VIEWDS

IDENTITY MANAGEMENT AND XML
DIRECTORY SERVICES SOLUTIONS

DSML vs XLDAP

A comparison of XML-based directory protocols

Introduction

The Directory Services Markup Language (DSML) version 2 and the XML Lightweight Directory Access Protocol are two different mechanisms for accessing a directory using an XML-based protocol.

On the surface it would appear that DSML and XLDAP are very much the same and that there isn't a need for both.

This paper attempts to identify and highlight the differences between DSML and XLDAP. It aims to highlight why XLDAP is truly an XML-based directory access protocol, whilst DSML is only XML-based in some areas.

This paper has been written by eNitiatives who develop ViewDS and promote and support the XLDAP protocol.

What is DSML trying to achieve?

DSML was created to meet the following objectives:

1. Provide a method that expresses query and update operations, and their results, as XML documents
2. Allow greater accessibility of directory services to devices that don't support LDAP
3. Allow greater accessibility of directory services through firewalls
4. To allow XML programming tools to access directory services without requiring LDAP-specific capabilities

What is XLDAP trying to achieve?

XLDAP was created to meet the following objectives:

1. Provide a method that expressed query and update operations, and their results, as XML documents
2. Allow greater accessibility of directory services to devices that don't support LDAP
3. Allow greater accessibility of directory services through firewalls
4. To allow XML programming tools to access directory services without requiring LDAP-specific capabilities
5. Allow all of the above capability to operate natively with Directory Services that are used to store XML

What are some of the differences between DSML and LDAP?

LDAP Controls

Controls are something that you will find in most requests sent to a directory. An example of a control used frequently is a request for paged search results. The pages control portion of a search operation in DSML that asks for results to be paged (1 at a time) will look like this:

```
<control type="1.2.840.113556.1.4.319" criticality="true">
<controlValue
        xsi:type="xsd:base64Binary">MIQAAAAGAgID6AQA</controlValue>
</control>
```

In XLDAP, the same control would look like this:

```
<controls>
<control>
    <controlType>1.2.840.113556.1.4.319</controlType>
    <controlValue>
        <request>
            <size>1</size>
            <cookie/>
        </request>
    </controlValue>
</control>
</controls>
```

Why is this significant?

What we can see here is a significant difference between the operations represented in XML.

On one hand we have DSML, which requires a Base64 encoding of a BER encoding of a value of an ASN.1 type. Unfortunately this requires DSML-enabled clients to understand the ASN.1 type of each LDAP control, be able to BER encode a suitable value and then Base64 encode the value. DSML clients can't get away with using the same value over and over either, because they must be able to decode the BASE64 and BER of the responses they receive.

On the other hand, the entire LDAP control is represented in XLDAP in natural and readily accessible XML.

When compared to XLDAP, which requires no ASN.1, BER or Base64 encoding, DSML is burdensome to the XML programmer for a non-LDAP-enabled client.

LDAP Extended Operations

Extended operations in DSML suffer the same fate as LDAP Controls. Within LDAP, the actual request that forms the extended operation is defined by an ASN.1 type.

An example of an Extended Operation from the DSMLv2 specification shows:

```
<extendedRequest>
   <requestName>1.3.563.52.424</requestName>
   <requestValue xsi:type="xsd:base64Binary">
     TFNNTHYyLjAgcm9ja3MhIQ==
   </requestValue>
</extendedRequest>
```

Decoding the Base64 value in this example reveals “LSMLv2.0 rocks!!”. It should be noted that in practice the Base64 decoded value will never be readable in this fashion; if present it will always be the BER encoding of a value of an ASN.1 type.

XLDAP on the other hand natively supports the XML markup of the entire protocol operation, including Extended Operations.

For example, the request of a Cancel Extended Operation defined in RFC 3909 has the following ASN.1 syntax:

```
CancelRequestValue ::= SEQUENCE {
        cancelID        MessageID
}
```

Supplying a value of this type in a DSML extended operation would require the value to be BER encoded and subsequently Base64 encoded. Using XLDAP, the value is marked up as XML:

```
<extendedReq>
     <requestName>1.3.6.1.1.8</requestName>
     <requestValue>
          <cancelID>1</cancelID>
     </requestValue>
</extendedReq>
```

Why is this significant?

Once again this requires DSML enabled clients to understand ASN.1 types, be able to BER encode and decode values of those types and then BASE64 encode (and decode) the BER encoded value.

When compared to XLDAP which requires no ASN.1, BER or BASE64 encoding, DSML is burdensome to the XML programmer for a non-LDAP-enabled client.

Attribute Values

DSML requires attribute values to be provided in their LDAP string encodings or alternatively in base64Binary or as referenced by a URI.

Examples of this include:

Distinguished Names

<value>CN=John Smith, DC=eNitiatives, DC=com</value>

Directory Strings

```
<value>Johnson</value>
```

XML Data

```
<value>&lt;?xml version='1.0'?>
&lt;value xmlns:n0=http://www.w3.org/2001/XMLSchema-
instance xmlns:n1="eb2bcom:testSchema"
n0:type="n1:SkillSet">
     &lt;item>
          &lt;category>IT Development&lt;/category>
          &lt;description>Java&lt;/description>
          &lt;proficiency>10&lt;/proficiency>
     &lt;/item>
     &lt;item>
          &lt;category>IT  Development&lt;/category>
          &lt;description>C#&lt;/description>
          &lt; proficiency>8&lt;/proficiency>
     &lt;/item>
&lt;/value>
</value>
```

The string encodings for LDAP attributes are not algorithmic; they are defined within RFC's such as RFC4517, RFC4515 and RFC4514. These RFC's provide specific instructions relating to the string formatting of over 35 different syntaxes.

