Abstract:

This specification defines an approach for authentication between parties communicating using the SOAP protocol. The approach utilizes the Simple Authentication and Security Layer (SASL) framework.

Filename: draft-lib-arch-soap-authn-v1.0-03.pdf

Copyright © 2003 Liberty Alliance Project
Notice

Copyright © 2003 ActivCard; American Express Travel Related Services; America Online, Inc.; Bank of America; Bell Canada; Cingular Wireless; Cisco Systems, Inc.; Citigroup; Communicator, Inc.; Consignia; Deloitte & Touche LLP; Earthlink, Inc.; Electronic Data Systems, Inc.; Entrust, Inc.; Ericsson; Fidelity Investments; France Telecom; Gemplus; General Motors; Hewlett-Packard Company; i2 Technologies, Inc.; Internet2; Intuit Inc.; MasterCard International; NEC Corporation; Netegrity; NeuStar; Nextel Communications; Nippon Telegraph and Telephone Corporation; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OneName Corporation; Openwave Systems Inc.; Phaos Technology; PricewaterhouseCoopers LLP; Register.com; RSA Security Inc; Sabre Holdings Corporation; SAP AG; SchlumbergerSema; SK Telecom; Sony Corporation; Sun Microsystems, Inc.; Symlabs, Inc.; Trustgenix; United Airlines; VeriSign, Inc.; Visa International; Vodafone Group Plc; Wave Systems. All rights reserved.

This specification document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not, and shall not be held responsible in any manner, for identifying or failing to identify any or all such third party intellectual property rights. This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights, and fitness for a particular purpose. Implementors of this Specification are advised to review the Liberty Alliance Project’s website (http://www.projectliberty.org/) for information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management Board.

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08855-1331, USA
info@projectliberty.org
Revision History

Revision: 03 Date: 25 Jul 2003
Resolves bugs 380 & 381. Various text & reference fixes and enhancements.

Revision: 02 Date: 27 Jun 2003
Included a validated schema. Updated some processing rules details to account for schema changes and outright bugs.

Revision: 01 Date: 16 Jun 2003
initial draft
Contents

1. Introduction .................................................................................................................. 5
2. Notation and Conventions ............................................................................................... 5
   2.1. Requirements Keywords ......................................................................................... 6
   2.2. XML Namespaces .................................................................................................... 6
3. Terminology .................................................................................................................... 6
4. Abstract Usage Model ..................................................................................................... 7
5. SOAP Header Blocks and SOAP Binding ......................................................................... 8
   5.1. SOAP Binding ............................................................................................................ 9
6. Protocol Messages .......................................................................................................... 9
   6.1. The <SaslRequest> Message .................................................................................... 10
   6.2. The <SaslResponse> Message .................................................................................. 11
7. Concrete SASL Profile ................................................................................................... 12
8. Examples ........................................................................................................................ 14
9. Security Considerations .................................................................................................. 15
10. Acknowledgements ....................................................................................................... 16
11. References ..................................................................................................................... 17
13. B. Schema Listing ........................................................................................................ 19
1. Introduction

The Simple Object Access Protocol (SOAP) specifications, [SOAPv1.1] and [SOAPv1.2], define an XML-based [XML] messaging paradigm, but do not specify any particular security mechanisms. They do not, in particular, describe how one SOAP endpoint may authenticate with another SOAP endpoint via an exchange of SOAP messages.

This specification defines how to perform general identity authentication [WooLam92], also known as peer entity authentication [RFC2828], over SOAP, in the Liberty ID-WSF context [ID-WSF-Primer]. Rather than specify the particulars of one or more authentication mechanisms directly in this specification, we profile the Simple Authentication and Security Layer (SASL) framework [RFC2222].

SASL is an approach to modularizing protocol design such that the security design components, e.g. authentication and security layer mechanisms, are reduced to a uniform abstract interface. This facilitates a protocol’s use of an open-ended set of security mechanisms, as well as a so-called "late binding" between the implementations of the protocol and the security mechanisms’ implementations. This late binding can occur at implementation- and/or deployment-time. The SASL specification also defines how one packages authentication and security layer mechanisms to fit into the SASL framework, as SASL mechanisms, as well as register them with the Internet Assigned Numbers Authority (IANA) [IANA] for reuse.

