DRINKS
Internet-Draft
Intended status: Standards Track
Expires: October 24, 2014

K. Cartwright
V. Bhatia
TNS
J-F. Mule
CableLabs
A. Mayrhofer
enum.at GmbH
April 22, 2014

Session Peering Provisioning (SPP) Protocol over SOAP
draft-ietf-drinks-spp-protocol-over-soap-06

Abstract

The Session Peering Provisioning Framework (SPPF) specifies the data model and the overall structure to provision session establishment data into Session Data Registries and SIP Service Provider data stores. To utilize this framework one needs a transport protocol. Given that Simple Object Access Protocol (SOAP) is currently widely used for messaging between elements of such provisioning systems, this document specifies the usage of SOAP (via HTTPS) as the transport protocol for SPPF. The benefits include leveraging prevalent expertise, and a higher probability that existing provisioning systems will be able to easily migrate to using an SPPF based protocol.

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 October 24, 2014.

Copyright Notice

Copyright (c) 2014 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 (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Terminology 4
3. SOAP Features and Protocol Layering 4
4. HTTP(s) Features and SPP Protocol over SOAP 7
5. Authentication, Integrity and Confidentiality 7
6. Language Identification 7
7. SPP Protocol SOAP Data Structures 7
7.1. Concrete Object Key Types 8
7.1.1. Generic Object Key 8
7.1.2. Public Identity Object Key 9
7.1.3. SED Group Offer Key 10
7.2. Operation Request and Response Structures 10
7.2.1. Add Operation Structure 11
7.2.2. Delete Operation Structure 14
7.2.3. Accept Operation Structure 17
7.2.4. Reject Operation Structure 20
7.2.5. Batch Operation Structure 23
7.2.6. Get Operation Structure 26
7.2.7. Get SED Group Offers Operation Structure 28
7.2.8. Generic Query Response 29
7.2.9. Get Server Details Operation Structure 30
7.3. Response Codes and Messages 31
8. Protocol Operations 34
9. SPP Protocol over SOAP WSDL Definition 34
10. SPP Protocol over SOAP Examples 45
10.1. Add Destination Group 46
10.2. Add SED Records 47
10.3. Add SED Records -- URIType 49
10.4. Add SED Group 50
10.5. Add Public Identity -- Successful COR claim 52
10.6. Add LRN 54

10.7. Add TN Range 55
10.8. Add TN Prefix 56
10.9. Enable Peering -- SED Group Offer 58
10.10. Enable Peering -- SED Group Offer Accept 59
10.11. Add Egress Route 60
10.12. Remove Peering -- SED Group Offer Reject 62
10.13. Get Destination Group 63
10.14. Get Public Identity 65
10.15. Get SED Group Request 66
10.16. Get SED Group Offers Request 68
10.17. Get Egress Route 70
10.18. Delete Destination Group 72
10.19. Delete Public Identity 73
10.20. Delete SED Group Request 74
10.21. Delete SED Group Offers Request 75
10.22. Delete Egress Route 77
10.23. Batch Request 78
11. Security Considerations 80
11.1. Vulnerabilities 81
12. IANA Considerations 81
13. Acknowledgements 81
14. References 82
14.1. Normative References 82
14.2. Informative References 82
Authors' Addresses 83

1. Introduction

SPPF, defined in [I-D.draft-ietf-drinks-spp-framework], 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 SPPF 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 SPPF is likely to reside in a separate organization's network, resulting in the SPPF provisioning transactions traversing the Internet as they are propagated from the SPPF client to the SPPF 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 SPPF XML structures over SOAP and HTTP(S).

The specification in this document for transporting SPPF 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, (4) the Web Services Description Language (WSDL) definition for SPP Protocol over SOAP, and (5) "transport" specific XML Schema type definitions

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 Protocol Layering

The list of SOAP features that are explicitly used and required for SPP Protocol over SOAP are limited. Most SOAP features are not necessary for SPPF. SPP Protocol over SOAP 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 [SOAPREF], the SOAP header can contain optional, application specific, information about the message. The SOAP body contains the SPPF message itself, whose structure is defined by the combination of one of the WSDL operations defined in this document and the SPPF XML data structures defined in this document and the SPPF document. SPPF does not rely on any data elements in the SOAP header. All relevant data elements are defined in the SPPF XML schema described in [I-D.draft-ietf-drinks-spp-framework] and the SPPF WSDL types specification described in this document and in [I-D.draft-ietf-drinks-spp-framework].

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 SPPF SOAP messages is defined later in this document, which imports by reference the XML data types contained in the SPPF schema. The IANA registry where the SPPF schema resides is described in the IETF XML Registry [RFC3688].

There are multiple structural styles that WSDL allows. The best practice for this type of application is what is sometimes referred to as the "document/literal wrapped style". 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" a data structure containing all the required input or

output data elements. The figure below illustrates this high level technical structure as conceptual layers 3 through 6.

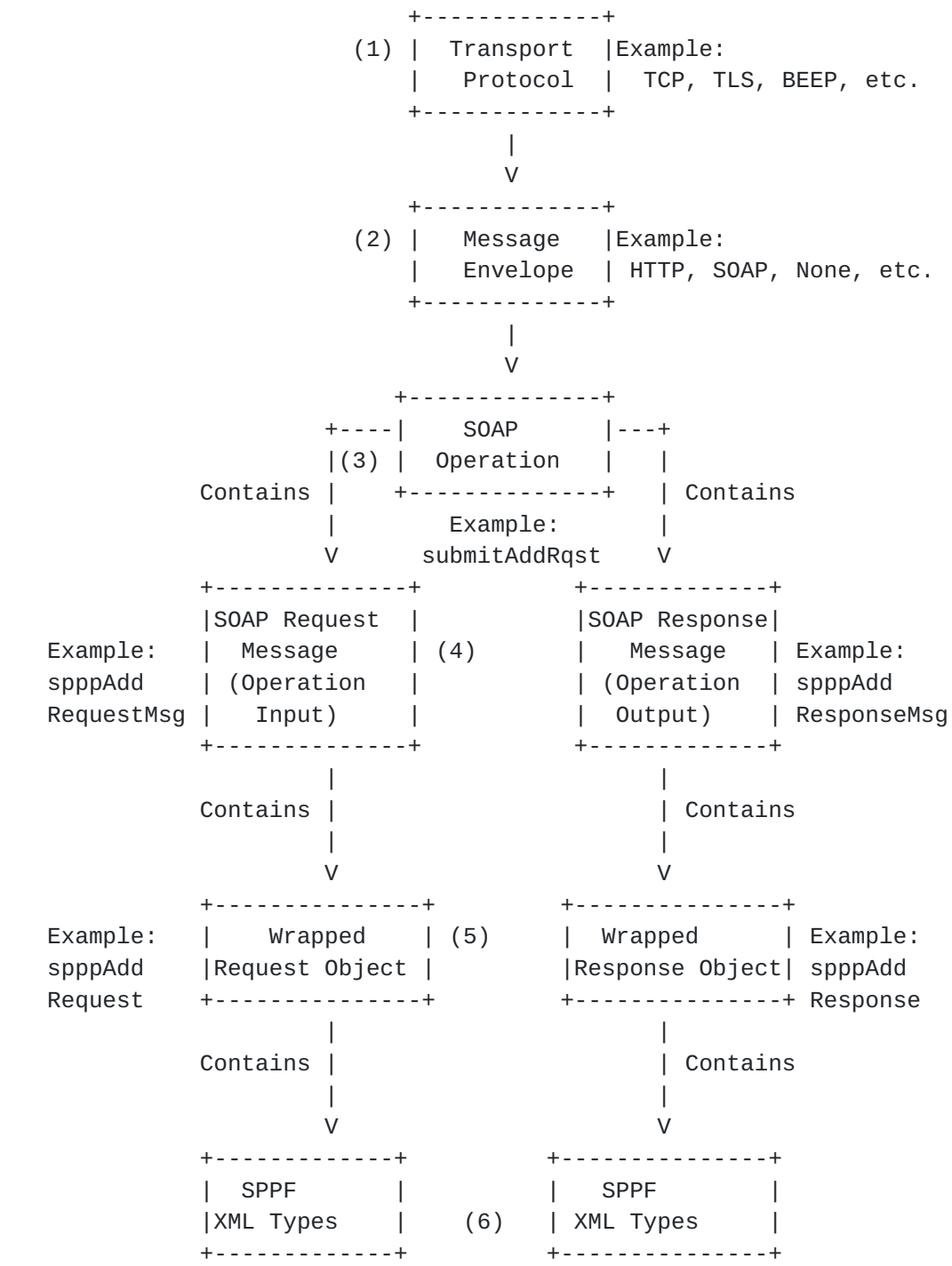


Figure 1: Layering and Technical Structure of the SPP Protocol over SOAP Messages

The operations supported by SPP Protocol over SOAP 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 SPPF XML data types that comprise the inputs and the outputs, respectively, of the SOAP operation.

SOAP faults are not used by the SPP Protocol over SOAP. All success and error responses are specified in Section 7.3 of this document. However, if a SOAP fault were to occur, perhaps due to failures in the SOAP message handling layer of a SOAP library, the client application should capture and handle the fault. Specifics on how to handle such SOAP faults, if they should occur, will be specific to the chosen SOAP implementation.

This document RECOMMENDS SOAP 1.2 [SOAPREF] or higher, and WSDL 1.1 [WSDLREF] or higher.

SPPF is a request/reply framework that allows a client application to submit provisioning data and query requests to a server. The SPPF data structures are designed to be protocol agnostic. Concerns regarding encryption, non-repudiation, and authentication are beyond the scope of this document. For more details, please refer to Section 4 ("Transport Protocol Requirements") of [I-D.draft-ietf-drinks-spp-framework].

As illustrated in the previous diagram, SPPF can be viewed as a set of layers that collectively define the structure of an SPPF request and response. Layers 1 and 2 represent the transport, envelope, and authentication technologies. This document defines layers 3, 4, 5, and 6 for SPP Protocol over SOAP.

1. Layer 1: The transport protocol layer represents the communication mechanism between the client and server. SPPF can be layered over any transport protocol that provides a set of basic requirements defined in Section 4 of [I-D.draft-ietf-drinks-spp-framework].

2. Layer 2: The message envelope layer is optional, but can provide features that are above the transport technology layer but below the application messaging layer. Technologies such as HTTP and SOAP are examples of messaging envelope technologies.

3. Layers 3,4,5,6: The operation and message layers provide an envelope-independent and transport-independent wrapper for the SPPF data model objects that are being acted on (created, modified, queried).

4. HTTP(s) Features and SPP Protocol over SOAP

While SOAP is not tied to HTTP(S), for reasons described in the introduction, HTTP(S) is a good choice as the transport mechanism for the SPP Protocol 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.

Implementations compliant with this document MUST use HTTP 1.1 [RFC2616] or higher. Also, implementations SHOULD use persistent connections.

