Network Working Group
Request for Comments: 4743
Category: Standards Track

T. Goddard
ICEsoft Technologies Inc.

December 2006

Using NETCONF over the Simple Object Access Protocol (SOAP)

Status of This Memo

Copyright Notice

Abstract

The Network Configuration Protocol (NETCONF) is applicable to a wide range of devices in a variety of environments. Web Services is one such environment and is presently characterized by the use of the Simple Object Access Protocol (SOAP). NETCONF finds many benefits in this environment: from the reuse of existing standards, to ease of software development, to integration with deployed systems. Herein, we describe SOAP over HTTP and SOAP over Blocks Exchange Extensible Protocol (BEEP) bindings for NETCONF.

Table of Contents

1. Introduction

Given the use of Extensible Markup Language (XML) [2] and the remote procedure call characteristics, it is natural to consider a binding of the NETCONF [1] operations to a SOAP [3] application protocol. This document proposes a binding of this form.

In general, SOAP is a natural messaging scheme for NETCONF, essentially because of the remote procedure call character of both. However, care must be taken with SOAP over HTTP as it is inherently synchronous and client-driven. SOAP over BEEP [11] is technically superior, but is not as widely adopted.

Four basic topics are presented: SOAP specifics of interest to NETCONF, specifics on implementing NETCONF as a SOAP-based web service, security considerations, and functional Web Services

Description Language (WSDL) definitions. In some sense, the most important part of the document is the brief WSDL document presented in Section 3.7. With the right tools, the WSDL combined with the base NETCONF XML Schemas provides machine-readable descriptions sufficient for the development of software applications using NETCONF.

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 [8].

2. SOAP Background for NETCONF

Why introduce SOAP as yet another wrapper around what is already a remote procedure call message? There are, in fact, both technical and practical reasons. The technical reasons are perhaps less compelling, but let’s examine them first.

The use of SOAP does offer a few technical advantages. SOAP is fundamentally an XML messaging scheme (which is capable of supporting remote procedure call), and it defines a simple message format composed of a "header" and a "body" contained within an "envelope". The "header" contains meta-information relating to the message and can be used to indicate such things as store-and-forward behaviour or transactional characteristics. In addition, SOAP specifies an optional encoding for the "body" of the message. However, this encoding is not applicable to NETCONF as one of the goals is to have highly readable XML, and SOAP-encoding is optimized instead for ease of automated de-serialization. These benefits of the SOAP message structure are simple, but worthwhile because they are already standardized.

It is the practical reasons that truly make SOAP an interesting choice for device management. It is not difficult to invent a mechanism for exchanging XML messages over TCP, but what is difficult is getting that mechanism supported in a wide variety of tools and operating systems and having that mechanism understood by a great many developers. SOAP over HTTP (with WSDL) is seeing good success at this, and this means that a device management protocol making use of these technologies has advantages in being implemented and adopted. Admittedly, there are interoperability problems with SOAP and WSDL, but such problems have wide attention and can be expected to be resolved.

2.1. Use and Storage of WSDL and XSD

One of the advantages of using machine-readable formats (such as Web Services Description Language (WSDL) [16] and XML Schemas [4]) is that they can be used automatically in the software development process. With appropriate tools, WSDL and XSD can be used to generate classes that act as remote interfaces or application-specific data structures. Other uses, such as document generation and service location, are also common. A great innovation found with many XML-based definition languages is the use of hyperlinks for referring to documents containing supporting definitions.

```
<import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
        location="http://www.iana.org/assignments/xml-registry/
                  schema/netconf.xsd" />
```

For instance, in WSDL, the above import statement imports the definitions of XML types and elements from the base NETCONF schema. Ideally, the file containing that schema is hosted on a web server under the authority of the standards body that defined the schema. In this way, dependent standards can be built up over time, and all are accessible to automated software tools that ensure adherence to the standards. The IANA-maintained registry for this purpose is described in "The IETF XML Registry" [13].

Note that WSDL declarations for SOAP over BEEP bindings are not yet standardized.

2.2. SOAP over HTTP

Although SOAP focuses on messages and can be bound to different underlying protocols such as HTTP, SMTP, or BEEP, most existing SOAP implementations support only HTTP or HTTP/TLS.

