DRINKS
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2011

J-F. Mule
CableLabs
K. Cartwright
TNS
S. Ali
NeuStar
A. Mayrhofer
enum.at GmbH
July 7, 2010

Session Peering Provisioning Protocol
draft-drinks-spprov-00

Abstract

This document defines a protocol for provisioning session establishment data into Session Data Registries and SIP Service Provider data stores. The provisioned data is typically used by various network elements for session peering.

This document describes the Session Peering Provisioning Protocol used by clients to provision registries. The document provides a set of guiding principles for the design of this protocol including extensibility and independent transport definitions, a basic data model and an XML Schema Document.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 8, 2011.

Copyright Notice

Copyright (c) 2010 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

Service providers and enterprises use registries to make call or session routing decisions for Voice over IP, SMS and MMS traffic exchanges. This document is narrowly focused on the provisioning protocol for these registries. This protocol prescribes a way for an entity to provision session-related data into a registry. The data being provisioned can be optionally shared with other participating peering entities. The requirements and use cases driving this protocol have been documented in [I-D.ietf-drinks-usecases-requirements]. The reader is expected to be familiar with the terminology defined in the previously mentioned document.

Three types of provisioning flows have been described in the use case document: client to registry provisioning, registry to local data repository and registry-to-registry. This document addresses a subset (client-to-registry provisioning) by defining a Session Peering Provisioning Protocol (SPPP) for provisioning Session Establishment Data (SED) into a Registry (arrow numbered one in the figure below). While the other "provisioning flows" are shown below as separate message flows, no determination has been made for whether one common baseline protocol could be used for all three, or whether distinct protocols are required.

```
                             *------------*               *------------*
    (1). Provisioning SED    |            | (3).Registry  |            |
    -----------------------> |  Registry  |<-------------->|  Registry  |
         data into Registries|            |  to Registry  |            |
                             *------------*  exchanges    *------------*
                                  /  \                          \
                                 /    \                          \
                                /      \                          \
                               /        \                          v
                              /          \                         ...
                             /            \
                            / (2).         \
                           / Distributing   \
                          /      SED         \
                         V                    V
                        +----------+       +----------+
                        |Local Data|       |Local Data|
                        |Repository|       |Repository|
                        +----------+       +----------+
```

Three Registry Provisioning Flows

Figure 1

The data provisioned for session establishment is typically used by various downstream SIP signaling systems to route a call to the next hop associated with the called domain. These systems typically use a local data store ("Local Data Repository") as their source of session routing information. More specifically, the SED data is the set of parameters that the outgoing signaling path border elements (SBEs) need to initiate the session. See [RFC5486] for more details.

A "terminating" SIP Service Provider (SSP) provisions SED into the registry to be selectively shared with other peer SSPs. Subsequently, a Registry may distribute the provisioned data into local Data Repositories used for look-up queries (identifier -> URI) or for lookup and location resolution (identifier -> URI -> ingress SBE of terminating SSP). In some cases, the Registry may additionally offer a central query resolution service (not shown in the above figure).

A key requirement for the SPPP protocol is to be able to accommodate two basic deployment scenarios:

1. A Look-Up Function (LUF) to determine the target domain to assist in call routing (as described in [RFC5486]). In this case, the querying entity may use other means to perform the Location

Routing Function (LRF) which in turn helps determine the actual location of the Signaling Function in that domain.

2. Both Look-Up function (LUF) and Location Routing Function (LRF) to locate the SED data fully.

In terms of protocol design, SPPP protocol is agnostic to the transport. This document includes the description of the data model and the means to enable protocol operations within a request and response structure. To encourage interoperability, the protocol supports extensibility aspects.

Transport requirements are provided in this document to help with the selection of the optimum transport mechanism. ([I-D.ietf-drinks-sppp-over-soap]) identifies a SOAP transport mechanism for SPPP.

This document is organized as follows:

- Section 3 provides an overview of the SPPP protocol, including the layering approach, functional entities and data model;
- Section 4 defines requirements for SPPP transport protocols;
- Section 5 defines XML considerations that XML parsers must meet to conform to this specification.
- Section 6 describes the protocol request-reply model;
- Section 8 defines the protocol commands for this version of SPPP, and how to extend them;

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].