5. Authentication, Integrity and Confidentiality

To accomplish authentication, conforming SPP Protocol over SOAP Clients and Servers MUST use HTTP Digest Authentication as defined in [RFC2617].

To achieve integrity and privacy, conforming SPP Protocol over SOAP Clients and Servers MUST support Transport Layer Security (TLS) as defined in [RFC5246] as the secure transport mechanism.

6. Language Identification

Section 9 of [I-D.draft-ietf-drinks-spp-framework] requires transport protocols to provide a mechanism to transmit language tags together with human-readable messages. When conforming SPP Protocol SOAP servers use such tagging, the XML "lang" attribute ([W3C.REC-xml-20081126], Section 2.12) MUST be used. Clients MAY use the HTTP "Accept-Language" header field (see Section 14.4 of [RFC2616]) in order to indicate their language preference.

7. SPP Protocol SOAP Data Structures

SPP Protocol over SOAP uses a set of XML based data structures for all the supported operations and any parameters that those operations are applied to. As also mentioned earlier in this document, these XML structures are envelope-independent and transport-independent. Refer the "Protocol Operations" (Section 8) of this document for a description of all the operations that MUST be supported.

The following sections describe the definition all the XML data structures.

7.1. Concrete Object Key Types

Certain operations in SPPF require an object key that uniquely identifies the object(s) on which a given operation needs to be performed. SPPF defines the XML structure of the any such object key in an abstract manner and delegates the concrete representation to any conforming transport protocol. The following sub-sections define the various types of concrete object key types used in various operations in SPP Protocol over SOAP.

7.1.1. Generic Object Key

Most objects in SPP Protocol over SOAP are uniquely identified by the attributes in the generic object key (Refer Section 5.2.1 "Generic Object Key Type" of [I-D.draft-ietf-drinks-spp-framework] for details). The concrete XML representation of ObjKeyType is as below:

```
<complexType name="ObjKeyType">
 <complexContent>
  <extension base="sppfb:ObjKeyType">
   <sequence>
    <element name="rant" type="sppfb:OrgIdType"/>
    <element name="name" type="sppfb:ObjNameType"/>
    <element name="type" type="sppfs:ObjKeyTypeEnum"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

The ObjKeyType has the data elements as described below:

- rant: The identifier of the registrant organization that owns the object.

- name: The character string that contains the name of the object.

- type: The enumeration value that represents the type of SPPF object. For example, both a Destination Group and a SED Group can have the same name "TestObj" and be associated with same Registrant Id. Hence, to uniquely identify the object that represents a Destination Group with the name "TestObj", the type "DestGrp" must be specified when using this concrete ObjKeyType structure to identify the Destination Group "TestObj".

The object types in SPP Protocol over SOAP MUST adhere to the above definition of generic object key, and are defined as an enumeration in the XML data structure as follows:

```
 <simpleType name="ObjKeyTypeEnum">
   <restriction base="token">
     <enumeration value="SedGrp"/>
     <enumeration value="DestGrp"/>
     <enumeration value="SedRec"/>
     <enumeration value="EgrRte"/>
   </restriction>
 </simpleType>
```

7.1.2. Public Identity Object Key

Public Identity type objects can further be of various sub-types like a Telephone Number (TN), Routing Number (RN), TN Prefix, URI, or a TN Range and cannot be cleanly identified with the attributes in the generic ObjKeyType. The definition of PubIdKeyType is as below:

```
   <complexType name="PubIdKeyType">
    <complexContent>
     <extension base="sppfb:PubIdKeyType">
      <sequence>
       <element name="rant" type="sppfb:OrgIdType"/>
       <choice>
        <element name="number"
        type="sppfb:NumberType"/>
        <element name="range"
         type="sppfb:NumberRangeType"/>
        <element name="uri"
         type="anyURI"/>
       </choice>
      </sequence>
     </extension>
    </complexContent>
   </complexType>
```

The PubIdKeyType has data elements, as described below:

- rant: The identifier of the registrant organization that owns the object.

- number: An element of type NumberType (refer Section 12 of [I-D.draft-ietf-drinks-spp-framework]) that contains the value and type of a number .

- range: An element of type NumberRangeType (refer Section 12 of [I-D.draft-ietf-drinks-spp-framework]) that contains a range of numbers.

- uri: A value that represents a Public Identifier.

Any instance of PubIdKeyType MUST contain exactly one element from the following set of elements: "number", "range", "uri".

7.1.3. SED Group Offer Key

In addition to the attributes in the generic ObjKeyType, a SED Group Offer object is uniquely identified by the organization ID of the organization to whom an SED Group has been offered. The definition of SedGrpOfferKeyType is as below:

```
<complexType name="SedGrpOfferKeyType">
 <complexContent>
  <extension base="sppfb:SedGrpOfferKeyType">
   <sequence>
    <element name="sedGrpKey" type="sppfs:ObjKeyType"/>
    <element name="offeredTo" type="sppfb:OrgIdType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

The SedGrpOfferKeyType has the data elements as described below:

- sedGrpKey: Identifies the SED Group that was offered.

- offeredTo: The organization ID of the organization that was offered the SED Group object identified by the sedGrpKey.

7.2. Operation Request and Response Structures

An SPPF client interacts with an SPPF server by sending one or more requests to the server, and by receiving corresponding responses from the server. The basic set of operations that an SPPF client can submit to an SPPF server and the semantics of those operations are

defined in Section 7 ("Framework Operations") of [I-D.draft-ietf-drinks-spp-framework]. The following sub-sections describe the XML data structures that are used for each of those types of operations for a SPP Protocol over SOAP implementation.

7.2.1. Add Operation Structure

In order to add (or modify) an object in the registry, an authorized entity can send the spppAddRequest to the registry.

An SPP Protocol over SOAP Add request is wrapped within the <spppAddRequest> element while an SPP Protocol over SOAP Add response is wrapped within an <spppAddResponse> element. The following sub-sections describe the spppAddRequest and spppAddResponse elements. Refer to Section 10 for an example of Add operation on each type of SPPF object.

7.2.1.1. Add Request

An SPP Protocol over SOAP Add request definition is contained within the generic <spppAddRequest> element.

```
<element name="spppAddRequest">
 <complexType>
  <sequence>
   <element name="clientTransId"
    type="sppfb:TransIdType" minOccurs="0"/>
   <element name="minorVer"
    type="sppfb:MinorVerType" minOccurs="0"/>
   <element name="obj" type="sppfb:BasicObjType"
   maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>
```

The data elements within the <spppAddRequest> element are described as follows:

- clientTransId: Zero or one client-generated transaction ID that, within the context of the SPPF client, identifies this request. This value can be used at the discretion of the SPPF client to track, log or correlate requests and their responses. SPPF server MUST echo back this value to the client in the corresponding response to the incoming request. SPPF server will not check this value for uniqueness.

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the "Get Server Details" operation described in Section 7.2.9.

- obj: One or more elements of abstract type BasicObjType (defined in [I-D.draft-ietf-drinks-spp-framework]). Each element contains all the attributes of an SPPF object that that the client is requesting the SPPF server to add. Refer to section 3.1 of [I-D.draft-ietf-drinks-spp-framework] for the XML structure of all concrete types, for various SPPF objects, that extend from abstract BasicObjType and hence are eligible to be passed into this element. The elements are processed by the SPPF server in the order in which they are included in the request. With respect to handling of error conditions, conforming SPPP SOAP servers MUST stop processing BasicObjType elements in the request at the first error, and roll back any BasicObjType elements that had already been processed for that add request ("stop and rollback").

7.2.1.2. Add Response

An SPP Protocol over SOAP add response object is contained within the generic <spppAddResponse> element. This response structure is used for all types of SPPF objects that are provisioned by the SPPF client.

```
<element name="spppAddResponse">
  <complexType>
    <sequence>
      <element name="clientTransId" type="sppfb:TransIdType"
       minOccurs="0"/>
      <element name="serverTransId" type="sppfb:TransIdType"/>
      <element name="overallResult" type="sppfb:ResultCodeType"/>
      <element name="detailResult" type="sppfs:ObjResultCodeType"
      minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

<complexType name="ResultCodeType">
  <sequence>
     <element name="code" type="int"/>
     <element name="msg" type="string"/>
  </sequence>
</complexType>

 <complexType name="ObjResultCodeType">
  <complexContent>
   <extension base="sppfs:ResultCodeType">
    <sequence>
     <element name="obj" type="sppfb:BasicObjType"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

An <spppAddResponse> contains the elements necessary for the SPPF client to precisely determine the overall result of the request, and if an error occurred, it provides information about the specific object(s) that caused the error.

The data elements within the SPP Protocol over SOAP Add response are described as follows:

- clientTransId: Zero or one client transaction ID. This value is simply an echo of the client transaction ID that SPPF client passed into the SPPF update request. When included in the request, the SPPF server MUST return it in the corresponding response message.

- serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value MUST be unique for a given SPPF server.

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- detailResult: An optional response code, response message, and BasicObjType (as defined in [I-D.draft-ietf-drinks-spp-framework]) triplet. This element will be present only if an object level error has occurred. It indicates the error condition and the exact request object that contributed to the error. The response code will reflect the exact error. See Section 7.3 for further details.

7.2.2. Delete Operation Structure

In order to remove an object from the registry, an authorized entity can send the spppDelRequest into the registry. An SPP Protocol over SOAP Delete request is wrapped within the <spppDelRequest> element while a SPP Protocol over SOAP Delete response is wrapped within the generic <spppDelResponse> element. The following sub-sections describe the spppDelRequest and spppDelResponse elements. Refer to Section 10 for an example of Delete operation on each type of SPPF object.

7.2.2.1. Delete Request

An SPP Protocol over SOAP Delete request definition is contained within the generic <spppDelRequest> element.

```
<element name="spppDelRequest">
 <complexType>
  <sequence>
   <element name="clientTransId"
   type="sppfb:TransIdType" minOccurs="0"/>
   <element name="minorVer"
   type="sppfb:MinorVerType" minOccurs="0"/>
  <element name="objKey" type="sppfb:ObjKeyType"
   maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>
```

The data elements within the <spppDelRequest> element are described as follows:

- clientTransId: Zero or one client-generated transaction ID that, within the context of the SPPF client, identifies this request. This value can be used at the discretion of the SPPF client to track, log or correlate requests and their responses. SPPF server MUST echo back this value to the client in the corresponding response to the incoming request. SPPF server will not check this value for uniqueness.

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the Get Server Details Operation described in Section 7.2.9.

- objKey: One or more elements of abstract type ObjKeyType (as defined in [I-D.draft-ietf-drinks-spp-framework]). Each element contains attributes that uniquely identify the object that the client is requesting the server to delete. Refer to Section 7.1 for a description of all concrete object key types, for various SPPF objects, which are eligible to be passed into this element. The elements are processed by the SPPF server in the order in which they are included in the request. With respect to handling of error conditions, conforming SPPP SOAP servers MUST stop processing ObjKeyType elements in the request at the first error, and roll back any ObjKeyType elements that had already been processed for that delete request ("stop and rollback").

7.2.2.2. Delete Response

An SPP Protocol over SOAP delete response object is contained within the generic <sppDeleteResponse> element. This response structure is used for a delete request on all types of SPPF objects that are provisioned by the SPPF client.