There are a number of advantages to considering SOAP over protocols other than HTTP, as HTTP assigns the very distinct client and server roles by connection initiation. This causes difficulties in supporting asynchronous notification and can be relieved in many ways by replacing HTTP with BEEP.

2.3. HTTP Drawbacks

HTTP is not the ideal transport for messaging, but it is adequate for the most basic interpretation of "remote procedure call". HTTP is based on a communication pattern whereby the client (which initiates the TCP connection) makes a "request" to the server. The server returns a "response", and this process is continued (possibly over a

persistent connection, as described below). This matches the basic idea of a remote procedure call where the caller invokes a procedure on a remote server and waits for the return value.

Potential criticisms of HTTP could include the following:

- Server-initiated data flow is awkward to provide.
- Headers are verbose and text-based
- Idle connections may be closed by intermediate proxies
- Data encapsulation must adhere to Multipurpose Internet Mail Extensions (MIME) [15].
- Bulk transfer relies on stream-based ordering.

In many ways, these criticisms are directed at particular compromises in the design of HTTP. As such, they are important to consider, but it is not clear that they result in fatal drawbacks for a device management protocol.

2.4. BCP56: On the Use of HTTP as a Substrate

Best Current Practice 56 [6] presents a number of important considerations on the use of HTTP in application protocols. In particular, it raises the following concerns:

- HTTP may be more complex than is necessary for the application.
- The use of HTTP may mask the application from some firewalls.
- A substantially new service should not reuse port 80 as assigned to HTTP.
- HTTP caching may mask connection state.

Fundamentally, these concerns lie directly with common usage of SOAP over HTTP, rather than the application of SOAP over HTTP to NETCONF. As BCP 56 indicates, it is debatable whether HTTP is an appropriate protocol for SOAP at all, and it is likely that BEEP would be a superior protocol for most SOAP applications. Unfortunately, SOAP over HTTP is in common use and must be supported if the practical benefits of SOAP are to be realized. Note that the verbose nature of SOAP actually makes it more readily processed by firewalls, albeit firewalls designed to process SOAP messages.

HTTP caches SHOULD NOT be inserted between NETCONF managers and agents as NETCONF session state is tied to the state of the underlying transport connection. Three defensive actions can be taken:

- Caching MUST be prohibited through the use of HTTP headers Cache-Control and Pragma: no-cache.
- HTTP proxies SHOULD NOT be deployed within the management network.
- Use HTTPS.

It is also possible to respond to the concern on the reuse of port 80. Any NETCONF SOAP service MUST always be supported over the new standard port for NETCONF over SOAP, and all conforming implementations MUST default to attempting connections over this new standard port for NETCONF. A standard port for NETCONF over SOAP (over HTTP) has been assigned in the IANA considerations of this document.

2.5. Important HTTP 1.1 Features

HTTP 1.1 [5] includes two important features that provide for relatively efficient transport of SOAP messages. These features are "persistent connections" and "chunked transfer-coding".

Persistent connections allow a single TCP connection to be used across multiple HTTP requests. This permits multiple SOAP request/response message pairs to be exchanged without the overhead of creating a new TCP connection for each request. Given that a single stream is used for both requests and responses, it is clear that some form of framing is necessary. For messages whose length is known in advance, this is handled by the HTTP header "Content-length". For messages of dynamic length, "Chunking" is required.

HTTP "Chunking" or "chunked transfer-coding" allows the sender to send an indefinite amount of binary data. This is accomplished by informing the receiver of the size of each "chunk" (substring of the data) before the chunk is transmitted. The last chunk is indicated by a chunk of zero length. Chunking can be effectively used to transfer a large XML document where the document is generated on-line from a non-XML form in memory.

In terms of its application to SOAP message exchanges, persistent connections are clearly important for performance reasons and are particularly important when the persistence of authenticated connections is at stake. When one considers that messages of dynamic length are the rule rather than the exception for SOAP messages, it

is also clear that Chunking is very useful. In some cases, it is possible to buffer a SOAP response and determine its length before sending, but the storage requirements for this are prohibitive for many devices. Together, these two features provide a good foundation for device management using SOAP over HTTP. HTTP chunking and persistent connections [5] SHOULD be used.

