SPPP Over SOAP and HTTP
draft-ietf-drinks-sppp-over-soap-04

Abstract

The Session Peering Provisioning Protocol (SPPP) is an XML protocol that exists to enable the provisioning of session establishment data into Session Data Registries or SIP Service Provider data stores. Sending XML data structures over Simple Object Access Protocol (SOAP) and HTTP(s) is a widely used, de-facto standard for messaging between elements of provisioning systems. Therefore the combination of SOAP and HTTP(s) as a transport for SPPP is a natural fit. The obvious benefits include leveraging existing industry expertise, leveraging existing standards, and a higher probability that existing provisioning systems can be more easily integrated with this protocol. This document describes the specification for transporting SPPP XML structures over SOAP and HTTP(s).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on January 7, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
Table of Contents

1. Introduction .............................................. 3
2. Terminology ........................................... 4
3. SOAP Features and SPPP .............................. 5
4. HTTP(s) Features and SPPP ......................... 7
5. Authentication and Session Management .......... 8
6. SPPP SOAP WSDL Definition .......................... 9
7. Security Considerations ............................... 12
   7.1. Integrity, Privacy, and Authentication .......... 12
   7.2. Vulnerabilities .................................. 12
   7.3. Deployment Environment Specifics ............. 12
8. IANA Considerations .................................. 13
9. Acknowledgements .................................... 14
10. Normative References ............................... 15

Author’s Address ........................................ 16
1. Introduction

SPPP, defined in [I-D.draft-ietf-drinks-spprov], is best supported by a transport and messaging infrastructure that is connection oriented, request-response oriented, easily secured, supports propagation through firewalls in a standard fashion, and that is easily integrated into back-office systems. This is due to the fact that the client side of SPPP is likely to be integrated with organizations’ operational support systems that facilitate transactional provisioning of user addresses and their associated session establishment data. While the server side of SPPP is likely to reside in a separate organization’s network, such that SPPP provisioning transactions will be required to traverse the Internet as they are propagated from the SPPP client to the SPPP server. Given the current state of industry practice and technologies, SOAP and HTTP(s) are well suited for this type of environment. This document describes the specification for transporting SPPP XML structures over SOAP and HTTP(s).

The specification in this document for transporting SPPP XML structures over SOAP and HTTP(s) is primarily comprised of five subjects: (1) a description of any applicable SOAP features, (2) any applicable HTTP features, (3) security considerations, and perhaps most importantly, (5) the Web Services Description Language (WSDL) definition for SPPP over SOAP.
2. Terminology

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].
3. SOAP Features and SPPP

The list of SOAP features that are explicitly used and required for SPPP are limited. Most SOAP features are not necessary for SPPP. SPPP primarily uses SOAP simply as a standard message envelope technology. The SOAP message envelope is comprised of the SOAP header and body. As described in the SOAP specifications, the SOAP header can contain optional, application specific, information about the message. The SOAP body contains the SPPP message itself, whose structure is defined by the combination of one of the WSDL operations defined in this document and the SPPP XML data structures defined in the SPPP protocol document. SPPP does not rely on any data elements in the SOAP header. All relevant data elements are defined in the SPPP XML schema described in [I-D.draft-ietf-drinks-spprov] and the SPPP WSDL specification described in this document.

WSDL is a widely standardized and adopted technology for defining the top-level structures of the messages that are transported within the body of a SOAP message. The WSDL definition for the SPPP SOAP messages is defined later in this document, which imports by reference the XML data types contained in the SPPP schema. The IANA registry where the SPPP schema resides is described in The IETF XML Registry [RFC3688].

There are multiple structural styles that SOAP WSDL allows. But the best practice for this type of application is what is sometimes referred to as the Document Literal Wrapped style of designing SOAP WSDL. This style is generally regarded as an optimal approach that enhances maintainability, comprehension, portability, and, to a certain extent, performance. It is characterized by setting the soapAction binding style as _document_, the soapAction encoding style as _literal_, and then defining the SOAP messages to simply contain a single data element that _wraps_ that is a data structure containing all the required input or output data elements. The figure below illustrates this high level technical structure.
The SOAP operations supported by SPPP are normatively defined later in this document. Each SOAP operation defines a request/input message and a response/output message. Each such request and response message then contains a single object that wraps the SPPP XML data types that comprise the inputs and the outputs, respectively, of the SOAP operation.

SOAP faults are not used by the SPPP SOAP mapping. All SPPP success and error responses are specified within the SPPP protocol specification [I-D.draft-ietf-drinks-spprov].

SOAP 1.2 [SOAPREF] or higher and WSDL 1.1 [WSDLREF] or higher SHOULD be used.
4. HTTP(s) Features and SPPP

SOAP is not tied to HTTP(s), however, for reasons described in the introduction, HTTP(s) is a good choice as the transport mechanism for the SPPP SOAP messages. HTTP 1.1 includes the "persistent connection" feature, which allows multiple HTTP request/response pairs to be transported across a single HTTP connection. This is an important performance optimization feature, particularly when the connections is an HTTPS connection where the relatively time consuming SSL handshake has occurred. Persistent connections SHOULD be used for the SPPP HTTP connections.

