netconf
Internet-Draft
Expires: April 15, 2004

T. Goddard
Wind River Systems
October 16, 2003

NETCONF Over SOAP
draft-ietf-netconf-soap-00

Status of this Memo

Copyright Notice

Abstract

The configuration protocol NETCONF is applicable to a wide range of devices in a variety of environments. The emergence of Web Services gives one such environment, and is presently characterized by the use of SOAP over HTTP. NETCONF finds many benefits in this environment: from the use of existing standards, to ease of software development, to integration with deployed systems. Herein, we describe a SOAP over HTTP binding that, when used with multiple persistent HTTP connections, yields an application protocol sufficient for NETCONF.

Table of Contents

1. Introduction

Given the use of XML [1] and the remote procedure call characteristics, it is natural to consider a binding of the NETCONF [13] operations to a SOAP [2] application protocol. This document proposes a binding of this form.

Note that a SOAP binding for NETCONF is not necessarily intended only for managing individual devices. For instance, a server providing a SOAP interface can act as a proxy for multiple devices, possibly connecting to those devices over BEEP [16] or serial lines. In this case it is important to define a data model that appropriately aggregates the devices.

In general, SOAP over HTTP is a natural application protocol for NETCONF (essentially because both emphasize remote procedure calls) but there are three areas that require care: the <rpc-progress> operation, the mechanism for aborting operations, and the notification channel. The reason for this is that all of these functions are asynchronous (from the point of view of the manager) and HTTP is inherently synchronous and client-driven.

Four basic topics are presented: SOAP specifics of interest to NETCONF, specifics on implementing NETCONF as a SOAP-based web service, security considerations, and an appendix with functional WSDL. In some sense, the most important part of the document is the brief WSDL document presented in the Appendix. With the right tools, the WSDL combined with the base NETCONF XML Schemas provide machine readable descriptions sufficient for the development of software applications using NETCONF.

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.

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 deserialization. These benefits of the SOAP message structure are basic, but worthwhile due to the fact that they are already standardized.

It is the practical reasons that make SOAP over HTTP 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) [3] 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. For instance, in WSDL, the import statement

```
<import namespace="http://iana.org/netconf/1.0/base"
        location="http://iana.org/netconf/1.0/base.xsd"/>
```

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. Thus, it will gradually become as important for iana.org to host documents like

http://iana.org/netconf/1.0/base/base.xsd

as the IETF now hosts documents such as

http://www.ietf.org/rfc/rfc2616.txt

2.2 SOAP over HTTP

While it is true that 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. For this discussion we will assume SOAP over HTTP or HTTP/TLS unless otherwise specified. (This also includes applications of IPSec to SOAP over HTTP.)

Note that there are a number of advantages to considering SOAP over protocols other than HTTP, as HTTP assigns its very distinct client and server roles by connection initiation. This causes difficulties in supporting asynchronous notification (possibly relieved by replacing SOAP/HTTP with SOAP/BEEP). However, it is also the case that the full potential of HTTP is not currently used by SOAP. For instance, multiple SOAP replies to a single request could be contained in a multipart MIME [6] response. This would be a similar strategy to the use of multipart/related with SOAP attachments [14].

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:

o server-initiated data flow is awkward

- o headers are verbose and text-based
- o idle connections may be closed by intermediate proxies
- o data encapsulation must adhere to MIME
- o 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 Important HTTP 1.1 Features

HTTP 1.1 [7] 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 application to SOAP message exchanges, persistent connections are clearly important for performance reasons, and are particularly important when it is the persistence of authenticated connections that 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.

3. A SOAP Web Service for NETCONF

3.1 Fundamental Use Case

The fundamental use case for NETCONF over SOAP (NETCONF/SOAP) over HTTP is that of a management console ("manager" role) managing one or more devices running NETCONF agents ("agent" role). The manager initiates one or more HTTP connections to the agent and drives the NETCONF sessions through repeated SOAP messages over HTTP requests. When the manager closes all HTTP connections associated with a session, the NETCONF session is also closed.