This specification presents a profile of SASL that is specific to the Liberty ID-WSF use-case. Thus this profile does not address various more general aspects of distributed system peer entity authentication that would otherwise be covered in a less-specific use context.
2. Notation and Conventions

This specification uses schema documents conforming to W3C XML Schema (see [Schema1]) and normative text to describe the syntax and semantics of XML-encoded protocol messages.

2.1. Requirements Keywords

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 [RFC2119]:

"they MUST only be used where it is actually required for interoperability or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)"

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

2.2. XML Namespaces

This specification uses the XML namespace prefixes listed in Table 1.

Table 1. XML Namespaces used in this specification

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>sa:</td>
<td>Represents the Liberty namespace: urn:liberty:sa:2003-08. This namespace is the default for instance fragments, type names, and element names in this specification.</td>
</tr>
<tr>
<td>sb:</td>
<td>Represents the Liberty namespace defined in [LibertySOAPBinding].</td>
</tr>
<tr>
<td>idpp:</td>
<td>Represents the namespace defined in [LibertyIDPP].</td>
</tr>
<tr>
<td>s:</td>
<td>Represents the SOAP namespace: <a href="http://www.w3.org/2001/12/soap-envelope">http://www.w3.org/2001/12/soap-envelope</a>, defined in [SOAPv1.1].</td>
</tr>
<tr>
<td>xs:</td>
<td>Represents the W3C XML schema namespace (<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>) defined in [Schema1].</td>
</tr>
</tbody>
</table>
3. Terminology

Table 2 defines key terminology used in this specification. Definitions for other Liberty-specific terms can be found in [LibertyGlossary]. See also [RFC2828] for overall definitions of security-related terms.

Table 2. Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication server</td>
<td>The precise, specific role played by a server in the protocol message exchanges defined in this specification.</td>
</tr>
<tr>
<td>client</td>
<td>A role assumed by a system entity who either explicitly or implicitly initiates an authentication exchange [RFC2828]. Client is implicitly defined in [RFC2222]. Also known as SASL client.</td>
</tr>
<tr>
<td>initial response</td>
<td>A [RFC2222] term referring to authentication exchange data sent by the client in the initial SASL request. It is used by a subset of SASL mechanisms. See Section 5.1 of [RFC2222].</td>
</tr>
<tr>
<td>role</td>
<td>A function or part performed, especially in a particular operation or process [Merriam-Webster].</td>
</tr>
<tr>
<td>server</td>
<td>A role assumed by a system entity who is intended to engage in defined exchanges with clients. This term is implicitly defined in [RFC2222] and in this specification is always synonymous with authentication server.</td>
</tr>
<tr>
<td>SOAP header block</td>
<td>A [SOAPv1.2] term meaning: An [element] used to delimit data that logically constitutes a single computational unit within the SOAP header. In [SOAPv1.1] these are known as simply SOAP headers, or simply headers. This specification uses the SOAPv1.2 terminology.</td>
</tr>
<tr>
<td>SOAP node</td>
<td>A [SOAPv1.2] term describing system entities who are parties to SOAP-based message exchanges that are, for purposes of this specification, also the ultimate destination of the exchanged messages, i.e. SOAP endpoints. In [SOAPv1.1], SOAP nodes are referred to as SOAP endpoints, or simply endpoints. This specification uses the SOAPv1.2 terminology.</td>
</tr>
<tr>
<td>system entity</td>
<td>An active element of a computer/network system. For example, an automated process or set of processes, a subsystem, a person or group of persons that incorporates a distinct set of functionality [SAMLGloss].</td>
</tr>
</tbody>
</table>
In this specification’s abstract usage model, a SOAP node acting in a client role makes an authentication request to another SOAP node acting as an authentication server, and if the server is willing and able, an authentication exchange will ensue.