```
<element name="spppDelResponse">
 <complexType>
  <sequence>
   <element name="clientTransId" type="sppfb:TransIdType"
            minOccurs="0"/>
   <element name="serverTransId" type="sppfb:TransIdType"/>
   <element name="overallResult" type="sppfb:ResultCodeType"/>
   <element name="detailResult" type="sppfs:ObjKeyResultCodeType"
            minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>

<complexType name="ResultCodeType">
 <sequence>
  <element name="code" type="int"/>
  <element name="msg" type="string"/>
 </sequence>
</complexType>

<complexType name="ObjKeyResultCodeType">
 <complexContent>
  <extension base="sppfs:ResultCodeType">
   <sequence>
    <element name="objKey" type="sppfb:ObjKeyType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

An <spppDelResponse> contains the elements necessary for the SPPF client to precisely determine the overall result of the request, and if an error occurred, it provides information about the specific object key(s) that caused the error.

The data elements within the SPP Protocol over SOAP Delete response are described as follows:

- clientTransId: Zero or one client transaction ID. This value is simply an echo of the client transaction ID that SPPF client passed into the SPPF update request. When included in the request, the SPPF server MUST return it in the corresponding response message.

- serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value MUST be unique for a given SPPF server.

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- detailResult: An optional response code, response message, and ObjKeyType (as defined in [I-D.draft-ietf-drinks-spp-framework]) triplet. This element will be present only if an specific object key level error has occurred. It indicates the error condition and the exact request object key that contributed to the error. The response code will reflect the exact error. See Section 7.3 for further details.

7.2.3. Accept Operation Structure

In SPPF, a SED Group Offer can be accepted or rejected by, or on behalf of, the registrant to whom the SED Group has been offered (refer Section 3.1 of [I-D.draft-ietf-drinks-spp-framework] for a description of the SED Group Offer object). The Accept operation is used to accept such SED Group Offers by, or on behalf of, the Registrant. The request structure for an SPP Protocol over SOAP Accept operation is wrapped within the <spppAcceptRequest> element while an SPP Protocol over SOAP Accept response is wrapped within the generic <spppAcceptResponse> element. The following sub-sections describe the spppAcceptRequest and spppAcceptResponse elements. Refer to Section 10 for an example of Accept operation on a SED Group Offer.

7.2.3.1. Accept Request Structure

An SPP Protocol over SOAP Accept request definition is contained within the generic <sppAcceptRequest> element.

```
<element name="spppAcceptRequest">
 <complexType>
  <sequence>
   <element name="clientTransId"
   type="sppfb:TransIdType" minOccurs="0"/>
   <element name="minorVer"
   type="sppfb:MinorVerType" minOccurs="0"/>
   <element name="sedGrpOfferKey"
   type="sppfs:SedGrpOfferKeyType"
   maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>
```

The data elements within the <spppAcceptRequest> element are described as follows:

- clientTransId: Zero or one client-generated transaction ID that, within the context of the SPPF client, identifies this request. This value can be used at the discretion of the SPPF client to track, log or correlate requests and their responses. SPPF server MUST echo back this value to the client in the corresponding response to the incoming request. SPPF server will not check this value for uniqueness.

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the Get Server Details Operation described in Section 7.2.9.

- sedGrpOfferKey: One or more elements of type SedGrpOfferKeyType (as defined in this document). Each element contains attributes that uniquely identify a SED Group Offer that the client is requesting the server to accept. The elements are processed by the SPPF server in the order in which they are included in the request. With respect to handling of error conditions, conforming SPPP SOAP servers MUST stop processing SedGrpOfferKeyType elements in the request at the first error, and roll back any SedGrpOfferKeyType elements that had already been processed for that accept request ("stop and rollback").

7.2.3.2. Accept Response

An SPP Protocol over SOAP accept response structure is contained within the generic <sppAcceptResponse> element. This response structure is used for an Accept request on a SED Group Offer.

```
<element name="spppAcceptResponse">
 <complexType>
  <sequence>
   <element name="clientTransId" type="sppfb:TransIdType"
            minOccurs="0"/>
   <element name="serverTransId" type="sppfb:TransIdType"/>
   <element name="overallResult" type="sppfb:ResultCodeType"/>
   <element name="detailResult"
            type="sppfs:SedGrpOfferKeyResultCodeType"
            minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>

<complexType name="ResultCodeType">
 <sequence>
  <element name="code" type="int"/>
  <element name="msg" type="string"/>
 </sequence>
</complexType>

<complexType name="SedGrpOfferKeyResultCodeType">
 <complexContent>
  <extension base="sppfs:ResultCodeType">
   <sequence>
    <element name="sedGrpOfferKey" type="sppfs:SedGrpOfferKeyType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

An <spppAcceptResponse> contains the elements necessary for the SPPF client to precisely determine the overall result of the request, and if an error occurred, it provides information about the specific SED Group Offer key(s) that caused the error.

The data elements within the SPP Protocol over SOAP Accept response are described as follows:

- clientTransId: Zero or one client transaction ID. This value is simply an echo of the client transaction ID that SPPF client passed into the SPPF update request. When included in the request, the SPPF server MUST return it in the corresponding response message.

- serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value MUST be unique for a given SPPF server.

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- detailResult: An optional response code, response message, and SedGrpOfferKeyType (as defined in this document) triplet. This element will be present only if any specific SED Group Offer key level error has occurred. It indicates the error condition and the exact request SED Group Offer key that contributed to the error. The response code will reflect the exact error. See Section 7.3 for further details.

7.2.4. Reject Operation Structure

In SPPF, SED Group Offer can be accepted or rejected by, or on behalf of, the registrant to whom the SED Group has been offered (refer "Framework Data Model Objects" section of [I-D.draft-ietf-drinks-spp-framework] for a description of the SED Group Offer object). The Reject operation is used to reject such SED Group Offers by, or on behalf of, the Registrant. The request structure for an SPP Protocol over SOAP Reject operation is wrapped within the <spppRejectRequest> element while an SPP Protocol over SOAP Reject response is wrapped within the generic <spppRejecResponse> element. The following sub-sections describe the spppRejectRequest and spppRejecResponse elements. Refer to Section 10 for an example of Reject operation on a SED Group Offer.

7.2.4.1. Reject Request

An SPP Protocol over SOAP Reject request definition is contained within the generic <spppRejectRequest> element.

```
<element name="spppRejectRequest">
 <complexType>
  <sequence>
   <element name="clientTransId"
   type="sppfb:TransIdType" minOccurs="0"/>
   <element name="minorVer"
   type="sppfb:MinorVerType" minOccurs="0"/>
   <element name="sedGrpOfferKey"
   type="sppfs:SedGrpOfferKeyType"
   maxOccurs="unbounded"/>
 </complexType>
</element>
```

The data elements within the <spppRejectRequest> element are described as follows:

- clientTransId: Zero or one client-generated transaction ID that, within the context of the SPPF client, identifies this request. This value can be used at the discretion of the SPPF client to track, log or correlate requests and their responses. SPPF server MUST echo back this value to the client in the corresponding response to the incoming request. SPPF server will not check this value for uniqueness.

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the Get Server Details Operation described in Section 7.2.9.

- sedGrpOfferKey: One or more elements of type SedGrpOfferKeyType (as defined in this document). Each element contains attributes that uniquely identify a SED Group Offer that the client is requesting the server to reject. The elements are processed by the SPPF server in the order in which they are included in the request. With respect to handling of error conditions, conforming SPPF servers MUST stop processing SedGrpOfferKeyType elements in the request at the first error, and roll back any SedGrpOfferKeyType elements that had already been processed for that reject request ("stop and rollback").

7.2.4.2. Reject Response

An SPP Protocol over SOAP reject response structure is contained within the generic <sppRejectResponse> element. This response structure is used for an Reject request on a SED Group Offer.

```
<element name="spppRejectResponse">
 <complexType>
  <sequence>
   <element name="clientTransId" type="sppfb:TransIdType"
            minOccurs="0"/>
   <element name="serverTransId" type="sppfb:TransIdType"/>
   <element name="overallResult" type="sppfb:ResultCodeType"/>
   <element name="detailResult"
            type="sppfs:SedGrpOfferKeyResultCodeType"
            minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>

<complexType name="ResultCodeType">
 <sequence>
  <element name="code" type="int"/>
  <element name="msg" type="string"/>
 </sequence>
</complexType>

<complexType name="SedGrpOfferKeyResultCodeType">
 <complexContent>
  <extension base="sppfs:ResultCodeType">
   <sequence>
    <element name="sedGrpOfferKey" type="sppfs:SedGrpOfferKeyType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

An <spppRejectResponse> contains the elements necessary for the SPPF client to precisely determine the overall result of the request, and if an error occurred, it provides information about the specific SED Group Offer key(s) that caused the error.

The data elements within the SPP Protocol over SOAP Reject response are described as follows:

- clientTransId: Zero or one client transaction ID. This value is simply an echo of the client transaction ID that SPPF client passed into the SPPF update request. When included in the request, the SPPF server MUST return it in the corresponding response message.

- serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value MUST be unique for a given SPPF server.

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- detailResult: An optional response code, response message, and SedGrpOfferKeyType (as defined in this document) triplet. This element will be present only if any specific SED Group Offer key level error has occurred. It indicates the error condition and the exact request SED Group Offer key that contributed to the error. The response code will reflect the exact error. See Section 7.3 for further details.

7.2.5. Batch Operation Structure

An SPP Protocol over SOAP Batch request XML structure allows the SPPF client to send any of of Add, Del, Accept or Reject operations together in one single request. This gives an SPPF Client the flexibility to use one single request structure to perform more than operations (verbs). The batch request structure is wrapped within the <spppBatchRequest> element while a SPPF Batch response is wrapped within the <spppBatchResponse> element. This following sub-sections describe the spppBatchRequest and spppBatchResponse elements. Refer to Section 10 for an example of a batch operation.

7.2.5.1. Batch Request Structure

An SPP Protocol over SOAP Batch request definition is contained within the generic <spppBatchRequest> element.

```
 <element name="spppBatchRequest">
 <complexType>
  <sequence>
   <element name="clientTransId"
   type="sppfb:TransIdType" minOccurs="0"/>
   <element name="minorVer"
   type="sppfb:MinorVerType" minOccurs="0"/>
    <choice minOccurs="1" maxOccurs="unbounded">
     <element name="addObj" type="sppfb:BasicObjType"/>
     <element name="delObj" type="sppfb:ObjKeyType"/>
     <element name="acceptSedGrpOffer"
     type="sppfs:SedGrpOfferKeyType"/>
     <element name="rejectSedGrpOffer"
     type="sppfs:SedGrpOfferKeyType"/>
    </choice>
  </sequence>
 </complexType>
</element>
```

The data elements within the <sppBatchRequest> element are described as follows:

- clientTransId: Zero or one client-generated transaction ID that, within the context of the SPPF Client, identifies this request. This value can be used at the discretion of the SPPF client to track, log or correlate requests and their responses. SPPF Server MUST echo back this value to the Client in the corresponding response to the incoming request. SPPF Server will not check this value for uniqueness.