This document reuses terms from [RFC3261], [RFC5486], use cases and requirements documented in [I-D.ietf-drinks-usecases-requirements] and the ENUM Validation Architecture [RFC4725].

In addition, this document specifies the following additional terms:

SPPP: Session Peering Provisioning Protocol, the protocol used to provision data into a Registry (see arrow labeled "1." in Figure 1 of [I-D.ietf-drinks-usecases-requirements]). It is the primary scope of this document.

SPDP: Session Peering Distribution Protocol, the protocol used to distribute data to Local Data Repository (see arrow labeled "2." in Figure 1 of [I-D.ietf-drinks-usecases-requirements]).

Client: An application that supports an SPPP Client; it is sometimes referred to as a "Registry Client".

Registry: The Registry operates a master database of Session Establishment Data for one or more Registrants.

A Registry acts as an SPPP Server.

Registrant: In this document, we extend the definition of a Registrant based on [RFC4725]. The Registrant is the end-user, the person or organization who is the "holder" of the Session Establishment Data being provisioned into the Registry. For example, in [I-D.ietf-drinks-usecases-requirements], a Registrant is pictured as a SIP Service Provider in Figure 2.

A Registrant is identified by its name in the data model.

Registrar: In this document, we also extend the definition of a Registrar from [RFC4725]. A Registrar performs provisioning operations on behalf of a Registrant by interacting with the Registry, in our case via the SPPP protocol defined in this

document.

A Registrar is identified by its name in the data model.

3. Protocol Definition

This section introduces the structure of the data model and provides the information framework for the SPPP protocol. An overview of the protocol operations is first provided with a typical deployment scenario. The data model is then defined along with all the objects manipulated by the protocol and their relationships.

3.1. Protocol Overview and Layering

SPPP is a simple request/reply protocol that allows a client application to submit provisioning data and query requests to a server. The SPPP 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 the Transport Protocol Requirements section.

```
                Layer                      Example
            +-------------+     +-----------------------------+
        (5) |Data Objects |     |       RteGrpType, etc.      |
            +-------------+     +-----------------------------+
                   |                           |
            +-------------+     +-----------------------------+
        (4) | Operations  |     |     addRteGrpsRqst, etc.    |
            +-------------+     +-----------------------------+
                   |                           |
            +-------------+     +-----------------------------+
        (3) |   Message   |     | spppRequest, spppResponse   |
            +-------------+     +-----------------------------+
                   |                           |
            +-------------+     +-----------------------------+
        (2) |   Message   |     |   HTTP, SOAP, None, etc.    |
            |   Envelope  |     |                             |
            +-------------+     +-----------------------------+
                   |                           |
            +-------------+     +-----------------------------+
        (1) |  Transport  |     |     TCP, TLS, BEEP, etc.    |
            |   Protocol  |     |                             |
            +-------------+     +-----------------------------+
```

SPPP Layering

Figure 2

SPPP can be viewed as a set of layers that collectively define the structure of an SPPP request and response. Layers 1 and 2, as

detailed below, are left to separate specifications to allow for potentially multiple SPPP transport, envelope, and authentication technologies. This document defines layers 3, 4, and 5 below.

1. The transport protocol layer provides a communication mechanism between the client and server. SPPP can be layered over any transport protocol that provides a set of basic requirements defined in the Transport Protocol Requirements section.

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. The message layer provides a simple, envelope-independent and transport-independent, SPPP wrapper for SPPP request and response messages.

4. The operation layer defines the set of base SPPP actions that can be invoked using an SPPP message. Operations are encoded using XML encoded actions and objects.

5. The data object layer defines the base set of SPPP data objects that can be included in update operations or returned in operation responses.

3.2. Data Model

The data model illustrated and described in Figure 3 defines the logical objects and the relationships between these objects that the SPPP protocol supports. SPPP defines the protocol operations through which an SPPP Client populates a Registry with these logical objects. Various clients belonging to different Registrants and distinct Registrars may use the protocol for populating the Registry's data.