The authentication exchange can involve an arbitrary number of round trips, dictated by the particular SASL mechanism employed [RFC2222]. At the end of this exchange of messages, the client SOAP node will either be authenticated or not; if so, it will possess material credentials attesting to this.

In terms of the client knowing where/how to contact the authentication server, it can either already have that information, e.g. via configuration, or it can be dynamically passed the information, e.g. in a response as a result of an inappropriately- or un-authenticated application-level request.

Also, when interacting with the authentication server, the client MAY have out-of-band knowledge of the server’s supported SASL mechanisms, or it MAY request them.

Note

This specification disallows negotiation of a SASL security layer in all cases. See Section 7 for details.
5. SOAP Header Blocks and SOAP Binding

This specification does not define any new SOAP header blocks. Section 5.1, below, constitutes the SOAP binding statement for this specification.

5.1. SOAP Binding

The element messages defined below in Section 6, e.g. <SaslRequest>, are bound to the [SOAPv1.1] protocol by mapping them directly into the <s:Body> element of the <s:Envelope> element comprising a SOAP message. [LibertySOAPBinding] normatively specifies this binding.

Note

Implementations of this specification MUST use the <sb:Correlation> SOAP header block defined in [LibertySOAPBinding] to correlate the message exchanges defined in this specification.
6. Protocol Messages

This section defines the protocol’s messages along with the message element attribute values and their associated semantics. The sequencing of protocol interactions is defined below in Section 7.

The XML schema for the protocol defined by this specification is normatively defined in [SOAPAuthnSchema].

6.1. The <SaslRequest> Message

Figure 1 shows the XML schema fragment [Schema1] describing the <SaslRequest> message. This message is used to initially convey to the server either:

- The client-selected SASL mechanism along with any "initial response" (see Section 5.1 of [RFC2222]), or,
- An indication that the server should respond with a list of the server’s supported SASL mechanisms.

In either case, the client will subsequently use the <SaslRequest> message to continue the authentication exchange as dictated by the asserted SASL mechanism.

Additionally, the mechanism attribute is used in subsequent <SaslRequest> message transmissions to signal the client’s intentions to the server. For example, the client can indicate to the server it wishes to abort the authentication exchange by setting mechanism to a value different than the value used in the prior client message, or "". See Section 7 for details.

The content model of the <SaslRequest> element itself is base64-encoded data.

Figure 1. <SaslRequest> Schema Fragment

6.1.1. Values for mechanism attribute of <SaslRequest>

This list below defines the allowable values for the mechanism attribute of the <SaslRequest> message element.

- Any single SASL mechanism name – as defined in [IANA]’s SASL Mechanism Registry [SASLReg]. The mechanism name SHOULD be taken from the column labeled as "MECHANISMS" in [SASLReg] (this registry document is reproduced in Appendix A for convenience). See Example 1.

Example 1. <SaslRequest> Specifying the "GSSAPI" Mechanism

```
<SaslRequest mechanism="GSSAPI">
  AAAC8IyPqcvt3wCcD......pyuvU0lONm2zf9gFz
</SaslRequest>
```
A list of SASL mechanism names. The name must be as defined in [IANA]'s SASL Mechanism Registry [SASLReg] and separated by ASCII space chars (hex "20"). The mechanism names SHOULD be taken from the column labeled as "MECHANISMS" in [SASLReg] (this registry document is reproduced in Appendix A for convenience). See Example 2. In this case, the <SaslRequest> message MUST NOT contain any "initial response" data.

Example 2. <SaslRequest> Specifying the "GSSAPI" Mechanism

```xml
<SaslRequest mechanism= "GSSAPI OTP PLAIN"/>
```

* (ascii asterix, hex "2A") – This signals the authentication server to send a <SaslResponse> message listing its supported SASL mechanisms. This form of the <SaslRequest> message is sometimes referred to as the "server SASL mechanisms request" message. The <SaslRequest> message SHOULD be devoid of element content in this case. See Section 6.2.1 for details on the returned <SaslResponse> message in this case. Example 3 illustrates this form of the <SaslRequest> message.