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the Get Server Details Operation described in Section 7.2.9.

- addObj: One or more elements of abstract type BasicObjType where each element identifies an object that needs to be added.

- delObj: One or more elements of abstract type ObjKeyType where each element identifies a key for the object that needs to be deleted .

- acceptSedGrpOffer: One or more elements of type SedGrpOfferKeyType where each element identifies a SED Group Offer that needs to be accepted.

- rejectSedGrpOffer: One or more elements of type SedGrpOfferKeyType where each element identifies a SED Group Offer that needs to be rejected.

With respect to handling of error conditions, conforming SPPP SOAP servers MUST stop processing elements in the request at the first error, and roll back any elements that had already been processed for that batch request ("stop and rollback").

7.2.5.2. Batch Response

An SPP Protocol over SOAP batch response structure is contained within the generic <sppBatchResponse> element. This response structure is used for an Batch request that contains many different types of SPPF operations.

```
<element name="spppBatchResponse">
  <complexType>
    <sequence>
      <element name="clientTransId" type="sppfb:TransIdType"
       minOccurs="0"/>
      <element name="serverTransId" type="sppfb:TransIdType"/>
      <element name="overallResult" type="sppfb:ResultCodeType"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="addResult"
         type="sppfs:ObjResultCodeType"/>
         <element name="delResult"
         type="sppfs:ObjKeyResultCodeType"/>
         <element name="acceptResult"
         type="sppfs:SedGrpOfferKeyResultCodeType"/>
         <element name="rejectResult"
         type="sppfs:SedGrpOfferKeyResultCodeType"/>
       </choice>
    </sequence>
  </complexType>
</element>
```

An <spppBatchResponse> contains the elements necessary for an SPPF client to precisely determine the overall result of various operations in the request, and if an error occurred, it provides

information about the specific objects or keys in the request that caused the error.

The data elements within the SPP Protocol over SOAP Batch response are described as follows:

- clientTransId: Zero or one client transaction ID. This value is simply an echo of the client transaction ID that SPPF client passed into the SPPF update request. When included in the request, the SPPF server MUST return it in the corresponding response message.

- serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value MUST be unique for a given SPPF server.

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- addResult: One or more elements of type ObjResultCodeType where each element identifies the result code, result message and the specific object that the result relates to.

- delResult: One or more elements of type ObjKeyResultCodeType where each element identifies the result code, result message and the specific object key that the result relates to.

- acceptResult: One or more elements of type SedGrpOfferKeyResultCodeType where each element identifies the result code, result message and the specific SED Group Offer key that the result relates to.

- rejectResult: One or more elements of type SedGrpOfferKeyResultCodeType where each element identifies the result code, result message and the specific SED Group Offer key that the result relates to.

7.2.6. Get Operation Structure

In order to query the details of an object from the Registry, an authorized entity can send the spppGetRequest to the registry with a GetRqstType XML data structure containing one or more object keys that uniquely identify the object whose details are being queried. The request structure for an SPP Protocol over SOAP Get operation is contained within the generic <spppGetRequest> element while an SPP Protocol over SOAP Get response is wrapped within the generic <spppGetResponse> element. The following sub-sections describe the

spppGetRequest and spppGetResponse element. Refer to Section 10 for an example of SPP Protocol over SOAP Get operation on each type of SPPF object

7.2.6.1. Get Request

```
<element name="spppGetRequest">
 <complexType>
  <sequence>
   <element name="minorVer"
   type="sppfb:MinorVerType" minOccurs="0"/>
   <element name="objKey"
   type="sppfb:ObjKeyType"
   maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>
```

The data elements within the <spppGetRequest> element are described as follows:

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the Get Server Details Operation described in Section 7.2.9.

- objKey: One or more elements of abstract type ObjKeyType (as defined in [I-D.draft-ietf-drinks-spp-framework]). Each element contains attributes that uniquely identify the object that the client is requesting the server to query. Refer to Section 7.1 of this document for a description of all concrete object key types, for various SPPF objects, which are eligible to be passed into this element.

7.2.6.2. Get Response

The spppGetResponse element is described in Section 7.2.8.

7.2.7. Get SED Group Offers Operation Structure

In addition to the ability to query the details of one or more SED Group offers using an a SED Group Offer key in the spppGetRequest, this operation also provides an additional, more flexible, structure to query for SED Group Offer objects. This additional structure is contained within the <getSedGrpOffersRequest> element while the response is wrapped within the generic <spppGetResponse> element. The following sub-sections describe the getSedGrpOffersRequest and spppGetResponse elements.

7.2.7.1. Get SED Group Offers Request

Using the details passed into this structure, the server will attempt to find SED Group Offer objects that satisfy all the criteria passed into the request. If no criteria is passed in then the SPPF Server will return the list of SED Group Offer objects that belongs to the registrant. If there are no matching SED Group Offers found then an empty result set will be returned.

```
<element name="getSedGrpOffersRequest">
<complexType>
 <sequence>
  <element name="minorVer" type="sppfb:MinorVerType"
  minOccurs="0"/>
 <element name="offeredBy" type="sppfb:OrgIdType"
 minOccurs="0" maxOccurs="unbounded"/>
 <element name="offeredTo" type="sppfb:OrgIdType"
 minOccurs="0" maxOccurs="unbounded"/>
 <element name="status" type="sppfb:SedGrpOfferStatusType"
  minOccurs="0"/>
 <element name="sedGrpOfferKey" type="sppfs:SedGrpOfferKeyType"
 minOccurs="0" maxOccurs="unbounded"/>
 </sequence>
</complexType>
</element>
```

The data elements within the <getSedGrpOffersRequest> element are described as follows:

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPPF API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of SPPF that the client is using. If the element is not present, the server assumes that the client is using the latest minor version

supported by the SPPF server for the given major version. The versions supported by a given SPPF server can be retrieved by the client using the Get Server Details Operation described in Section 7.2.9.

- offeredBy: Zero or more organization IDs. Only offers that are offered to the organization IDs in this list should be included in the result set. The result set is also subject to other query criteria in the request.

- offeredTo: Zero or more organization IDs. Only offers that are offered by the organization IDs in this list should be included in the result set. The result set is also subject to other query criteria in the request.

- status: The status of the offer, offered or accepted. Only offers in the specified status should be included in the result set. If this element is not present then the status of the offer should not be considered in the query. The result set is also subject to other query criteria in the request.

- sedGrpOfferKey: Zero or more SED Group Offer Keys. Only offers having one of these keys should be included in the result set. The result set is also subject to other query criteria in the request.

7.2.7.2. Get SED Group Offers Response

The spppGetResponse element is described in Section 7.2.8.

7.2.8. Generic Query Response

An SPP Protocol over SOAP query response object is contained within the generic <spppGetResponse> element.

```
<element name="spppGetResponse">
 <complexType>
  <sequence>
   <element name="overallResult"
   type="sppfs:ResultCodeType"/>
   <element name="resultObj"
   type="sppfb:BasicObjType"
   minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>
```

An <spppGetResponse> contains the elements necessary for the SPPF client to precisely determine the overall result of the query, and details of any SPPF objects that matched the criteria in the request.

The data elements within the SPP Protocol over SOAP query response are described as follows:

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- resultObj: The set of zero or more objects that matched the query criteria. If no objects matched the query criteria then the result object(s) MUST be empty and the overallResult value MUST indicate success (if no matches are found for the query criteria, the response is considered a success).

7.2.9. Get Server Details Operation Structure

In order to query certain details of the SPPF server, such as the SPPF server's status and the major/minor version supported by the server, the Server Details operation structure SHOULD be used. This structure is contained within the <spppServerStatusRequest> element whereas a SPPF server status response is wrapped within the <spppServerStatusResponse> element. The following sub-sections describe the spppServerStatusRequest and spppServerStatusResponse elements.

7.2.9.1. Get Server Details Request

```
<element name="spppServerStatusRequest">
 <complexType>
  <sequence>
   <element name="minorVer"
   type="sppfb:MinorVerType" minOccurs="0"/>
  </sequence>
 </complexType>
</element>
```

The data elements within the <spppServerStatusRequest> element are described as follows:

- minorVer: Zero or one minor version identifier, indicating the minor version of the SPP Protocol over SOAP API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace to identify the version of

SPP Protocol over SOAP that the client is using. If the element is not present, the server assumes that the client is using the latest minor version of SPP Protocol over SOAP supported by the SPPF server for the given major version. The versions of SPP Protocol over SOAP supported by a given SPPF server can be retrieved by the client using this same spppServerStatusRequest without passing in the minorVer element.

7.2.9.2. Get Server Details Response

An SPP Protocol over SOAP server details response structure is contained within the generic <spppServerStatusResponse> element.

```
<element name="spppServerStatusResponse">
 <complexType>
  <sequence>
   <element name="overallResult" type="sppfs:ResultCodeType"/>
   <element name="svcMenu" type="sppfb:SvcMenuType"/>
  </sequence>
 </complexType>
</element>
```

The data elements within the <spppServerStatusResponse> element are described as follows:

- overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See Section 7.3 for further details.

- svcMenu: Exactly one element of type SvcMenuType which in turn contains the elements to return the server status, the major and minor versions of the SPP Protocol over SOAP supported by the SPPF server (refer Section 12 of [I-D.draft-ietf-drinks-spp-framework] for definition of SvcMenuType).

7.3. Response Codes and Messages

This section contains the listing of response codes and their corresponding human-readable text. These response codes are in conformance with the response types defined in Section 5.3 of [I-D.draft-ietf-drinks-spp-framework].

The response code numbering scheme generally adheres to the theory formalized in section 4.2.1 of [RFC5321]:

- The first digit of the response code can only be 1 or 2: 1 = a positive result, 2 = a negative result.

- The second digit of the response code indicates the category: 0 = Protocol Syntax, 1 = Implementation Specific Business Rule, 2 = Security, 3 = Server System.

- The third and fourth digits of the response code indicate the individual message event within the category defines by the first two digits.

The response codes are also categorized as to whether they are overall response codes that may only be returned in the "overallResult" data element in SPPF responses, or object level response codes that may only be returned in the "detailResult" element of the SPPF responses.

Result Code	Result Message	Overall or Object Level
1000	Request Succeeded.	Overall Response Code
2000	Request syntax invalid.	Overall Response Code
2001	Request too large. MaxSupported:[Maximum requests supported]	Overall Response Code
2002	Version not supported.	Overall Response Code
2100	Command invalid.	Overall Response Code
2300	System temporarily unavailable.	Overall Response Code
2301	Unexpected internal system or server error.	Overall Response Code
2101	Attribute value invalid. AttrName:[AttributeName] AttrVal:[AttributeValue]	Object Level Response Code
2102	Object does not exist. AttrName:[AttributeName] AttrVal:[AttributeValue]	Object Level Response Code
2103	Object status or ownership does not allow for operation. AttrName:[AttributeName] AttrVal:[AttributeValue]	Object Level Response Code

Table 1: Response Codes Numbering Scheme and Messages

Response message for response code 2001 is "parameterized" with the following parameter: "[Maximum requests supported]". When the request is too large, this parameter MUST be used to indicate the maximum number of requests supported by the server in a single protocol operation.