3.2.1. Structure of the SPPP Data Model

The logical structure presented below is consistent with the terminology and requirements defined in [I-D.ietf-drinks-usecases-requirements]. Note that the current version of this data model does not yet address the notion of Data Recipient Groups (left for a future revision of this document).

```
   +-------------+      +------------------+
   | all object  |      |Organization:     |
   | types       |      |orgName*,         |
   +------+------+      |sourceIdentLabels,|
          +------------>|peerPrefs,        |
                        |extension         |
    All objects are     |                  |
    associated with 2   |                  |
    Organizations to    +------------------+
    identify the            ^
    registrant and          |A Route Group is
    the registrar           |associated with
                            |zero or more
                            |Organizations
                            |
                   +--------+--------------+
                   |Route Group:           |        +-----[abstract]-+
                   |  registrantOrgName*,  |        |                |
                   |  registrarOrgName,    |        | Route Record:  |
                   |  rteGrpName*,         |        |  rteRecName*,  |
                   |  dgName*,             +------->|  priority,     |
                   |  isInService,         |        |  extension     |
                   |  rteRec*,             |        |                |
                   |  sourceOrgs,          |        +----------------+
                   |  sourceIdentLabels,   |            ^
                   |  extension            |            |Various types
                   +-----------------------+            |of Route
                         ^                              |Records...
                         |                       +------+-----------...
                         |                       |        |         |
                         |                     +----+  +-------+ +----+
                         |                     | URI|  | NAPTR | | NS |
        +----------------+-----+               +----+  +-------+ +----+
        |Destination           |
        |Group:                |     +----------[abstract]-+
        |  registrantOrgName*, |     |Public               |
        |  registrarOrgName,   |     |Identifier:          |
        |  dgName*,            |     |  registrantOrgName*,|
        |  extension           |<----+  registrarOrgName,  |
        +----------------------+     |  publicIdentifier*, |
                                     |  dgName*,           |
                                     |  extension          |
                                     +---------------------+
```

SPPP Data Model

Figure 3

Note that the attributes whose names end with the character * are mandatory attributes.

3.2.2. Data Model Objects and Attributes

The objects and attributes that comprise the data model can be described as follows (objects listed from the bottom up):

- Public Identifier (publicIdentifier):
  A public identifier is a well known attribute that is often used to perform lookup functions. For the purposes of this document, a Public Identifier can be an email address, a telephone number, a range of telephone numbers or a PSTN Routing Number (RN).

  A Destination Group may be associated with a Public Identifier to create a logical grouping and share a common set of Routes.

  A Public Identifier may optionally be associated with zero or more individual route records. This ability for a Public Identifier to be directly associated with a set of routes (e.g. target URI), as opposed to being associated with a Destination Group, supports the use cases where the target URI contains data specifically tailored to an individual Public Identifier.

- Telephone Number Range (TNRType, tn, endTn):
  A public identifier may represent an inclusive range of telephone numbers. The TN range is defined by the first and last telephone number of the inclusive range. For example, a TN range of (tn=12125550000, endTn=12125560000) means all the TNs from 12125550000 to 12125560000 are included.

- Destination Group (dgName):
  A collection of zero or more Public Identifiers that are related to one or more Route Group relationships.

- Route Group (rteGrpName):
  A Route Group contains a set of route records (RteRecs) that are associated with Public Identifiers. To support the use cases defined in [I-D.ietf-drinks-usecases-requirements], this document defines the following types of RteRecs: NAPTRType, NSType, and URIType. To support the Look-Up Function resolution, it is assumed that the administrative domain will be defined as a URI and it can be expressed as a URIType or a NAPTRType.
  A Route Group can be either in or out of service (as indicated by 'isInService' attribute). It also contains a list of organizations that can query the object (peeringOrg) and have access to its content (sourceIdent).

- o Source Identity (SourceIdentType, sourceIdentLabels, sourceIdentScheme):
  In some scenarios, it is important to identify the source of a query. The source identity label is a character string that identifies the source of a resolution lookup and can be used for source-based routing. We define several ways of identifying the source: by IP address, by the source URI or a domain name.

- o Route Record (RteRecType):
  A Route Record is the data that the resolution systems return in response to a successful query with the Public Identifier as the query string. It is associated with a Route Group for routes that are not specific to a Public Identifier.
  To support the use cases defined in [I-D.ietf-drinks-usecases-requirements], SPPP protocol defines three type of Route Records: URIType, NAPTRType, and NSType. These Route Records extend the abstract type RteRecType and inherit the common attribute 'priority' that is meant for setting precedence across the route records defined within a Route Group in a protocol agnostic fashion.