Example 3. <SaslRequest> Message Requesting the Server's List of Supported SASL Mechanisms

```xml
<SaslRequest mechanism= "/*"/>
```

### 6.2. The <SaslResponse> Message

Figure 2 shows the <SaslResponse> message. This message is sent by the authentication server in response to a client <SaslRequest> message. The content model of the <SaslRequest> message is "any". This means that it may contain literally anything: base64-encoded data, arbitrary XML, ascii character data, the latest King Crimson tune in FLAC format [Schema1]. The data formats allowable in a <SaslRequest> message are defined in Section 7.

**Figure 2. <SaslResponse> Element - Schema Fragment**

```xml
<xsd:schema name="LibUtilitySchema" xmlns="http://xml.liberty alliance.org/urn:libutility"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="SaslResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Status"/>
        <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

The <Status> element is used to convey the authentication server's assessment of the status of the authentication exchange to the client. The code element is materialized by inclusion of [LibUtilitySchema] in [SOAPAuthnSchema]

**Figure 3. <Status> Element and Type - Schema Fragment (from lib-arch-utility.xsd)**

```xml
<xsd:element name="Status" type="Statustype"/>
```

```xml
<xsd:complexType name="Statustype">
  <xsd:sequence>
    <xsd:element ref="Status" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```
6.2.1. Returning the Server’s Supported SASL Mechanisms

If a client requests the server return a list of its supported SASL mechanisms (by using the "*" value for the mechanism attribute of the <SaslRequest> message; see Section 6.1.1). The returned mechanism names are typically taken from the column labeled as "MECHANISMS" in [SASLReg], but one or more of them MAY be site-specific. The status code value will be "ok". The SASL mechanism names are listed as whitespace-separated element content in plain ascii text, i.e. CDATA [Schema1]. See Example 4 for an example of such a response.

Note

This is the only case where a <SaslResponse> message’s element content MUST NOT be base64-encoded.

Example 4. <SaslResponse> Listing Server’s Supported SASL Mechanisms

```xml
<SaslResponse>
  <Status code="ok"/>
  GSSAPI KERBEROS_V4 EXTERNAL CRAM-MD5 ANONYMOUS SECURID
</SaslResponse>
```

6.2.2. Values for the code attribute of <Status>

If the value of code is:

- "continue" – the client is expected to craft a new <SaslRequest> message containing data requisite for whatever step it is at, in the particular SASL mechanism being negotiated. In this case the incoming <SaslResponse> message MAY contain SASL mechanism challenge data in base64 encoded form.

- "abort" – the server is aborting the authentication exchange. It will not send any more messages on this message thread.

- "success" – the server considers the authentication process to have completed successfully. The <SaslResponse> will contain a security token, exact form of which is defined in [LibertySecMech], that facilitates the client to make a discovery request subsequent to completing the authentication exchange.
7. Concrete SASL Profile

Section 4 of [RFC2222] requires the following information be specified by protocol definitions profiling SASL. Refer to this specification’s Section 6 (above) for definition of the details of the messages themselves.

The items below constitute the normative definition of the sequencing of the authentication exchange defined by this specification.

• **Service name**: "soap"

• **Initiation sequence**: The sequence MAY optionally begin with the client sending a "server SASL mechanisms request" message, as defined in Section 6.1.1. If so, the server responds with a `<SaslResponse>` message as described in Section 6.2.1. Then the authentication exchange MAY proceed as described below in the paragraphs under the Completion of Initiation Sequence: header, OR the client MAY choose to simply not send any further messages in this message thread, effectively ending this authentication exchange.

Alternatively to the above paragraph, the sequence MAY begin with the client sending a "client SASL mechanisms request" message, as defined in Section 6.1.1. In this message, the client lists the set of SASL mechanisms the client supports and is prepared to negotiate. In this case, the server responds with a `<SaslResponse>` message as described in Section 6.2.1 listing one of (1) the same set of SASL mechanisms as listed by the client, or (2) a subset of the SASL mechanisms the client listed, or (3) a null SASL mechanism list.