Each of the object level response messages are "parameterized" with the following parameters: "AttributeName" and "AttributeValue".

For example, if an SPPF client sends a request to delete a Destination Group with a name "TestDG", and it does not already exist, then the error message returned should be: "Attribute value invalid. AttrName:dgName AttrVal:TestDG".

The use of these parameters MUST adhere to the rules defined in Section 5.3 of [I-D.draft-ietf-drinks-spp-framework].

8. Protocol Operations

Refer to Section 7 of [I-D.draft-ietf-drinks-spp-framework] for a description of all SPPF operations, and any necessary semantics that MUST be adhered to in order to conform with SPPF.

9. SPP Protocol over SOAP WSDL Definition

The SPP Protocol over SOAP WSDL and data types are 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 SPPF protocol. There is a single WSDL structure for each type of SPPF operation. Each such WSDL structure contains exactly one input structure and one output structure that wraps any data elements that are part of the incoming request and the outgoing response respectively. 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.

Notes: The following WSDL has been formatted (e.g. tabs, spaces) to meet IETF document requirements. Deployments MUST replace "REPLACE_WITH_ACTUAL_URL" in the WSDL below with the URI of the SPPF Server instance.

```
<?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:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sppfb="urn:ietf:params:xml:ns:sppf:base:1"
xmlns:sppfs="urn:ietf:params:xml:ns:sppf:soap:1"
targetNamespace="urn:ietf:params:xml:ns:sppf:soap:1">
 <wsdl:types>
  <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:sppfs="urn:ietf:params:xml:ns:sppf:soap:1"
  targetNamespace="urn:ietf:params:xml:ns:sppf:soap:1">
   <annotation>
```

```
      <documentation>
       ---- Import base schema ----
      </documentation>
     </annotation>
     <import namespace="urn:ietf:params:xml:ns:sppf:base:1"
     schemaLocation="sppfbase.xsd"/>
     <annotation>
      <documentation>
       ---- Key type(s) extended
       from base schema. ----
      </documentation>
     </annotation>
     <complexType name="ObjKeyType">
      <complexContent>
       <extension base="sppfb:ObjKeyType">
        <sequence>
         <element name="rant" type="sppfb:OrgIdType"/>
         <element name="name" type="sppfb:ObjNameType"/>
         <element name="type" type="sppfs:ObjKeyTypeEnum"/>
        </sequence>
       </extension>
      </complexContent>
     </complexType>
   <simpleType name="ObjKeyTypeEnum">
     <restriction base="token">
       <enumeration value="SedGrp"/>
       <enumeration value="DestGrp"/>
       <enumeration value="SedRec"/>
       <enumeration value="EgrRte"/>
     </restriction>
   </simpleType>

     <complexType name="SedGrpOfferKeyType">
      <complexContent>
       <extension base="sppfb:SedGrpOfferKeyType">
        <sequence>
         <element name="sedGrpKey"
         type="sppfs:ObjKeyType"/>
         <element name="offeredTo"
         type="sppfb:OrgIdType"/>
        </sequence>
       </extension>
      </complexContent>
     </complexType>

     <complexType name="PubIdKeyType">
      <complexContent>
       <extension base="sppfb:PubIdKeyType">
```

```
        <sequence>
         <element name="rant" type="sppfb:OrgIdType"/>
         <choice>
          <element name="number"
          type="sppfb:NumberType"/>
          <element name="range"
          type="sppfb:NumberRangeType"/>
         </choice>
        </sequence>
       </extension>
      </complexContent>
     </complexType>

     <annotation>
      <documentation>
       ---- Generic Request and
       Response Definitions ----
      </documentation>
     </annotation>
     <element name="spppAddRequest">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
        <element name="obj" type="sppfb:BasicObjType"
        maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppDelRequest">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
        <element name="objKey"
        type="sppfb:ObjKeyType" maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppAcceptRequest">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
```

```
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
        <element name="sedGrpOfferKey"
        type="sppfs:SedGrpOfferKeyType"
        maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppRejectRequest">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
        <element name="sedGrpOfferKey"
        type="sppfs:SedGrpOfferKeyType"
        maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppGetRequest">
      <complexType>
       <sequence>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
        <element name="objKey"
        type="sppfb:ObjKeyType"
        maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppBatchRequest">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
         <choice minOccurs="1" maxOccurs="unbounded">
          <element name="addObj" type="sppfb:BasicObjType"/>
          <element name="delObj" type="sppfb:ObjKeyType"/>
          <element name="acceptSedGrpOffer"
          type="sppfs:SedGrpOfferKeyType"/>
          <element name="rejectSedGrpOffer"
          type="sppfs:SedGrpOfferKeyType"/>
         </choice>
       </sequence>
```

```
      </complexType>
     </element>
     <element name="spppServerStatusRequest">
      <complexType>
       <sequence>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
       </sequence>
      </complexType>
     </element>
     <element name="getSedGrpOffersRequest">
      <complexType>
       <sequence>
        <element name="minorVer"
        type="sppfb:MinorVerType" minOccurs="0"/>
       <element name="offeredBy"
       type="sppfb:OrgIdType" minOccurs="0"
       maxOccurs="unbounded"/>
       <element name="offeredTo" type="sppfb:OrgIdType"
       minOccurs="0" maxOccurs="unbounded"/>
       <element name="status"
       type="sppfb:SedGrpOfferStatusType" minOccurs="0"/>
       <element name="sedGrpOfferKey"
       type="sppfs:SedGrpOfferKeyType"
       minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppAddResponse">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="serverTransId"
        type="sppfb:TransIdType"/>
        <element name="overallResult"
        type="sppfs:ResultCodeType"/>
        <element name="detailResult"
        type="sppfs:ObjResultCodeType"
        minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppDelResponse">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
```

```
        <element name="serverTransId"
        type="sppfb:TransIdType"/>
        <element name="overallResult"
        type="sppfs:ResultCodeType"/>
        <element name="detailResult"
        type="sppfs:ObjKeyResultCodeType"
        minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppAcceptResponse">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="serverTransId"
        type="sppfb:TransIdType"/>
        <element name="overallResult"
        type="sppfs:ResultCodeType"/>
        <element name="detailResult"
        type="sppfs:SedGrpOfferKeyResultCodeType"
        minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
     <element name="spppRejectResponse">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="serverTransId"
        type="sppfb:TransIdType"/>
        <element name="overallResult"
        type="sppfs:ResultCodeType"/>
        <element name="detailResult"
        type="sppfs:SedGrpOfferKeyResultCodeType"
        minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
      </complexType>
     </element>
      <element name="spppBatchResponse">
      <complexType>
       <sequence>
        <element name="clientTransId"
        type="sppfb:TransIdType" minOccurs="0"/>
        <element name="serverTransId"
        type="sppfb:TransIdType"/>
        <element name="overallResult"
```

```
          type="sppfs:ResultCodeType"/>
           <choice minOccurs="0" maxOccurs="unbounded">
            <element name="addResult"
                     type="sppfs:ObjResultCodeType"/>
            <element name="delResult"
                     type="sppfs:ObjKeyResultCodeType"/>
            <element name="acceptResult"
                     type="sppfs:SedGrpOfferKeyResultCodeType"/>
            <element name="rejectResult"
                   type="sppfs:SedGrpOfferKeyResultCodeType"/>
           </choice>
          </sequence>
         </complexType>
        </element>
        <element name="spppGetResponse">
         <complexType>
          <sequence>
           <element name="overallResult"
           type="sppfs:ResultCodeType"/>
           <element name="resultObj"
           type="sppfb:BasicObjType"
           minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
         </complexType>
        </element>
        <element name="spppServerStatusResponse">
         <complexType>
          <sequence>
           <element name="overallResult"
           type="sppfs:ResultCodeType"/>
           <element name="svcMenu"
           type="sppfb:SvcMenuType"/>
          </sequence>
         </complexType>
        </element>
        <annotation>
         <documentation>
          ---- Operation Result Type
          Definitions ----
         </documentation>
        </annotation>
        <complexType name="ResultCodeType">
         <sequence>
          <element name="code" type="sppfs:ResultCodeValType"/>
          <element name="msg" type="sppfs:MsgType"/>
         </sequence>
        </complexType>
```

```
<simpleType name="ResultCodeValType">
  <restriction base="unsignedShort">
    <enumeration value="1000"/>
    <enumeration value="2000"/>
    <enumeration value="2001"/>
    <enumeration value="2002"/>
    <enumeration value="2100"/>
    <enumeration value="2101"/>
    <enumeration value="2102"/>
    <enumeration value="2103"/>
    <enumeration value="2300"/>
    <enumeration value="2301"/>
  </restriction>
</simpleType>

<simpleType name="MsgType">
  <restriction base="token">
   <minLength value="3"/>
   <maxLength value="255"/>
  </restriction>
 </simpleType>

<complexType name="ObjResultCodeType">
 <complexContent>
  <extension base="sppfs:ResultCodeType">
   <sequence>
    <element name="obj" type="sppfb:BasicObjType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="ObjKeyResultCodeType">
 <complexContent>
  <extension base="sppfs:ResultCodeType">
   <sequence>
    <element name="objKey" type="sppfb:ObjKeyType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
   <complexType name="SedGrpOfferKeyResultCodeType">
 <complexContent>
  <extension base="sppfs:ResultCodeType">
   <sequence>
    <element name="sedGrpOfferKey"
    type="sppfs:SedGrpOfferKeyType"/>
   </sequence>
  </extension>
```

```
      </complexContent>
      </complexType>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="spppAddRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppAddRequest"/>
  </wsdl:message>
  <wsdl:message name="spppDelRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppDelRequest"/>
  </wsdl:message>
  <wsdl:message name="spppAcceptRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppAcceptRequest"/>
  </wsdl:message>
  <wsdl:message name="spppRejectRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppRejectRequest"/>
  </wsdl:message>
  <wsdl:message name="spppBatchRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppBatchRequest"/>
  </wsdl:message>
  <wsdl:message name="spppGetRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppGetRequest"/>
  </wsdl:message>
  <wsdl:message name="spppGetSedGrpOffersRequestMsg">
   <wsdl:part name="rqst" element="sppfs:getSedGrpOffersRequest"/>
  </wsdl:message>
  <wsdl:message name="spppAddResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppAddResponse"/>
  </wsdl:message>
   <wsdl:message name="spppDelResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppDelResponse"/>
  </wsdl:message>
   <wsdl:message name="spppAcceptResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppAcceptResponse"/>
  </wsdl:message>
   <wsdl:message name="spppRejectResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppRejectResponse"/>
  </wsdl:message>
   <wsdl:message name="spppBatchResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppBatchResponse"/>
  </wsdl:message>
  <wsdl:message name="spppGetResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppGetResponse"/>
  </wsdl:message>
  <wsdl:message name="spppServerStatusRequestMsg">
   <wsdl:part name="rqst" element="sppfs:spppServerStatusRequest"/>
  </wsdl:message>
  <wsdl:message name="spppServerStatusResponseMsg">
   <wsdl:part name="rspns" element="sppfs:spppServerStatusResponse"/>
```

```
    </wsdl:message>
    <wsdl:portType name="spppPortType">
     <wsdl:operation name="submitAddRqst">
      <wsdl:input message="sppfs:spppAddRequestMsg"/>
      <wsdl:output message="sppfs:spppAddResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitDelRqst">
      <wsdl:input message="sppfs:spppDelRequestMsg"/>
      <wsdl:output message="sppfs:spppDelResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitAcceptRqst">
      <wsdl:input message="sppfs:spppAcceptRequestMsg"/>
      <wsdl:output message="sppfs:spppAcceptResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitRejectRqst">
      <wsdl:input message="sppfs:spppRejectRequestMsg"/>
      <wsdl:output message="sppfs:spppRejectResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitBatchRqst">
      <wsdl:input message="sppfs:spppBatchRequestMsg"/>
      <wsdl:output message="sppfs:spppBatchResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitGetRqst">
      <wsdl:input message="sppfs:spppGetRequestMsg"/>
      <wsdl:output message="sppfs:spppGetResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitGetSedGrpOffersRqst">
      <wsdl:input message="sppfs:spppGetSedGrpOffersRequestMsg"/>
      <wsdl:output message="sppfs:spppGetResponseMsg"/>
     </wsdl:operation>
     <wsdl:operation name="submitServerStatusRqst">
      <wsdl:input message="sppfs:spppServerStatusRequestMsg"/>
      <wsdl:output message="sppfs:spppServerStatusResponseMsg"/>
     </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="spppSoapBinding" type="sppfs:spppPortType">
     <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http"/>
     <wsdl:operation name="submitAddRqst">
      <soap:operation soapAction="submitAddRqst" style="document"/>
      <wsdl:input>
       <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
       <soap:body use="literal"/>
      </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="submitDelRqst">
```

```
      <soap:operation soapAction="submitDelRqst" style="document"/>
      <wsdl:input>
       <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
       <soap:body use="literal"/>
      </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="submitAcceptRqst">
      <soap:operation soapAction="submitAcceptRqst" style="document"/>
      <wsdl:input>
       <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
       <soap:body use="literal"/>
      </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="submitRejectRqst">
      <soap:operation soapAction="submitRejectRqst" style="document"/>
      <wsdl:input>
       <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
       <soap:body use="literal"/>
      </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="submitBatchRqst">
      <soap:operation soapAction="submitBatchRqst" style="document"/>
      <wsdl:input>
       <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
       <soap:body use="literal"/>
      </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="submitGetRqst">
      <soap:operation soapAction="submitGetRqst" style="document"/>
      <wsdl:input>
       <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
       <soap:body use="literal"/>
      </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="submitGetSedGrpOffersRqst">
      <soap:operation soapAction="submitGetSedGrpOffersRqst"
      style="document"/>
      <wsdl:input>
```