2.6. SOAP over BEEP

Although not widely adopted by the Web Services community, BEEP is an excellent substrate for SOAP [12]. In particular, it provides for request/response message exchanges initiated by either BEEP peer and allows the number of response messages to be arbitrary (including zero). The BEEP profile for SOAP simply makes use of a single BEEP channel for exchanging SOAP messages and benefits from BEEP's inherent strengths for message exchange over a single transport connection.

2.7. SOAP Implementation Considerations

It is not the goal of this document to cover the SOAP [3] specification in detail. Instead, we provide a few comments that may be of interest to an implementor of NETCONF over SOAP.

2.7.1. SOAP Feature Exploitation

NETCONF over SOAP does not make extensive use of SOAP features. For instance, NETCONF operations are not broken into SOAP message parts, and the SOAP header is not used to convey <rpc> metadata. This is a deliberate design decision as it allows the implementor to provide NETCONF over multiple substrates easily while handling the messages over those different substrates in a common way.

2.7.2. SOAP Headers

Implementers of NETCONF over SOAP should be aware of the following characteristic of SOAP headers: a SOAP header may have the attribute "mustUnderstand", and, if it does, the recipient must either process the header block or not process the SOAP message at all, and instead generate a fault. A "mustUnderstand" header must not be silently discarded.

In general, however, SOAP headers are intended for application-specific uses. The NETCONF SOAP binding does not make use of SOAP headers.

2.7.3. SOAP Faults

A SOAP Fault is returned in the event of a NETCONF <rpc-error>. It is constructed essentially as a wrapper for the <rpc-error>, but it allows SOAP processors to propagate the <rpc-error> to application code using a language-appropriate exception mechanism.

A SOAP Fault is constructed from an <rpc-error> as follows: the SOAP Fault Code Value is "Receiver" in the SOAP envelope namespace, the SOAP Fault Reason Text is the contents of the NETCONF <rpc-error> "error-tag", and the SOAP Fault detail is the original <rpc-error> structure.

For instance, given the following <rpc-error>,

```
<rpc-error xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <error-type>rpc</error-type>
  <error-tag>MISSING_ATTRIBUTE</error-tag>
  <error-severity>error</error-severity>
  <error-info>
    <bad-attribute>message-id</bad-attribute>
    <bad-element>rpc</bad-element>
  </error-info>
</rpc-error>
```

the associated SOAP Fault message is

```
<soapenv:Envelope
    xmlns:soapenv=
      "http://www.w3.org/2003/05/soap-envelope"
    xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <soapenv:Body>
    <soapenv:Fault>
      <soapenv:Code>
        <soapenv:Value>env:Receiver</soapenv:Value>
      </soapenv:Code>
      <soapenv:Reason>
        <soapenv:Text
            xml:lang="en">MISSING_ATTRIBUTE</soapenv:Text>
      </soapenv:Reason>
      <detail>
        <rpc-error xmlns=
            "urn:ietf:params:xml:ns:netconf:base:1.0">
          <error-type>rpc</error-type>
          <error-tag>MISSING_ATTRIBUTE</error-tag>
          <error-severity>error</error-severity>
          <error-info>
            <bad-attribute>message-id</bad-attribute>
```

```
                <bad-element>rpc</bad-element>
              </error-info>
            </rpc-error>
          </detail>
        </soapenv:Fault>
      </soapenv:Body>
    </soapenv:Envelope>
```

3. A SOAP Service for NETCONF

3.1. Fundamental Use Case

The fundamental use case for NETCONF over SOAP is that of a management console ("manager" role) managing one or more devices running NETCONF agents ("agent" role). The manager initiates an HTTP or BEEP connection to an agent and drives the NETCONF session via a sequence of SOAP messages. When the manager closes the connection, the NETCONF session is also closed.

3.2. NETCONF Session Establishment

A NETCONF over SOAP session is established by the initial message exchange on the underlying substrate. For HTTP, a NETCONF session is established once a SOAP message is POSTed to the NETCONF web application URI. For BEEP, a NETCONF session is established once the BEEP profile for SOAP handshake establishes the SOAP channel.