An application using DSML must contend with these string encodings, imposing a heavy reliance on LDAP specifications. The coverage of the DSML XML operations does not extend down to the directory attribute value level. Even worse, if DSML is used with directory attribute values containing XML, then the client must go through additional steps of escaping all of the XML content!

XLDAP doesn't have any of these shortcomings because it is completely XML based, including the directory attribute values. XLDAP clients do not need to learn about ASN.1 types or LDAP string encodings.

Examples of the previous values in XLDAP would be (note various namespace declarations have been removed from elements):

Distinguished Names

```
<value>
     <item>
          <item>
               <type>0.9.2342.19200300.100.1.25</type>
               <value>com</value>
          </item>
     </item>
     <item>
          <item>
               <type>0.9.2342.19200300.100.1.25</type>
               <value>eNitiatives</value>
          </item>
     </item>
     <item>
          <item>
               <type>2.5.4.3</type>
               <value>John Smith</value>
          </item>
     </item>
</value>
```

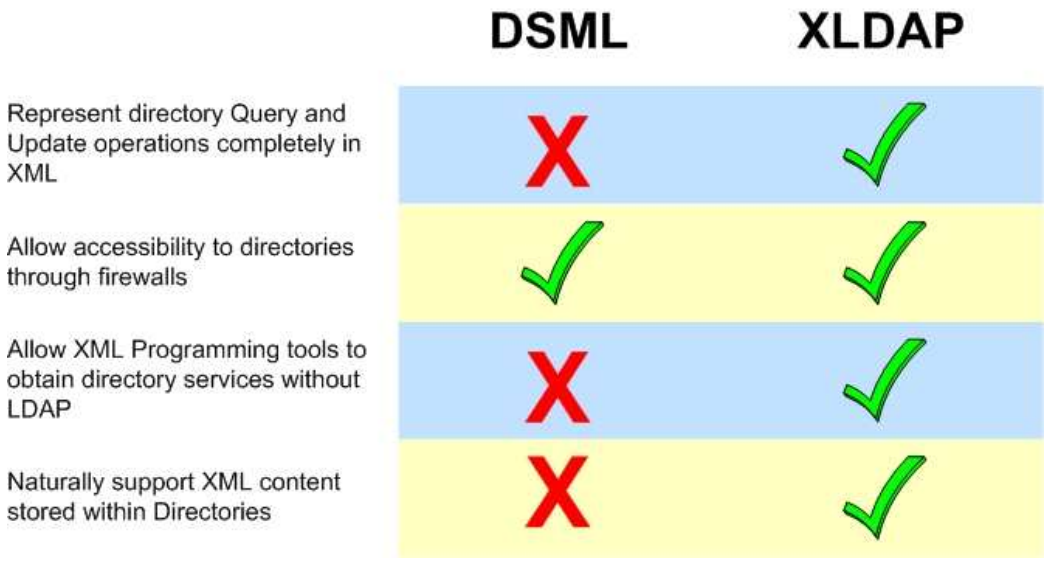
Directory Strings

```
<value>Johnson</value>
```

XML Data

```
<value>
     <item>
          <category>IT Development</category>
          <description>Java</description>
          <proficiency>10</proficiency>
     </item>
     <item>
          <category>IT Development</category>
          <description>C#</description>
          <proficiency>10</proficiency>
     </item>
</value>
```

What is actually achieved by DSML and XLDAP?



I thought DSML could represent directory operations as XML!

DSML only provides certain aspects of directory query and update operations as XML. This does not include Controls, Extended Operations or Attribute Values.

This places the following burdens on DSML Client applications and developers:

- Must support the X.690 Basic Encoding Rules (BER) for ASN.1 Types
- Must additionally Base64 encode binary information that is naturally occurring within the operations (e.g. Controls)
- Must support LDAP String Representations of LDAP ASN.1 Syntaxes
- Must escape any XML values appearing within the directory

XLDAP does not place any of these restrictions on the client application. The entire query and update operations, including directory attribute values, controls and extended operations are provided in XML.

Accessing Directory Services through a Firewall

Both XLDAP and DSML provide a directory service that operates using HTTP and SOAP.

This allows both protocols to operate through firewalls that can audit SOAP or XML based content.

Allowing XML based programming tools to access Directory Services without LDAP

One of the main objectives of DSMLv2 was to increase the accessibility of directory services. A motivating scenario stated in the DSMLv2 specification is “A smart cell phone or PDA needs to access directory information but does not contain an LDAP client”.

In situations such as these, where an XML-based lightweight directory access protocol is required, DSML additionally requires many other components of LDAP; namely BER encoding and LDAP string encoding rules.

XLDAP however operates completely within the realm of XML. All data types are represented purely in XML, without the need for external encoding instructions from LDAP or ASN.1.

XLDAP allows easier adoption, since it only relies on XML-based technologies.

Support for XML Content

Whilst DSML itself is an XML protocol, it does not accommodate for the fact that directories themselves may be storing XML information. In such cases, additional processing is placed on the server and client to ensure that all of the XML value is escaped.

Conclusion

The points discussed in this document hope to convey that whilst DSML takes a step towards providing an XML-based access protocol, it’s realistically just a half measure. XLDAP is a complete solution for an XML-based directory protocol.

XLDAP allows the development of a completely XML based directory client and access protocol; DSML does not.