In cases (1) and (2), the client selects a mechanism from the list provided by the server, and then proceeds with the negotiation as described below in the paragraphs under the Completion of Initiation Sequence: header, below. D1 In case (3), the server is indicating that it doesn’t support, or is not prepared to negotiate, any of the SASL mechanisms proposed by the client. The client MUST cease sending messages on this message thread – the negotiation is terminated. **Completion of Initiation Sequence:**

• A `<SaslRequest>` message is sent by the party acting as the client to the party acting as the authentication server, specifying the SASL mechanism the client wishes to employ in authenticating. The optional element content of the `<SaslRequest>` element corresponds to the optional "initial response" that MAY be sent by a SASL client. See section 5.1 of [RFC2222].

In this case, the server MUST remember the specified value of the mechanism attribute of the `<SaslRequest>` message and associate it with this message thread.

• **Exchange sequence and authentication**: Subsequent to the initiation sequence, the authentication server responds with a `<SaslResponse>` message. If required by the employed SASL mechanism, and if the server did not indicate either completion or an error, the client sends another `<SaslRequest>` message, with content as dictated by the SASL mechanism being negotiated, and the mechanism set to the same value as it was in the initiation sequence. This sequence of `<SaslRequest>` followed by `<SaslResponse>` continues until the server indicates completion or failure, or the client aborts. The `<status>` element of the `<SaslResponse>` element is used by the authentication server to convey the status of the exchange to the client, and to provide further information about failures.

The server will signal the final `<SaslResponse>` message by setting the code attribute of the `<Status>` to "success", and returning credentials sufficient for interaction with, and targeted for, the Discovery Service.

**Note**

After the initiation sequence, and during the exchange sequence, either the client or the server MAY terminate the authentication exchange via the means specified below:
• **Client-initiated exchange termination:** The client MUST either send a `<SaslRequest>` message with a mechanism attribute value different than the prior such message, or with a value of "."

• **Server-initiated exchange termination:** The server MUST return a status code value of "abort" on a `<SaslResponse>` message, as defined in Section 6.2.2.

• **Security layer negotiation:** Security layer negotiation is **DISALLOWED**. The client and the server both MUST NOT initiate security layer negotiation in any case.

• **Use of the authorization identity:** An authorization identity is asserted from the first successful authentication exchange completion until the client successfully engages in a subsequent authentication exchange. The authorization identity is derived from the authentication credentials supplied by the server.