3.3. NETCONF Capabilities Exchange

Capabilities exchange and session ID establishment are performed through the exchange of <hello> messages. In the case of SOAP over HTTP, the HTTP client MUST send the first <hello> message. The case of SOAP over BEEP imposes no ordering constraints. For instance, the following example shows the exchange of <hello> messages and establishes a session ID value of 4. Observe that the management client initiates the exchange and the server agent assigns the session ID.

```
C: POST /netconf HTTP/1.1
C: Host: netconfdevice
C: Content-Type: text/xml; charset=utf-8
C: Accept: application/soap+xml, text/*
C: Cache-Control: no-cache
C: Pragma: no-cache
C: Content-Length: 376
C:
C: <?xml version="1.0" encoding="UTF-8"?>
C: <soapenv:Envelope
C:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
C:   <soapenv:Body>
C:     <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:       <capabilities>
C:         <capability>
C:           urn:ietf:params:netconf:base:1.0
C:         </capability>
C:       </capabilities>
C:     </hello>
C:   </soapenv:Body>
C: </soapenv:Envelope>
S: HTTP/1.1 200 OK
S: Content-Type: application/soap+xml; charset=utf-8
S: Content-Length: 600
S:
S: <?xml version="1.0" encoding="UTF-8"?>
S: <soapenv:Envelope
S:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
S:   <soapenv:Body>
S:     <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:       <capabilities>
S:         <capability>
S:           urn:ietf:params:netconf:base:1.0
S:         </capability>
S:         <capability>
S:           urn:ietf:params:netconf:capability:startup:1.0
S:         </capability>
S:         <capability>
S:           http:/example.net/router/2.3/myfeature
S:        </capability>
S:       </capabilities>
S:       <session-id>4</session-id>
S:     </hello>
S:   </soapenv:Body>
S: </soapenv:Envelope>
```

3.4. NETCONF Session Usage

NETCONF sessions are persistent for both performance and semantic reasons. NETCONF session state contains the following:

1. Authentication Information
2. Capability Information
3. Locks
4. Pending Operations
5. Operation Sequence Numbers

Authentication must be maintained throughout a session due to the fact that it is expensive to establish. Capability Information is maintained so that appropriate operations can be applied during a session. Locks are released upon termination of a session as this makes the protocol more robust. Pending operations come and go from existence during the normal course of remote procedure call (RPC) operations. Operation sequence numbers provide the small but necessary state information to refer to operations during the session.

In the case of SOAP over HTTP, a NETCONF session is supported by an HTTP connection with an authenticated user. For SOAP over BEEP, a NETCONF session is supported by a BEEP channel operating according to the BEEP profile for SOAP [12].

3.5. NETCONF Session Teardown

To allow automated cleanup, NETCONF over SOAP session teardown takes place when the underlying connection (in the case of HTTP) or channel (in the case of BEEP) is closed. Note that the root cause of such teardown may be the closure of the TCP connection under either HTTP or BEEP as the case may be. NETCONF managers and agents must be capable of programatically closing the transport connections associated with NETCONF sessions, such as in response to a <close-session> operation; thus, the HTTP or BEEP substrate implementation must expose this appropriately.

3.6. A NETCONF over SOAP Example

Since the proposed WSDL (in Section 3.7) uses document/literal encoding, the use of a SOAP header and body has little impact on the representation of a NETCONF operation. This example shows HTTP/1.1 for simplicity. An example for BEEP would be similar.

```
C: POST /netconf HTTP/1.1
C: Host: netconfdevice
C: Content-Type: text/xml; charset=utf-8
C: Accept: application/soap+xml, text/*
C: Cache-Control: no-cache
C: Pragma: no-cache
C: Content-Length: 465
C:
C: <?xml version="1.0" encoding="UTF-8"?>
C: <soapenv:Envelope
C:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
C:   <soapenv:Body>
C:     <rpc message-id="101"
C:          xmlns="xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:       <get-config>
C:         <filter type="subtree">
C:           <top xmlns="http://example.com/schema/1.2/config">
C:             <users/>
C:           </top>
C:         </filter>
C:       </get-config>
C:     </rpc>
C:   </soapenv:Body>
C: </soapenv:Envelope>
```