3.2 Mapping NETCONF Channels to HTTP Connections

While the transport of SOAP over BEEP [17] has been specified, the purpose of this discussion is to describe how to map the channel semantics and performance characteristics already assumed by NETCONF onto (possibly persistent) SOAP over HTTP connections. This configuration is chosen because it is the one that benefits most from existing SOAP tools and implementations. It is true that BEEP has many advantages over HTTP for the transport of SOAP messages, but the fact remains that HTTP is currently more widely deployed than BEEP. At some point in the future, NETCONF/SOAP over BEEP may also be of interest. At that time it can be easily dealt with as many of the issues already discussed in this document are pertinent. There would simply be a few enhancements regarding asynchronous notification.

NETCONF employs potentially three channels per session: the management channel, the operation channel, and the notification channel. In the SOAP over HTTP binding, each of these channels can be mapped to an individual HTTP connection (although the notification channel may be a BEEP channel in a separate TCP connection). Thus, SOAP messages on one connection (corresponding to the management channel) must be able to refer to SOAP messages on another connection (corresponding to the operation channel) as the "session" is potentially spread across multiple TCP connections. For instance, it may be necessary to abort a time-extended SOAP request on the "operation" HTTP connection by sending an "<rpc-abort>" message on the "management" HTTP connection.

Distinct "operation" and "management" HTTP connections are not defined; the agent may limit the number of HTTP connections in the same session, and each is capable those "management" and "operation" procedure calls supported by NETCONF over SOAP.

3.2.1 Asynchronous Functionality

NETCONF uses two types of asynchronous functionality, and the mapping

of these onto SOAP over HTTP is somewhat problematic. The two asynchronous functions are <rpc-progress> and notifications on the notification channel, and these are not supported in the SOAP over HTTP application protocol. Instead, the client can periodically poll the appropriate elements of via <get-state> (on a secondary HTTP connection) to obtain progress information or notification log entries.

Additionally, the notification mechanism for NETCONF is specified in an existing standard for reliable syslog [12] and it is suggested that the same mechanism be used with the SOAP binding (it is simply external). If notifications via SOAP over HTTP are desired, it is probably most effective if an HTTP connection is established from the agent to the management console. Such a connection could be established in response to the manager connecting to the device. More sophisticated functionality, such as multiple SOAP replies to a single request, would require enhancements to the SOAP over HTTP specification.

3.3 NETCONF Sessions

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 RPC operations. Operation sequence numbers provide the small but necessary state information to refer to operations during the session.

Since it is generally not possible to support a full NETCONF session with a single HTTP connection, it is necessary to identify the NETCONF session in a way that can span multiple HTTP connections. This can be performed with the HTTP request URI, as in the following POST request with the target session "sid-123":

```
POST /netconf/sid-123 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Content-Length: 470
```

Note that the session identifier must either be known by the manager (in order to attach to an existing session) or be communicated from the agent to the manager prior to the exchange of any significant NETCONF messages. For this, it is recommended that the session identifier be determined via <get-state>. An empty session identifier may be used in the case where only an operations channel is required (in this case the agent assigns a new session to that HTTP connection).

Thus, in the case of SOAP over HTTP, a NETCONF "session" is a collection of HTTP connections with common authenticated users and a common session identifier as indicated in the HTTP reqest URI header. To support automated cleanup, a NETCONF over SOAP session is closed when all connections associated with that session are closed.

3.4 Capabilities Exchange

Capabilities exchange, if defined through a NETCONF RPC operation, can easily be accommodated in the SOAP binding.

3.5 A NETCONF/SOAP example

Since the proposed WSDL (in Appendix A.1) 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/1.0 for simplicity.

```
POST /netconf HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, text/*
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 470

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <rpc id="101" xmlns="http://ietf.org/netconf/1.0/base">
      <get-config>
        <source>
          <running/>
        </source>
        <config xmlns="http://example.com/schema/1.2/config">
```

```
            <users/>
          </config>
          <format>xml</format>
        </get-config>
      </rpc>
    </soapenv:Body>
  </soapenv:Envelope>
```

The HTTP/1.0 response is also straightforward:

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <rpc-reply id="101" xmlns="http://ietf.org/netconf/1.0/base">
      <config xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
            <type>superuser</type>
          </user>
          <user>
            <name>fred</name>
            <type>admin</type>
          </user>
          <user>
            <name>barney</name>
            <type>admin</type>
          </user>
        </users>
      </config>
    </rpc-reply>
  </soapenv:Body>
</soapenv:Envelope>
```

3.6 Managing Multiple Devices

When a server is acting as a proxy for multiple devices, the URL for the HTTP POST can be used to indicate which device is the target. It may also be desirable to use the HTTP POST URL as a means for selecting from multiple virtual devices on a single device.

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.

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 [10] 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 [8])
- within SOAP (such as a digital signature in the header [15])

HTTP and SOAP level authentication can be integrated with RADIUS [11] to support remote authentication databases.

4.2 Vulnerabilities

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

NETCONF itself may have vulnerabilities due to the fact that 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/SOAP, this constancy must be given particular attention as a session may span multiple HTTP connections.

4.3 Environmental Specifics

Some deployments of NETCONF/SOAP may choose to use HTTP 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 other than NETCONF) with devices managed by other protocols, all of differing security. Each point of entry brings with it a potential vulnerability.

Normative References

[1] 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>.

[2] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S. and D. Winer, "Simple Object Access Protocol (SOAP) 1.1", W3C Note NOTE-SOAP-20000508, May 2000, <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>.

[3] 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>.

[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] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, <http://www.ietf.org/rfc/rfc2045.txt>.

[6] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996, <http://www.ietf.org/rfc/rfc2046.txt>.

[7] 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, <http://www.ietf.org/rfc/rfc2616.txt>.

[8] Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E. and L. Stewart, "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997, <http://www.ietf.org/rfc/rfc2069.txt>.

[9] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.

[10] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999, <http://www.ietf.org/rfc/rfc2246.txt>.

[11] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000, <http://www.ietf.org/rfc/rfc2865.txt>.

[12] Rose, M. and D. New, "Reliable Delivery for syslog", RFC 3195, November 2001, <http://www.ietf.org/rfc/rfc3195.txt>.

Informative References

[13] Enns, R., "NETCONF Configuration Protocol", draft-ietf-netconf-prot-00 (work in progress), Aug 2003, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-00.txt>.

[14] Barton, J., Nielsen, H. and S. Thatte, "SOAP Messages with Attachments", W3C Note NOTE-SOAP-attachments-20001211, Dec 2000, <http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211>.

[15] 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/2001/NOTE-SOAP-dsig-20010206/>.

[16] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001, <http://www.ietf.org/rfc/rfc3080.txt>.

[17] O'Tuathail, E. and M. Rose, "Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP)", RFC 3288, June 2002, <http://www.ietf.org/rfc/rfc3288.txt>.

Author's Address

Ted Goddard
Wind River Systems
#180, 6815-8th Street NE
Calgary, AB T2E 7H7
Canada

Phone: (403) 730-7590
EMail: ted.goddard@windriver.com
URI: http://www.windriver.com

Appendix A. WSDL Definitions

A.1 NETCONF SOAP Binding

The following WSDL document assumes a hypothetical location for the NETCONF schema.

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

  <import namespace="http://ietf.org/netconf/1.0/base"
          location="base.xsd"/>

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

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

  <binding name="rpcBinding" type="tns:rpcPortType">
    <SOAP:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="rpc">
      <SOAP:operation/>
      <input>
        <SOAP:body use="literal"
             namespace="http://ietf.org/netconf/1.0/base"/>
      </input>
      <output>
        <SOAP:body use="literal"
             namespace="http://ietf.org/netconf/1.0/base"/>
      </output>
    </operation>
  </binding>
```

```
</definitions>
```

A.2 Sample Service Definition

The following WSDL document assumes a hypothetical location for the NETCONF/SOAP WSDL definitions. A typical deployment of a device manageable via NETCONF/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:xs="http://ietf.org/netconf/1.0/soap"
  targetNamespace="urn:myNetconfService"
  name="myNetconfService.wsdl">

  <import namespace="http://ietf.org/netconf/1.0/soap"
          location="soap.wsdl"/>

  <service name="netconf">
    <port name="rpcPort" binding="xs:rpcBinding">
      <SOAP:address location="http://localhost:8080/netconf"/>
    </port>
  </service>

</definitions>
```

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

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