syntax used in this example follows

```
      -- Abstraction of a fstab entry (a "mount").
      -- These lookups would likely be performed by an
      -- an automounter type application.
      mount   OBJECT-CLASS ::= {
```

```
        SUBCLASS OF { top }
        MUST CONTAIN { mountHost |
                       mountDirectory |
                       mountType
                     }
        MAY CONTAIN { mountOption |
                      mountDumpFrequency |
                      mountPassNo
                    }
        ID { <oid1> }
}


- The mount host.
mountHost       ATTRIBUTE ::= {
                WITH SYNTAX caseIgnoreString
                EQUALITY MATCHING RULE caseIgnoreMatch
                SINGLE VALUE
                ID { <oid2> }
}


- The file system to mount.
mountDirectory  ATTRIBUTE ::= {
                WITH SYNTAX caseIgnoreString
                EQUALITY MATCHING RULE caseIgnoreMatch
                SINGLE VALUE
                ID { <oid3> }
}
```



```
- The type of file system being mounted.
mountType       ATTRIBUTE ::= {
                WITH SYNTAX INTEGER { ufs(1),
                                      hsfs(2),
                                      nfs(3),
                                      rfs(4)
                                    }
                EQUALITY MATCHING RULE integerMatch
                SINGLE VALUE
                ID { <oid4> }
}
```

```
        - Options for the mount operation.
        mountOption     ATTRIBUTE ::= {
                        WITH SYNTAX caseIgnoreString
                        EQUALITY MATCHING RULE caseIgnoreString
                        ID { <oid5> }
        }


        - How often to dump the file system.
        mountDumpFrequency      ATTRIBUTE :: = {
                                WITH SYNTAX  INTEGER (0..9)
                                EQUALITY MATCHING RULE integerMatch
                                SINGLE VALUE
                                ID { <oid6> }
        }

        - Boot time mount pass number.
        mountPassNo     ATTRIBUTE ::= {
                        WITH SYNTAX INTEGER
                        EQUALITY MATCHING RULE integerMatch
                        SINGLE VALUE
                        ID { <oid7> }
        }
```

The translated SLP template is:

```
  template-type = mount

  template-version = 1.0

  template-description = "Describes a remote filesystem access protocol"

  template-url-syntax =
              filesystem   = 1*[ DIGIT / ALPHA ]
              urlpath = "/" filesystem
```



```
  mountHost = STRING L
  # ASN.1: Case Ignore String, Single Value
  # The mount host

  mountDirectory = STRING L
```

```
# ASN.1: Case Ignore String, Single Value
# The filesystem to mount

mountType = STRING L
ufs
# ASN.1: Enumeration, Single Value
# ASN.1 Mapping: ufs = 1, hsfs = 2, nfs = 3, rfs = 4
# The type of the filesystem being mounted
ufs, hsfs, nfs, rfs

mountOption = STRING M O L
# ASN.1: Case Ignore String
# mount options for this filesystem

mountDumpFrequency = INTEGER O
0
# ASN.1: Integer Range, Single Value
# How often to dump this filesystem
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

mountPassNo = INTEGER O
# ASN.1: Integer, Single Value
# Boot time mount pass number
```

5.0 Representing SLP Service Advertisments in an LDAP DIT

In addition to translating between SLP templates and LDAP schema, another area requiring compatibility is the representation of SLP service advertisements in an LDAP DIT. A standardized representation for service information allows SLP DAs to store service advertisements in LDAP, and for LDAP clients to query the DIT for those services. Similarly, if LDAP clients represent service information in the same form, SLP clients can benefit from interoperability.

A service advertisement contains the service URL in a 'labeledURI' attribute [11]. The labeledURI attribute in a service advertisement should only contain the service URL for the service, with no additional label. It is recommended that the labeledURI be used as the RDN for the service object in the DIT.

Although service advertisements can appear anywhere within the DIT,

it is recommended that all services be stored under a single common point, or root node, to facilitate searching in a domain. This allows a client to search for all of advertisements of a particular service type, say, for all printers. The recommended parent entry is one named "ou=service" below the entry which is the representation of the domain, as described in RFC 2247.

For example, a printer service with labeledURI of "service:lpr://printsrv/queue1" in the domain foobar.com advertised in the LDAP server that holds the entry "dc=foobar,dc=com" tree has the following DN:

```
"labeledURI=service:lpr://printsrv/queue1, ou=service, dc=foobar, dc=com"
```

While this leads to a flat space of service storage, since SLP uses search filters from LDAP for searches, these filters can be used for one-level searches from the root node.