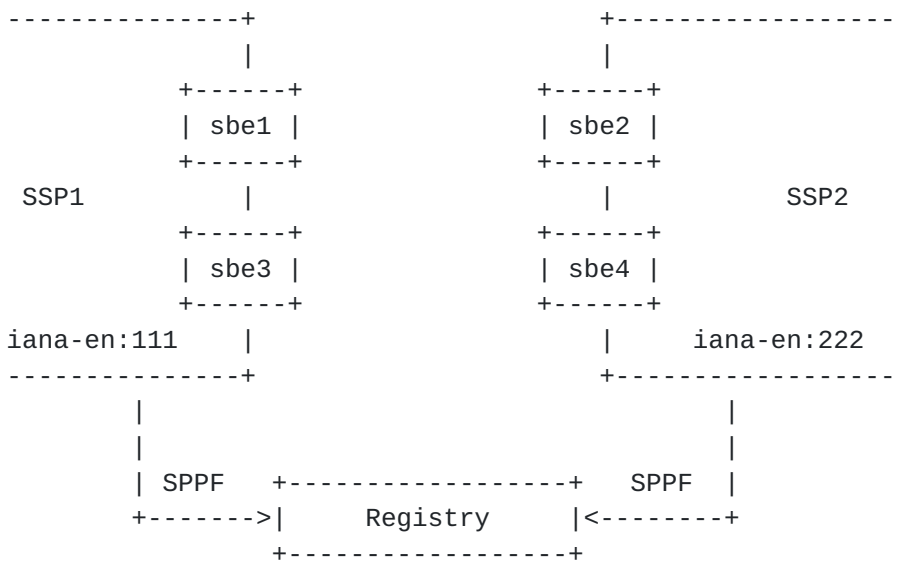
```
      <soap:body use="literal"/>
     </wsdl:input>
     <wsdl:output>
      <soap:body use="literal"/>
     </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="submitServerStatusRqst">
     <soap:operation soapAction="submitServerStatusRqst"
     style="document"/>
     <wsdl:input>
      <soap:body use="literal"/>
     </wsdl:input>
     <wsdl:output>
      <soap:body use="literal"/>
     </wsdl:output>
    </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="spppService">
    <wsdl:port name="spppPort" binding="sppfs:spppSoapBinding">
     <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
    </wsdl:port>
   </wsdl:service>
  </wsdl:definitions>
```

Figure 2: WSDL

10. SPP Protocol over SOAP Examples

This section shows XML message exchange between two SIP Service Providers (SSP) and a registry. The messages in this section are valid XML instances that conform to the SPP Protocol over SOAP schema version within this document. This section also relies on the XML data structures defined in the SPPF specification [I-D.draft-ietf-drinks-spp-framework]. Which should also be referenced to understand XML object types embedded in these example messages.

In this sample use case scenario, SSP1 and SSP2 provision resource data in the registry and use SPPF constructs to selectively share the SED groups. In the figure below, SSP2 has two ingress SBE instances that are associated with the public identities that SSP2 has the retail relationship with. Also, the two Session Border Element instances for SSP1 are used to show how to use SPPF to associate route preferences for the destination ingress routes and exercise greater control on outbound traffic to the peer's ingress SBEs.



10.1. Add Destination Group

SSP2 adds a destination group to the registry for use later. The SSP2 SPPF client sets a unique transaction identifier 'txn_1479' for tracking purposes. The name of the destination group is set to DEST_GRP_SSP2_1

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <!--Optional:-->
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:DestGrpType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

The registry processes the request and return a favorable response confirming successful creation of the named destination group. Also, besides returning a unique server transaction identifier, Registry also returns the matching client transaction identifier from the request message back to the SPPF client.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.2. Add SED Records

SSP2 adds SED records in the form of ingress routes to the registry.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <!--Optional:-->
   <clientTransId>txn_1479</clientTransId>
   <!--1 or more repetitions:-->
   <obj xsi:type="urn1:NAPTRType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:sedName>SED_SSP2_SBE2</urn1:sedName>
    <urn1:isInSvc>true</urn1:isInSvc>
    <urn1:order>10</urn1:order>
    <urn1:flags>u</urn1:flags>
    <urn1:svcs>E2U+sip</urn1:svcs>
    <urn1:regx>
     <urn1:ere>^(.*)$</urn1:ere>
     <urn1:repl>sip:\1@sbe2.ssp2.example.com</urn1:repl>
    </urn1:regx>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

The registry returns a success response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.3. Add SED Records -- URIType

SSP2 adds another SED record to the registry and makes use of URIType

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:URIType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:sedName>SED_SSP2_SBE4</urn1:sedName>
    <urn1:isInSvc>true</urn1:isInSvc>
    <urn1:ere>^(.*)$</urn1:ere>
    <urn1:uri>sip:\1;npdi@sbe4.ssp2.example.com</urn1:uri>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

The registry returns a success response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.4. Add SED Group

SSP2 creates the grouping of SED records (e.g. ingress routes) and chooses higher precedence for SED_SSP2_SBE2 by setting a lower number for the "priority" attribute, a protocol agnostic precedence indicator.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:SedGrpType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:sedGrpName>SED_GRP_SSP2_1</urn1:sedGrpName>
    <urn1:sedRecRef>
     <urn1:sedKey xsi:type="urn:ObjKeyType">
      <rant>iana-en:222</rant>
      <name>SED_SSP2_SBE2</name>
      <type>SedRec</type>
     </urn1:sedKey>
     <urn1:priority>100</urn1:priority>
    </urn1:sedRecRef>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
    <urn1:isInSvc>true</urn1:isInSvc>
    <urn1:priority>10</urn1:priority>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

To confirm successful processing of this request, registry returns a well-known result code '1000' to the SSP2 client.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.5. Add Public Identity -- Successful COR claim

SSP2 activates a TN public identity by associating it with a valid destination group. Further, SSP2 puts forth a claim that it is the carrier-of-record for the TN.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:TNType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
    <urn1:tn>+12025556666</urn1:tn>
    <urn1:corInfo>
     <urn1:corClaim>true</urn1:corClaim>
    </urn1:corInfo>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Assuming that the registry has access to TN authority data and it performs the required checks to verify that SSP2 is in fact the service provider of record for the given TN, the request is processed successfully. In the response message, the registry sets the value of <cor> to "true" in order to confirm SSP2 claim as the carrier of record and the <corDate> reflects the time when the carrier of record claim is processed.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
   <detailResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
    <obj xsi:type="ns2:TNType">
     <ns2:rant>iana-en:222</ns2:rant>
     <ns2:rar>iana-en:223</ns2:rar>
     <ns2:cDate>2010-05-30T09:30:10Z</ns2:cDate>
     <ns2:dgName>DEST_GRP_SSP2_1</ns2:dgName>
     <ns2:tn>+12025556666</ns2:tn>
     <ns2:corInfo>
      <ns2:corClaim>true</ns2:corClaim>
      <ns2:cor>true</ns2:cor>
      <ns2:corDate>2010-05-30T09:30:11Z</ns2:corDate>
     </ns2:corInfo>
    </obj>
   </detailResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.6. Add LRN

If another entity that SSP2 shares session establishment information (e.g. routes) with has access to Number Portability data, it may choose to perform route lookups by routing number. Therefore, SSP2 associates a routing number to a destination group in order to facilitate ingress route discovery.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:RNType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
    <urn1:rn>2025550000</urn1:rn>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response to the SPPF client.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.7. Add TN Range

Next, SSP2 activates a block of ten thousand TNs and associate it to a destination group.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:TNRType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
    <urn1:range>
     <urn1:startTn>+12026660000</urn1:startTn>
     <urn1:endTn>+12026669999</urn1:endTn>
    </urn1:range>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.8. Add TN Prefix

Next, SSP2 activates a block of ten thousand TNs using the TNPType structure and identifying a TN prefix.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:TNPType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
    <urn1:tnPrefix>+1202777</urn1:tnPrefix>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.9. Enable Peering -- SED Group Offer

In order for SSP1 to complete session establishment for a destination TN where the target subscriber has a retail relationship with SSP2, it first requires an asynchronous bi-directional handshake to show mutual consent. To start the process, SSP2 initiates the peering handshake by offering SSP1 access to its SED group.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:SedGrpOfferType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:sedGrpOfferKey xsi:type="urn:SedGrpOfferKeyType">
     <sedGrpKey xsi:type="urn:ObjKeyType">
      <rant>iana-en:222</rant>
      <name>SED_GRP_SSP2_1</name>
      <type>SedGrp</type>
     </sedGrpKey>
     <offeredTo>iana-en:111</offeredTo>
    </urn1:sedGrpOfferKey>
    <urn1:status>offered</urn1:status>
    <urn1:offerDateTime>
      2006-05-04T18:13:51.0Z
    </urn1:offerDateTime>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and confirms that the SSP1 will now have the opportunity to weigh in on the offer and either accept or reject it. The registry may employ out-of-band notification mechanisms for quicker updates to SSP1 so they can act faster, though this topic is beyond the scope of this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.10. Enable Peering -- SED Group Offer Accept

SSP1 responds to the offer from SSP2 and agrees to have visibility to SSP2 session establishment information (e.g. ingress routes).