- o Organization (OrgIdType):
  An Organization represents an entity that is authorized to access given data elements. All objects are associated with two organizations to identify the registrant and the registrar. An entity authorized to view a Route Group (typically a SSP peering partner) is identified a peering Organization (peeringOrg).

3.2.3. Applicability for LUF-only Data Provisioning

This section describes the data model for SPPP clients that only provision data for LUF resolution.

The purpose of LUF data provisioning is to provide the administrative domain given a destination group. As such, a client provisioning LUF-only data only needs to provide one or more route groups that contain a route group name and a URI for the target domain.

Note that source-based routing is supported: depending on what entity requests the look-up resolution (sourceIdent), a different URI may be returned by using different Route Groups.

Certain protocol operations could be added in future revisions of this document as "short-cuts" for LUF related data provisioning.

```
                        +-----------------------+
                        |Route Group:           |
                        |  rteGrpName*,         |
                        |  isInService,         |
                        |  rteRec*,             |
                        |  extension            |
                        |                       |
                        +-----------------------+
                                   ^
                                   |
                         +---------+------------+
                         |Destination           |
                         |Group:                |
                         |  dgName*,            |<----+
                         |  extension           |     |
                         +----------------------+     |
                                                      |
                                        +-------------+---------+
                                        |Public                 |
                                        |Identifier:            |
                                        |  publicIdentifier*,   |
                                        |  dgName*,             |
                                        |  extension            |
                                        +-----------------------+
```

LUF-only Data Model Example for SPPP

Figure 4

As an example, a request to add a route group where public identifiers resolve into the URI sip:ssp1.example.com during look-up resolution would be:

```
       <?xml version="1.0" encoding="UTF-8"?>
       <addRteGrpsRqst
              xmlns="urn:ietf:params:xml:ns:sppp:base:1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
               <clientTransId>id-12317123</clientTransId>
               <minorVer>20</minorVer>
               <rteGrp>
                   <base>
                       <rantId>registrantID123</rantId>
                       <rarId>registrarId0</rarId>
                   </base>
                   <rteGrpName>route_grp_1</rteGrpName>
                   <rteRec xsi:type="URIType">
                       <ere>^(.*)$</ere>
                       <uri>urn:ssp1.example.com</uri>
                   </rteRec>
                   <isInSvc>true</isInSvc>
               </rteGrp>
       </addRteGrpsRqst>
```

Figure 5

3.2.4. Applicability for LUF+LRF data Provisioning

This section provides a read-out of the data model for SPPP clients that provision data for both LUF and LRF resolution.

The purpose of LUF+LRF data provisioning is to provide a URI given a destination group as well as the location routing for that target domain. As such, a client provisioning LUF+LRF data provides one or more route groups that contain a route group name and a URI for the target domain and each route group is associated with a Route Record which can be in the form of a URI, NAPTR or NS resource record.

```
                   +-----------------------+
                   |Route Group:           |        +-----[abstract]-+
                   |  rteGrpName*,         |        |                |
                   |  isInService,         |        | Route Record:  |
                   |  targetDomain,        +------->|  rteRecName*,  |
                   |  extension            |        |  priority,     |
                   |                       |        |  extension     |
                   +-----------------------+        |                |
                              ^                     +----------------+
                              |
                    +---------+------------+
                    |Destination           |
                    |Group:                |
                    |  dgName*,            |<----+
                    |  extension           |     |
                    +----------------------+     |
                                                 |
                                   +-------------+-[abstract]-+
                                   |Public                    |
                                   |Identifier:               |
                                   |  publicIdentifier*,      |
                                   |  dgName*,                |
                                   |  extension               |
                                   +--------------------------+
```