The following example illustrates how an advertisement having a simple service type is represented. The advertisment (in conceptual form) for a printer is:

```
Service Type: service:lpr://printsrv/queue1
Scopes: eng,corp
Attributes:
  description = A general printer for all to use.
  security-mechanisms-supported = none
Authentication: none
```

The RDN of the object is labeledURI=service:lpr://printsrv/queue1, and the following LDAP search filter will return this object, along with any others of the service type ``service:lpr'' that match the other attributes:

```
(&(service-advert-service-type=service:lpr)
  (service-advert-scopes=eng)
  (service-advert-scopes=corp)
  (description=A general printer for all to use.)
  (security-mechanisms-supported=none))
```

Service advertisements in SLP also have a lease time associated with them. In LDAP servers that support the extensions for dynamic directory services [12], the service advertisement entry objectClass should be extended with the dynamicObject class. This allows the service advertisment to time out within the LDAP directory server. If the LDAP directory server does not support the dynamic directory services extension, then advertisement lease timeouts must be handled

by the SLP agent.

While the service advertisement schema outlined in this section is primarily for SLP DAs that use LDAP as a backing store, if LDAP agents register services using the same format, complete interoperability with SLP is achieved.

6.0 Internationalization Considerations

SLP specifies that an RFC 1766 [13] language code accompanies every service advertisement. Language codes for service advertisements in LDAP must be represented according to RFC 2596 [14].

RFC 2596 prohibits language codes in DNs, and specifies that a directory server which does not support language codes must treat an attribute with a language code as an unrecognized attributes. According to RFC 2596, language codes are appended to attribute names with a semicolon (``;''). For example, the following attribute/value pair is in the German locale:

(address;lang-de=44 Bahnhofstrasse, 2365 Weibstadt, Deutschland)

An attribute with a language tag in a specific locale is considered a separate attribute from attributes in other locales.

If the service advertisement is in the default SLP locale (``en'', no dialect), then the language code need not be appended to the attribute name.

SLP queries in locales other than the default need not be rewritten to include language tags before being submitted to the directory server. RFC 2596 specifies that all entries that match are returned, including those with language tags, without requiring the language tags to be explicitly present in the query. The SLP DA can then postprocess the result to select the entries from the required locale.

7.0 Security Considerations

SLP authenticators are stored with the service advertisement in the DIT, as discussed in Section~7ef{slpdit}. LDAP clients need to use LDAP authentication [15] to assure that they are connecting with a

secure server. In particular, SLP DAs that use LDAP as a back end store and that implement SLP authentication MUST use LDAP authentication to assure that the LDAP entries for their service registrations are secure.

Acknowledgements

Many thanks are due to Mark Wahl whose detailed and insightful comments were instrumental in helping improve the technical accuracy of this draft with respect to LDAP.

8.0 References

[1] E. Guttman, C. Perkins, J. Kempf. Service Templates and service: Schemes. RFC 2609, April, 1999.

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

[3] International Telecommunications Union. The Directory:Selected Attribute Types. ITU Recommendation X.520. August, 1997.

[4] L. McLaughlin. Line Printer Daemon Protocol. RFC 1179. August, 1990.

[5] E. Guttman, C ~Perkins, J. Veizades, and M. Day. Service Location Protocol Version 2. RFC 2608. April, 1999.

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

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

[8] M. Wahl, A. Coulbeck, T. Howe, and S. Kille. Lightweight Directory Access Protocol (v3): Attribute Syntax Definition. RFC 2252. December, 1997.

[9] ITU-T Rec. X.680. Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation. 1994.

[10] P. St. Pierre, S. Isaccson, I. McDonald. Definition of printer:

URLs for use with Service Location. draft-ietf-svrloc-printer-scheme-xx.txt. Work in Progress.

[11] M. Smith. Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs). RFC 2079. January, 1997.

[12] Y. Yaacovi, M. Wahl, and T. Genovese. Lightweight Directory Access Protocol (v3): Extensions for Dynamic Directory Services. RFC 2589. May, 1999.

[13] H. Alvestrand. Tags for the Identification of Languages. RFC 1766. December, 1997.

[14] M. Wahl and T. Howes. Use of Language Codes in LDAP. RFC 2596. May, 1999.

[15] M. Wahl, H. Alvestrand, J. Hodges, and R. Morgan. Authentication Methods for LDAP. draft-ietf-ldapext-authmeth-xx.txt. Work in Progress.

[16] S. Bradner. Key Words for Use in RFCs to Indicate Requirement Levels. RFC 2119. March 1997.

[17] Dubuisson, O. ASN.1: Communication between Heterogeneous Systems. OSS Nokalva, 2000.

[18] http://www.srvloc.org

9.0 Full Copyright Statement

developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

10.0 Authors' Address

James Kempf
Sun Microsystems
901 San Antonio Avenue
Palo Alto, CA 94303
USA

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

Ryan Moats
Coreon, Inc.
15621 Drexel Circle
Omaha, NE, 68135

rmoats@coreon.net

Pete St. Pierre
Sun Microsystems
901 San Antonio Avenue
Palo Alto, CA 94303
USA

Phone: +1 415 786-5790
Email: Pete.StPierre@Eng.Sun.COM