The HTTP/1.1 response is also straightforward:

```
S: HTTP/1.1 200 OK
S: Content-Type: application/soap+xml; charset=utf-8
S: Content-Length: 917
S:
S: <?xml version="1.0" encoding="UTF-8"?>
S: <soapenv:Envelope
S:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
S:   <soapenv:Body>
S:     <rpc-reply message-id="101"
S:                xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:       <data>
S:         <top xmlns="http://example.com/schema/1.2/config">
S:           <users>
S:             <user>
S:               <name>root</name>
S:               <type>superuser</type>
S:               <full-name>Charlie Root</full-name>
S:                 <dept>1</dept>
S:                 <id>1</id>
S:               </company-info>
S:             </user>
```

```
S:             <user>
S:               <name>fred</name>
S:               <type>admin</type>
S:               <full-name>Fred Flintstone</full-name>
S:                 <dept>2</dept>
S:                 <id>2</id>
S:               </company-info>
S:             </user>
S:           </users>
S:         </top>
S:       </data>
S:     </rpc-reply>
S:   </soapenv:Body>
S: </soapenv:Envelope>
```

3.7. NETCONF SOAP WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="urn:ietf:params:xml:ns:netconf:soap:1.0"
  xmlns:netb="urn:ietf:params:xml:ns:netconf:base:1.0"
  targetNamespace="urn:ietf:params:xml:ns:netconf:soap:1.0"
  name="netconf-soap_1.0.wsdl">

  <import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
          location="http://www.iana.org/assignments/xml-registry/
                    schema/netconf.xsd" />

  <message name="helloRequest">
    <part name="in" element="netb:hello"/>
  </message>
  <message name="helloResponse">
    <part name="out" element="netb:hello"/>
  </message>

  <message name="rpcRequest">
    <part name="in" element="netb:rpc"/>
  </message>
  <message name="rpcResponse">
    <part name="out" element="netb:rpc-reply"/>
  </message>

  <portType name="netconfPortType">
    <operation name="rpc">
      <input message="tns:rpcRequest"/>
      <output message="tns:rpcResponse"/>
```

```
      </operation>
      <operation name="hello">
        <input message="tns:helloRequest"/>
        <output message="tns:helloResponse"/>
      </operation>
    </portType>

    <binding name="netconfBinding" type="tns:netconfPortType">
      <SOAP:binding style="document"
           transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="hello">
        <SOAP:operation/>
        <input>
          <SOAP:body use="literal"
               namespace="urn:ietf:params:xml:ns:netconf:soap:1.0"/>
        </input>
        <output>
          <SOAP:body use="literal"
               namespace="urn:ietf:params:xml:ns:netconf:soap:1.0"/>
        </output>
      </operation>
      <operation name="rpc">
        <SOAP:operation/>
        <input>
          <SOAP:body use="literal"
               namespace="urn:ietf:params:xml:ns:netconf:base:1.0"/>
        </input>
        <output>
          <SOAP:body use="literal"
               namespace="urn:ietf:params:xml:ns:netconf:base:1.0"/>
        </output>
      </operation>
    </binding>

  </definitions>
```

3.8. Sample Service Definition WSDL

The following WSDL document assumes a local location for the NETCONF over SOAP WSDL definitions. A typical deployment of a device manageable via NETCONF over SOAP would provide a service definition similar to the following to identify the address of the device.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:nets="urn:ietf:params:xml:ns:netconf:soap:1.0"
```

```
    targetNamespace="urn:myNetconfService"
    name="myNetconfService.wsdl">

    <import namespace="urn:ietf:params:xml:ns:netconf:soap:1.0"
            location="http://localhost:8080/netconf/
                      schema/netconf-soap_1.0.wsdl"/>

    <service name="netconf">
      <port name="netconfPort" binding="nets:netconfBinding">
        <SOAP:address location="http://localhost:8080/netconf"/>
      </port>
    </service>

  </definitions>
```

4. Security Considerations

NETCONF is used to access and modify configuration information, so the ability to access this protocol should be limited to users and systems that are authorized to view or modify the agent's configuration data.