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAcceptRequest>
   <!--Optional:-->
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <sedGrpOfferKey>
    <sedGrpKey>
     <rant>iana-en:222</rant>
     <name>SED_GRP_SSP2_1</name>
     <type>SedGrp</type>
    </sedGrpKey>
    <offeredTo>iana-en:111</offeredTo>
   </sedGrpOfferKey>
  </urn:spppAcceptRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry confirms that the request has been processed successfully. From this point forward, if SSP1 looks up a public identity through the query resolution server, where the public identity is part of the destination group by way of "SED_GRP_SSP2_1" session establishment data association, SSP2 ingress SBE information will be shared with SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAcceptResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12350</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAcceptResponse>
 </S:Body>
</S:Envelope>
```

10.11. Add Egress Route

SSP1 wants to prioritize all outbound traffic to the ingress route associated with the "SED_GRP_SSP2_1" SED Group record, through "sbe1.ssp1.example.com".

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppAddRequest>
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <obj xsi:type="urn1:EgrRteType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:egrRteName>EGR_RTE_01</urn1:egrRteName>
    <urn1:pref>50</urn1:pref>
    <urn1:regxRewriteRule>
     <urn1:ere>^(.*@)(.*)$</urn1:ere>
     <urn1:repl>\1\2?route=sbe1.ssp1.example.com</urn1:repl>
    </urn1:regxRewriteRule>
    <urn1:ingrSedGrp xsi:type="urn:ObjKeyType">
     <rant>iana-en:222</rant>
     <name>SED_GRP_SSP2_1</name>
     <type>SedGrp</type>
    </urn1:ingrSedGrp>
   </obj>
  </urn:spppAddRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Since peering has already been established, the request to add the egress route has been successfully completed.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppAddResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12345</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppAddResponse>
 </S:Body>
</S:Envelope>
```

10.12. Remove Peering -- SED Group Offer Reject

SSP1 had earlier accepted to have visibility to SSP2 session establishment data. SSP1 now decides to no longer maintain this visibility and hence rejects the SED Group Offer.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppRejectRequest>
   <!--Optional:-->
   <clientTransId>txn_1479</clientTransId>
    <!--1 or more repetitions:-->
   <sedGrpOfferKey>
    <sedGrpKey>
     <rant>iana-en:222</rant>
     <name>SED_GRP_SSP2_1</name>
     <type>SedGrp</type>
    </sedGrpKey>
    <offeredTo>iana-en:111</offeredTo>
   </sedGrpOfferKey>
  </urn:spppRejectRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry confirms that the request has been processed successfully. From this point forward, if SSP1 looks up a public identity through the query resolution server, where the public identity is part of the destination group by way of "SED_GRP_SSP2_1" session establishment data association, SSP2 ingress SBE information will not be shared with SSP1 and hence SSP2 ingress SBE will not be returned in the query response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppRejectResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <clientTransId>txn_1479</clientTransId>
   <serverTransId>tx_12350</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppRejectResponse>
 </S:Body>
</S:Envelope>
```

10.13. Get Destination Group

SSP2 uses the 'spppGetRequest' operation to tally the last provisioned record for destination group DEST_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppGetRequest>
   <!--1 or more repetitions:-->
   <objKey xsi:type="urn:ObjKeyType">
    <rant>iana-en:222</rant>
    <name>DEST_GRP_SSP2_1</name>
    <type>DestGrp</type>
   </objKey>
  </urn:spppGetRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <S:Body>
  <ns3:spppGetResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <overallResult>
    <code>1000</code>
    <msg>success</msg>
   </overallResult>
   <resultObj xsi:type="ns2:DestGrpType">
    <ns2:rant>iana-en:222</ns2:rant>
    <ns2:rar>iana-en:223</ns2:rar>
    <ns2:cDate>2012-10-22T09:30:10Z</ns2:cDate>
    <ns2:dgName>DEST_GRP_SSP2_1</ns2:dgName>
   </resultObj>
  </ns3:spppGetResponse>
 </S:Body>
</S:Envelope>
```

10.14. Get Public Identity

SSP2 obtains the last provisioned record associated with a given TN.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppGetRequest>
   <!--1 or more repetitions:-->
   <objKey xsi:type="urn:PubIdKeyType">
    <rant>iana-en:222</rant>
    <number>
     <urn1:value>+12025556666</urn1:value>
     <urn1:type>TN</urn1:type>
    </number>
   </objKey>
  </urn:spppGetRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <S:Body>
  <ns3:spppGetResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <overallResult>
    <code>1000</code>
    <msg>success</msg>
   </overallResult>
   <resultObj xsi:type="ns2:TNType">
    <ns2:rant>iana-en:222</ns2:rant>
    <ns2:rar>iana-en:223</ns2:rar>
    <ns2:cDate>2012-10-22T09:30:10Z</ns2:cDate>
    <ns2:dgName>DEST_GRP_SSP2_1</ns2:dgName>
    <ns2:tn>+12025556666</ns2:tn>
    <ns2:corInfo>
     <ns2:corClaim>true</ns2:corClaim>
     <ns2:cor>true</ns2:cor>
     <ns2:corDate>2010-05-30T09:30:10Z</ns2:corDate>
    </ns2:corInfo>
   </resultObj>
  </ns3:spppGetResponse>
 </S:Body>
</S:Envelope>
```

10.15. Get SED Group Request

SSP2 obtains the last provisioned record for the SED group SED_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppGetRequest>
   <!--1 or more repetitions:-->
   <objKey xsi:type="urn:ObjKeyType">
    <rant>iana-en:222</rant>
    <name>SED_GRP_SSP2_1</name>
    <type>SedGrp</type>
   </objKey>
  </urn:spppGetRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <S:Body>
  <ns3:spppGetResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <overallResult>
    <code>1000</code>
    <msg>success</msg>
   </overallResult>
   <resultObj xsi:type="ns2:SedGrpType">
    <ns2:rant>iana-en:222</ns2:rant>
    <ns2:rar>iana-en:223</ns2:rar>
    <ns2:cDate>2012-10-22T09:30:10Z</ns2:cDate>
    <ns2:sedGrpName>SED_GRP_SSP2_1</ns2:sedGrpName>
    <ns2:sedRecRef>
     <ns2:sedKey xsi:type="ns3:ObjKeyType">
      <rant>iana-en:222</rant>
      <name>SED_SSP2_SBE2</name>
      <type>SedRec</type>
     </ns2:sedKey>
     <ns2:priority>100</ns2:priority>
    </ns2:sedRecRef>
    <ns2:sedRecRef>
     <ns2:sedKey xsi:type="ns3:ObjKeyType">
      <rant>iana-en:222</rant>
      <name>SED_SSP2_SBE4</name>
      <type>SedRec</type>
     </ns2:sedKey>
     <ns2:priority>101</ns2:priority>
    </ns2:sedRecRef>
    <ns2:dgName>DEST_GRP_SSP2_1</ns2:dgName>
    <ns2:isInSvc>true</ns2:isInSvc>
    <ns2:priority>10</ns2:priority>
   </resultObj>
  </ns3:spppGetResponse>
 </S:Body>
</S:Envelope>
```

10.16. Get SED Group Offers Request

SSP2 fetches the last provisioned SED group offer to the <peeringOrg> SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:getSedGrpOffersRequest>
   <offeredTo>iana-en:111</offeredTo>
  </urn:getSedGrpOffersRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry processes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <S:Body>
  <ns3:spppGetResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <overallResult>
    <code>1000</code>
    <msg>success</msg>
   </overallResult>
   <resultObj xsi:type="ns2:SedGrpOfferType">
    <ns2:rant>iana-en:222</ns2:rant>
    <ns2:rar>iana-en:223</ns2:rar>
    <ns2:cDate>2012-10-22T09:30:10Z</ns2:cDate>
    <ns2:sedGrpOfferKey
     xsi:type="ns3:SedGrpOfferKeyType">
     <sedGrpKey>
      <rant>iana-en:222</rant>
      <name>SED_GRP_SSP2_1</name>
      <type>SedGrp</type>
     </sedGrpKey>
     <offeredTo>iana-en:111</offeredTo>
    </ns2:sedGrpOfferKey>
    <ns2:status>offered</ns2:status>
    <ns2:offerDateTime>
     2006-05-04T18:13:51.0Z
    </ns2:offerDateTime>
   </resultObj>
  </ns3:spppGetResponse>
 </S:Body>
</S:Envelope>
```

10.17. Get Egress Route

SSP1 wants to verify the last provisioned record for the egress route called EGR_RTE_01.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppGetRequest>
   <!--1 or more repetitions:-->
   <objKey xsi:type="urn:ObjKeyType">
    <rant>iana-en:111</rant>
    <name>EGR_RTE_01</name>
    <type>EgrRte</type>
   </objKey>
  </urn:spppGetRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <S:Body>
  <ns3:spppGetResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <overallResult>
    <code>1000</code>
    <msg>success</msg>
   </overallResult>
   <resultObj xsi:type="ns2:EgrRteType">
    <ns2:rant>iana-en:222</ns2:rant>
    <ns2:rar>iana-en:223</ns2:rar>
    <ns2:cDate>2012-10-22T09:30:10Z</ns2:cDate>
    <ns2:egrRteName>EGR_RTE_01</ns2:egrRteName>
    <ns2:pref>50</ns2:pref>
    <ns2:regxRewriteRule>
     <ns2:ere>^(.*)$</ns2:ere>
     <ns2:repl>sip:\1@sbe1.ssp1.example.com</ns2:repl>
    </ns2:regxRewriteRule>
    <ns2:ingrSedGrp xsi:type="ns3:ObjKeyType">
     <rant>iana-en:222</rant>
     <name>SED_GRP_SSP2_1</name>
     <type>SedRec</type>
    </ns2:ingrSedGrp>
   </resultObj>
  </ns3:spppGetResponse>
 </S:Body>
</S:Envelope>
```