**Example 5. A SaslRequest Bound into a SOAP Message**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:sb="urn:liberty:wsf:soap-bind:1.0"
  xmlns:idpp="rn:liberty:idpp:1.0">
  <s:Header>
    <sb:Correlation id="A13454...245"
      mustUnderstand="1"
      actor="http://schemas.../next"
      messageID="uuid:efefefef-aaaa-ffff-cccc-eeefefefbabb"
      timestamp="2112-03-15T11:12:12Z"/>
  </s:Header>
  <s:Body>
    <SaslRequest Mechanism="foo">
      qwyGHsNwpjQu5yq......vUUvONmOZtfgzFz
    </SaslRequest>
  </s:Body>
</s:Envelope>
```
8. Examples

Note

TODO: A plethora of examples of various authentication exchanges, and aborted authentication exchanges goes here.
9. Security Considerations

Note

TODO: A plethora of security considerations go here. For the most part they will be a large intersection of the security considerations 'sections of the Internet-Draft draft-nystrom-http-sasl-06.txt and RFC 2251 & 2830 of LDAPv3.
10. Acknowledgements

This spec leverages techniques and ideas from draft-nystrom-http-sasl-06, RFC3080, RFC2251, RFC2830, et al. The authors of those specs are gratefully acknowledged. Also, thanks to Brad, Bob Mould, Weather Report, Liz Phair, The Wallflowers, Alan Holdsworth, Chick Corea, Jennifer Trynin, The Cowboy Junkies, Fugazi, Blues Traveler, Blink-182, CSN, Pearl Jam, plus many others for keeping me company whilst I groveled over the keyboard.

References

Normative


Error: no bibliography entry: LibUtilitySchema found in ../../../../document/refs/refs.xml


Error: no bibliography entry: SOAPAuthnSchema found in ../../../../document/refs/refs.xml


Informational

A. Listing of Simple Authentication and Security Layer (SASL) Mechanisms

Ref: [SASLReg]

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores. SASL mechanism names must be registered with the IANA.

<table>
<thead>
<tr>
<th>MECHANISM</th>
<th>USAGE</th>
<th>REFERENCE</th>
<th>OWNER</th>
</tr>
</thead>
<tbody>
<tr>
<td>KERBEROS_V4</td>
<td>LIMITED</td>
<td>[RFC2222]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>GSSAPI</td>
<td>COMMON</td>
<td>[RFC2222]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>SKEY</td>
<td>OBSOLETE</td>
<td>[RFC2444]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>EXTERNAL</td>
<td>COMMON</td>
<td>[RFC2222]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>CRAM-MD5</td>
<td>LIMITED</td>
<td>[RFC2195]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>ANONYMOUS</td>
<td>COMMON</td>
<td>[RFC2245]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>OTP</td>
<td>COMMON</td>
<td>[RFC2444]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>GSS-SPNEGO</td>
<td>LIMITED</td>
<td>[RFC2444]</td>
<td>Paul Leach <a href="mailto:paulle@microsoft.com">paulle@microsoft.com</a></td>
</tr>
<tr>
<td>PLAIN</td>
<td>COMMON</td>
<td>[RFC2595]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>SECURID</td>
<td>COMMON</td>
<td>[RFC2808]</td>
<td>Magnus Nyström <a href="mailto:magnus@rsasecurity.com">magnus@rsasecurity.com</a></td>
</tr>
<tr>
<td>NTLM</td>
<td>LIMITED</td>
<td>[Leach]</td>
<td>Paul Leach <a href="mailto:paulle@microsoft.com">paulle@microsoft.com</a></td>
</tr>
<tr>
<td>NMAS_LOGIN</td>
<td>LIMITED</td>
<td>[Gayman]</td>
<td>Mark G. Gayman <a href="mailto:mgayman@novell.com">mgayman@novell.com</a></td>
</tr>
<tr>
<td>NMAS_AUTHEM</td>
<td>LIMITED</td>
<td>[Gayman]</td>
<td>Mark G. Gayman <a href="mailto:mgayman@novell.com">mgayman@novell.com</a></td>
</tr>
<tr>
<td>DIGEST-MD5</td>
<td>COMMON</td>
<td>[RFC2831]</td>
<td>IESG <a href="mailto:iesg@ietf.org">iesg@ietf.org</a></td>
</tr>
<tr>
<td>9798-U-RSA-SHA1-ENC</td>
<td>COMMON</td>
<td>[RFC3163]</td>
<td>Robert <a href="mailto:Zuccherato@entrust.com">Zuccherato@entrust.com</a></td>
</tr>
<tr>
<td>9798-M-RSA-SHA1-ENC</td>
<td>COMMON</td>
<td>[RFC3163]</td>
<td>Robert <a href="mailto:Zuccherato@entrust.com">Zuccherato@entrust.com</a></td>
</tr>
<tr>
<td>9798-U-DSA-SHA1</td>
<td>COMMON</td>
<td>[RFC3163]</td>
<td>Robert <a href="mailto:Zuccherato@entrust.com">Zuccherato@entrust.com</a></td>
</tr>
</tbody>
</table>
B. Schema Listing

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    targetNamespace="urn:liberty:sas:2003-08"
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:sa="urn:liberty:sas:2003-08"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:liberty:sas:2003-08"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="02"
>
<xs:annotation>
  <xs:documentation>
  Liberty Alliance Project:
  Liberty SASL-based SOAP Authentication Specification
  </xs:documentation>
</xs:annotation>

<xs:schema version="1.0-03">
  <xs:annotation>
    # Notice #
    Copyright (c) 2003
  </xs:annotation>

  <xs:element name="SaslRequest">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:base64Binary">
          <xs:attribute name="mechanism" type="xs:string" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="SaslResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Status" />
        <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
</xs:complexType>
</xs:element>

</xs:schema>