Because configuration information is sent in both directions, it is not sufficient for just the client or user to be authenticated with the server. The identity of the server should also be authenticated with the client.

Configuration data may include sensitive information, such as user names or security keys. So, NETCONF should only be used over communications channels that provide strong encryption for data privacy.

If the NETCONF server provides remote access through insecure protocols, such as HTTP, care should be taken to prevent execution of the NETCONF program when strong user authentication or data privacy is not available.

The IANA assigned port SHOULD be used, as this provides a means for efficient firewall filtering during possible denial-of-service attacks.

4.1. Integrity, Privacy, and Authentication

The NETCONF SOAP binding relies on an underlying secure transport for integrity and privacy. Such transports are expected to include TLS [9] (which, when combined with HTTP, is referred to as HTTPS) and IPsec. There are a number of options for authentication (some of which are deployment-specific):

- within the transport (such as with TLS client certificates)
- within HTTP (such as Digest Access Authentication [7])
- within SOAP (such as a digital signature in the header [17])

HTTP, BEEP, and SOAP level authentication can be integrated with Remote Authentication Dial-In User Service (RADIUS) [10] to support remote authentication databases.

At a miniumum, all conforming NETCONF over SOAP implementations MUST support TLS. Specifically, NETCONF over SOAP over HTTP MUST support NETCONF over SOAP over HTTPS, and NETCONF over SOAP over BEEP MUST support NETCONF over SOAP over BEEP over TLS.

4.2. Vulnerabilities

The above protocols may have various vulnerabilities, and these may be inherited by NETCONF over SOAP.

NETCONF itself may have vulnerabilities because an authorization model is not currently specified.

It is important that device capabilities and authorization remain constant for the duration of any outstanding NETCONF session. In the case of NETCONF, it is important to consider that device management may be taking place over multiple substrates (in addition to SOAP), and it is important that the different substrates have a common authentication model.

4.3. Environmental Specifics

Some deployments of NETCONF over SOAP may choose to use transports without encryption. This presents vulnerabilities but may be selected for deployments involving closed networks or debugging scenarios.

A device managed by NETCONF may interact (over protocols besides NETCONF) with devices managed by other protocols, all of differing security. Each point of entry brings with it a potential vulnerability.

5. IANA Considerations

IANA assigned TCP port (833) for NETCONF over SOAP over BEEP, and TCP port (832) for NETCONF over SOAP over HTTPS.

IANA will allow for the assignment of an XML namespace within the NETCONF namespace "urn:ietf:params:xml:ns:netconf" for the NETCONF over SOAP WSDL definitions. Following the policies outlined in RFC 2434 [14], assigned values in this subordinate namespace are requested to be allocated according to the "Specification Required" policy.

URI: urn:ietf:params:xml:ns:netconf:soap

6. References

6.1. Normative References

[1] Enns, R., Ed., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[2] Bray, T., Paoli, J., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C REC REC-xml-20001006, October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>.

[3] Gudgin, M., Hadley, M., Moreau, JJ., and H. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation REC-soap12-part1-20030624, June 2002, <http://www.w3.org/TR/soap12-part1/>.

[4] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures", W3C Recommendation REC-xmlschema-1-20010502, May 2001, <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.

[5] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[6] Moore, K., "On the use of HTTP as a Substrate", RFC 3205, February 2002.

[7] Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[8] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[9] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[10] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[11] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

[12] O'Tuathail, E. and M. Rose, "Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP)", RFC 4227, January 2006.

[13] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.

[14] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

6.2. Informative References

[15] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[16] Christensen, E., Curbera, F., Meredith, G., and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note NOTE-wsdl-20010315, March 2001, <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>.

[17] Brown, A., Fox, B., Hada, S., LaMacchia, B., and H. Maruyama, "SOAP Security Extensions: Digital Signature", W3C Note NOTE-SOAP-dsig-20010206, Feb 2001, <http://www.w3.org/TR/SOAP-dsig/>.

Author's Address

Ted Goddard
ICEsoft Technologies Inc.
Suite 300, 1717 10th St. NW
Calgary, AB T2M 4S2
Canada

Phone: (403) 663-3322
EMail: ted.goddard@icesoft.com
URI: http://www.icesoft.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.