10.18. Delete Destination Group

SSP2 initiates a request to delete the destination group DEST_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppDelRequest>
    <!--1 or more repetitions:-->
   <objKey xsi:type="urn:ObjKeyType">
    <rant>iana-en:222</rant>
    <name>DEST_GRP_SSP2_1</name>
    <type>DestGrp</type>
   </objKey>
  </urn:spppDelRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppDelResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <serverTransId>tx_12354</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppDelResponse>
 </S:Body>
</S:Envelope>
```

10.19. Delete Public Identity

SSP2 chooses to de-activate the TN and remove it from the registry.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppDelRequest>
   <!--1 or more repetitions:-->
   <objKey xsi:type="urn:PubIdKeyType">
    <rant>iana-en:222</rant>
    <number>
     <urn1:value>+12025556666</urn1:value>
     <urn1:type>TN</urn1:type>
    </number>
   </objKey>
  </urn:spppDelRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppDelResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <serverTransId>tx_12354</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppDelResponse>
 </S:Body>
</S:Envelope>
```

10.20. Delete SED Group Request

SSP2 removes the SED group called SED_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppDelRequest>
    <!--1 or more repetitions:-->
   <objKey xsi:type="urn:ObjKeyType">
    <rant>iana-en:222</rant>
    <name>SED_GRP_SSP2_1</name>
    <type>SedGrp</type>
   </objKey>
  </urn:spppDelRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppDelResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <serverTransId>tx_12354</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppDelResponse>
 </S:Body>
</S:Envelope>
```

10.21. Delete SED Group Offers Request

SSP2 no longer wants to share SED group SED_GRP_SSP2_1 with SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppDelRequest>
    <!--1 or more repetitions:-->
   <objKey xsi:type="urn:SedGrpOfferKeyType">
    <sedGrpKey>
     <rant>iana-en:222</rant>
     <name>SED_GRP_SSP2_1</name>
     <type>SedGrp</type>
    </sedGrpKey>
    <offeredTo>iana-en:111</offeredTo>
   </objKey>
  </urn:spppDelRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response. Restoring this resource sharing will require a new SED group offer from SSP2 to SSP1 followed by a successful SED group accept request from SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppDelResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <serverTransId>tx_12354</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppDelResponse>
 </S:Body>
</S:Envelope>
```

10.22. Delete Egress Route

SSP1 decides to remove the egress route with the label EGR_RTE_01.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppDelRequest>
   <!--1 or more repetitions:-->
   <objKey xsi:type="urn:ObjKeyType">
    <rant>iana-en:111</rant>
    <name>EGR_RTE_01</name>
    <type>EgrRte</type>
   </objKey>
  </urn:spppDelRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppDelResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <serverTransId>tx_12354</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppDelResponse>
 </S:Body>
</S:Envelope>
```

10.23. Batch Request

Following is an example of how some of the operations mentioned in previous sections MAY be performed by an SPPF client as a batch in one single SPP Protocol over SOAP request.

In the sample request below SSP1 wants to accept a SED Group Offer from SSP3, add a Destination Group, add a NAPTR SED Record, add a SED Group, add a SED Group Offer, delete a previously provisioned TN type Public Identifier, delete a previously provisioned SED Group, and reject a SED Group Offer from SSP4.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:ietf:params:xml:ns:sppf:soap:1"
 xmlns:urn1="urn:ietf:params:xml:ns:sppf:base:1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:spppBatchRequest>
   <clientTransId>txn_1467</clientTransId>
   <minorVer>1</minorVer>

   <acceptSedGrpOffer>
    <sedGrpKey>
     <rant>iana-en:225</rant>
     <name>SED_SSP3_SBE1_Offered</name>
     <type>SedGrp</type>
    </sedGrpKey>
    <offeredTo>iana-en:222</offeredTo>
   </acceptSedGrpOffer>

   <addObj xsi:type="urn1:DestGrpType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
   </addObj>

   <addObj xsi:type="urn1:NAPTRType">
    <urn1:rant>iana-en:222</urn1:rant>
    <urn1:rar>iana-en:223</urn1:rar>
    <urn1:sedName>SED_SSP2_SBE2</urn1:sedName>
    <urn1:order>10</urn1:order>
    <urn1:flags>u</urn1:flags>
    <urn1:svcs>E2U+sip</urn1:svcs>
    <urn1:regx>
```

```
        <urn1:ere>^(.*)$</urn1:ere>
        <urn1:repl>sip:\1@sbe2.ssp2.example.com</urn1:repl>
       </urn1:regx>
      </addObj>

      <addObj xsi:type="urn1:SedGrpType">
       <urn1:rant>iana-en:222</urn1:rant>
       <urn1:rar>iana-en:223</urn1:rar>
       <urn1:sedGrpName>SED_GRP_SSP2_1</urn1:sedGrpName>
       <urn1:sedRecRef>
        <urn1:sedKey xsi:type="urn:ObjKeyType">
         <rant>iana-en:222</rant>
         <name>SED_SSP2_SBE2</name>
         <type>SedRec</type>
        </urn1:sedKey>
       <urn1:priority>100</urn1:priority>
       </urn1:sedRecRef>
        <urn1:dgName>DEST_GRP_SSP2_1</urn1:dgName>
        <urn1:isInSvc>true</urn1:isInSvc>
        <urn1:priority>10</urn1:priority>
      </addObj>

      <addObj xsi:type="urn1:SedGrpOfferType">
       <urn1:rant>iana-en:222</urn1:rant>
       <urn1:rar>iana-en:223</urn1:rar>
       <urn1:sedGrpOfferKey xsi:type="urn:SedGrpOfferKeyType">
        <sedGrpKey xsi:type="urn:ObjKeyType">
         <rant>iana-en:222</rant>
         <name>SED_GRP_SSP2_1</name>
         <type>SedGrp</type>
        </sedGrpKey>
        <offeredTo>iana-en:111</offeredTo>
       </urn1:sedGrpOfferKey>
       <urn1:status>offered</urn1:status>
       <urn1:offerDateTime>
        2006-05-04T18:13:51.0Z
       </urn1:offerDateTime>
      </addObj>

      <delObj xsi:type="urn:PubIdKeyType">
       <rant>iana-en:222</rant>
       <number>
        <urn1:value>+12025556666</urn1:value>
        <urn1:type>TN</urn1:type>
       </number>
      </delObj>

      <delObj xsi:type="urn:ObjKeyType">
```

```
      <rant>iana-en:222</rant>
      <name>SED_GRP_SSP2_Previous</name>
      <type>SedGrp</type>
     </delObj>

     <rejectSedGrpOffer>
      <sedGrpKey>
       <rant>iana-en:226</rant>
       <name>SED_SSP4_SBE1_Offered</name>
       <type>SedGrp</type>
      </sedGrpKey>
      <offeredTo>iana-en:222</offeredTo>
     </rejectSedGrpOffer>

   </urn:spppBatchRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns3:spppBatchResponse
   xmlns:ns2="urn:ietf:params:xml:ns:sppf:base:1"
   xmlns:ns3="urn:ietf:params:xml:ns:sppf:soap:1">
   <serverTransId>tx_12354</serverTransId>
   <overallResult>
    <code>1000</code>
    <msg>Request Succeeded.</msg>
   </overallResult>
  </ns3:spppBatchResponse>
 </S:Body>
</S:Envelope>
```

11. Security Considerations

SPP Protocol over SOAP 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 may not be 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, which is often accomplished using the TLS certificate exchange and validation described in [RFC2818].

11.1. Vulnerabilities

Section 5 describes the use of HTTP and TLS as the underlying transport protocols for SPP Protocol over SOAP. These underlying protocols may have various vulnerabilities, and these may be inherited by SPP Protocol over SOAP. SPP Protocol over SOAP itself may have vulnerabilities because an authorization model is not explicitly specified in this document.

During a TLS handshake, TLS servers can optionally request a certificate from a TLS client; that option is not a requirement for this protocol. This presents a denial of service risk in which unauthenticated clients can consume server CPU resources by creating TLS sessions. The risk is increased if the server supports client-initiated renegotiation. This risk can be mitigated by disabling client-initiated renegotiation on the server and by ensuring that other means (such as firewall access control lists) are used to restrict unauthenticated client access to servers.

In conjunction with the above, it is important that SPP Protocol over SOAP implementations implement an authorization model that considers the source of each query or update request and determines whether it is reasonable to authorize that source to perform that specific query or update.

12. IANA Considerations

This document uses URNs to describe XML Namespaces and XML Schemas. According to [RFC3688], IANA is requested to perform the following URN assignment:

URN: urn:ietf:params:xml:ns:sppf:soap:1

Registrant Contact: IESG

XML: See Section 9 of [THISDOCUMENT]

13. Acknowledgements

This document is a result of various discussions held with the IETF DRINKS working group, specifically the protocol design team, with contributions from the following individuals, in alphabetical order: Alexander Mayrhofer, David Schwartz, Deborah A Guyton, Jean-Francois Mule Kenneth Cartwright, Lisa Dusseault, Manjul Maharishi, Mickael Marrache, Otmar Lendl, Peter Saint-Andre, Richard Shockey, Samuel

Melloul, Scott Hollenbeck, Sumanth Channabasappa, Syed Ali, and Vikas Bhatia .

14. References

14.1. Normative References

[I-D.draft-ietf-drinks-spp-framework]
Cartwright, K., Bhatia, V., Ali, S., and D. Schwartz, "Session Peering Provisioning Framework", draft-ietf-drinks-spp-framework-06 (work in progress), October 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] 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.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[SOAPREF] Gudgin, M., Hadley, M., Moreau, J., and H. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation REC-SOAP12-part1-20030624, June 2002.

14.2. Informative References

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[W3C.REC-xml-20081126]
Sperberg-McQueen, C., Yergeau, F., Bray, T., Maler, E., and J. Paoli, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

[WSDLREF] Christensen, E., Curbera, F., Meredith, G., and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note NOTE-wsdl-20010315, March 2001.

Authors' Addresses

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

Email: kcartwright@tnsi.com

Vikas Bhatia
TNS
1939 Roland Clarke Place
Reston, VA 20191
USA

Email: vbhatia@tnsi.com

Jean-Francois Mule
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
USA

Email: jfm@cablelabs.com

Alexander Mayrhofer
enum.at GmbH
Karlsplatz 1/9
Wien A-1010
Austria

Email: alexander.mayrhofer@enum.at