HTTP 1.1 [RFC2616] or higher SHOULD be used.
5. Authentication and Session Management

All SOAP and HTTP SPPP Clients and Servers MUST support Transport Layer Security (TLS) as defined in [RFC5246] as the secure transport mechanism. All SOAP SPPP Clients and Servers MUST use HTTP Digest Authentication as defined in [RFC2617] as the secure authentication mechanism. As a result, the communication session is established as a result of the initial HTTP connection setup, the digest authentication, handshake, and the TLS handshake. When the HTTP connection is broken down, the communication session ends.
6. SPPP SOAP WSDL Definition

The SPPP WSDL is defined below. The WSDL design approach is commonly referred to as _Generic WSDL_. It is generic in the sense that there is not a specific WSDL operation defined for each object type that is supported by the SPPP protocol. There is a single WSDL update operation called submitUpdateRqst, and a single WSDL query operation called submitQueryRqst. The submitUpdateRqst operation takes as input an spppUpdateRequestMsg object and returns as output an spppUpdateResponseMsg object. These objects _wrap_ the spppUpdateRequest and spppUpdateResponse objects respectively. These two object data structures are described in the SPPP protocol specification [I-D.ietf-drinks-spprov]. And finally, the spppSOAPBinding in the WSDL defines the binding style as _document_ and the encoding as _literal_. It is this combination of _wrapped_ input and output data structures, _document_ binding style, and _literal_ encoding that characterize the Document Literal Wrapped style of WSDL specifications.

The advantage of generic WSDL is that the WSDL is more succinct, much simpler, and therefore more easily maintained. As new types of protocol objects and actions are added into or removed from the SPPP protocol, the WSDL does not need to change. This approach is made possible by the fact that the SPP XML data types and supported actions are defined in the SPPP XML schema, not in the WSDL. As a result the supported actions do not need to be re-defined here inside the SPPP SOAP WSDL.

Note: The following WSDL has been formatted (e.g., tabs, spaces) to meet I-D requirements.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:sppps="urn:ietf:params:xml:ns:sppp:soap:1"
 targetNamespace="urn:ietf:params:xml:ns:sppp:soap:1"
 xsi:schemaLocation="spppbase.xsd">
  <wsdl:types>
    <xsd:schema>
                   schemaLocation="spppbase.xsd"/>
    </xsd:schema>
  </wsdl:types>
</wsdl:definitions>
```
<wsdl:message name="spppUpdateRequestMsg">
  <wsdl:part name="rqst" element="spppb:spppUpdateRequest"/>
</wsdl:message>

<wsdl:message name="spppUpdateResponseMsg">
  <wsdl:part name="rspns" element="spppb:spppUpdateResponse"/>
</wsdl:message>

<wsdl:message name="spppQueryRequestMsg">
  <wsdl:part name="rqst" element="spppb:spppQueryRequest"/>
</wsdl:message>

<wsdl:message name="spppQueryResponseMsg">
  <wsdl:part name="rspns" element="spppb:spppQueryResponse"/>
</wsdl:message>

<wsdl:message name="spppServerStatusRequestMsg">
  <wsdl:part name="rqst" element="spppb:spppServerStatusRequest"/>
</wsdl:message>

<wsdl:message name="spppServerStatusResponseMsg">
  <wsdl:part name="rspns" element="spppb:spppServerStatusResponse"/>
</wsdl:message>

<wsdl:portType name="spppPortType">
  <wsdl:operation name="submitUpdateRqst">
    <wsdl:input message="sppps:spppUpdateRequestMsg"/>
    <wsdl:output message="sppps:spppUpdateResponseMsg"/>
  </wsdl:operation>

  <wsdl:operation name="submitQueryRqst">
    <wsdl:input message="sppps:spppQueryRequestMsg"/>
    <wsdl:output message="sppps:spppQueryResponseMsg"/>
  </wsdl:operation>

  <wsdl:operation name="submitServerStatusRqst">
    <wsdl:input message="sppps:spppServerStatusRequestMsg"/>
    <wsdl:output message="sppps:spppServerStatusResponseMsg"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="spppSoapBinding" type="sppps:spppPortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <soap:operation name="submitUpdateRqst" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>

  <wsdl:operation name="submitQueryRqst" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>

  <wsdl:operation name="submitServerStatusRqst" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
Figure 2: WSDL
7. Security Considerations

SPPP is used to query and update session peering data and addresses, so the ability to access this protocol should be limited to users and systems that are authorized to query and update this data. Because this data 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 by the client. This data may include sensitive information, routing data, lists of resolvable addresses, etc. So when used in a production setting and across non-secure networks, SPPP should only be used over communications channels that provide strong encryption for data privacy.

7.1. Integrity, Privacy, and Authentication

The SPPP SOAP binding relies on an underlying secure transport for integrity and privacy. Such transports are expected to include TLS/HTTPS. In addition to the application level authentication imposed by an SPPP server, there are a number of options for authentication within the transport layer and the messaging envelope. These include TLS client certificates, HTTP Digest Access Authentication, and digital signatures within SOAP headers.

At a minimum, all conforming SPPP over SOAP implementations MUST support HTTPS.

7.2. Vulnerabilities

The above protocols may have various vulnerabilities, and these may be inherited by SPPP over SOAP. And SPPP itself may have vulnerabilities because an authorization model is not explicitly specified in the current specification.

It is important that SPPP implementations implement an authorization model that considers the source of each SPPP query or update request and determines whether it is reasonable to authorize that source to perform that specific query or update.

7.3. Deployment Environment Specifics

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

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688].

URN assignments are requested: urn:ietf:params:xml:ns:sppp:soap
9. Acknowledgements

This document is a result of various discussions held by the DRINKS design team, which is comprised of the following individuals, in no specific order: Syed Ali (NeuStar), Sumanth Channabasappa (Cable Labs), David Schwartz (XConnect), Jean-Francois Mule (CableLabs), Kenneth Cartwright (TNS, Inc.), Manjul Maharishi (TNS, Inc.), Alexander Mayrhofer (enum.at GmbH).
10. Normative References

[I-D.draft-ietf-drinks-spprov]


Author’s Address

Kenneth Cartwright
TNS
1939 Roland Clarke Place
Reston, VA 20191
USA

Email: kcartwright@tnsi.com