LUF+LRF Data Model Example for SPPP for DRINKS WG Review

Figure 6

As an example, a request to add a route group where public identifiers resolve into the URI ssp1.example.com and NAPTR associated with that domain based on the source Organization would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<addRteGrpsRqst xmlns="urn:ietf:params:xml:ns:sppp:base:1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <clientTransId>id-12317123</clientTransId>
        <minorVer>20</minorVer>
        <rteGrp>
            <base>
                <rantId>registrantID123</rantId>
                <rarId>registrarId0</rarId>
            </base>
            <rteGrpName>route_grp_1</rteGrpName>
            <isInSvc>true</isInSvc>
            <rteRec xsi:type="URIType">
                <ere>^(.*)$</ere>
                <uri>urn:ssp1.example.com</uri>
            </rteRec>
            <isInSvc>true</isInSvc>
        </rteGrp>
</addRteGrpsRqst>
```

Figure 7

3.3. Common Attributes

This section defines common object attributes. The protocol exchanges and operations in SPPP take various parameters. Some of these are common to several objects.

Two organization roles have been identified in the use cases and in this protocol. A registrant is the organization or business entity that "owns" the object while a registrar is an entity that can provision an object.

3.4. Known Issues and Current Limitations of the Data Model

The data model described in Figure 3 does not yet address all of the requirements and use cases defined in [I-D.ietf-drinks-usecases-requirements].

This section will list known protocol issues to be addressed in future revisions.

4. Transport Protocol Requirements

This section provides requirements for transport protocols suitable for SPPP. More specifically, this section specifies the services, features, and assumptions that SPPP delegates to the chosen transport and envelope technologies.

Two different groups of use cases are specified in [I-D.ietf-drinks-usecases-requirements]. One group of use cases describes the provisioning of data by a client into a Registry (Section 3.1 of the above referenced document), while the other group describes the distribution of data into local data repositories (Section 3.2). The current version of this document focuses on the first set of use cases (client to registry provisioning).

These use cases may involve the provisioning of very small data sets like the modification or update of a single public identifier. Other provisioning operations may deal with huge datasets like the "download" of a whole local number portability database to a Registry.

As a result, a transport protocol for SPPP must be very flexible and accommodate various sizes of data set sizes.

For the reasons outlined above, it is conceivable that provisioning and distributing may use different transport protocols. This document focuses on the provisioning protocol.

A few topics remain open for discussion:

- o The ability to establish multiple connections between a client and server may be desirable. If so, we may want to specify the relation of transactions between the various connections.

- o Pipelining of requests is required at the SPPP protocol layer. It may have impacts at the transport level that need to be outlined.

- o Scope: the current scope of this effort is based upon having a connection oriented transport. Is there any need to support a transport protocol with asynchronous operation?

- o If it is required that responses arrive in the order of the requests, this must be specified clearly.

4.1. Connection Oriented

The SPPP protocol follows a model where a Client establishes a connection to a Server in order to further exchange provisioning transactions over such point-to-point connection. A transport protocol for SPPP MUST therefore be connection oriented.

Note that the role of the "Client" and the "Server" only applies to the connection, and those roles are not related in any way to the type of entity that participates in a protocol exchange. For example, a Registry might also include a "Client" when such a Registry initiates a connection (for example, for data distribution to SSP).

4.2. Request & Response Model

Provisioning operations in SPPP follow the request - response model, where a transaction is initiated by a Client using a Request command, and the Server responds to the Client by means of a Response.

Multiple subsequent request-response exchanges MAY be performed over a single connection.

Therefore, a transport protocol for SPPP MUST follow the request-response model by allowing a response to be sent to the request initiator.

4.3. Connection Lifetime

Some use cases involve provisioning a single request to a network element - connections supporting such provisioning requests might be short-lived, and only established on demand.

Other use cases involve either provisioning a huge set of data, or a constant stream of small updates, which would require long-lived connections.

Therefore, a protocol suitable for SPPP SHOULD support short lived as well as long lived connections.

4.4. Authentication

Many use cases require the Server to authenticate the Client, and potentially also the Client to authenticate the Server. While authentication of the Server by the Client is expected to be used only to prevent impersonation of the Server, authentication of the Client by the Server is expected to be used to identify and further authorize the Client to certain resources on the Server.

Therefore, an SPPP transport protocol MUST provide means for a Server to authenticate and authorize a Client, and MAY provide means for Clients to authenticate a Server.

However, SPPP transport SHOULD also allow for unauthenticated connections.

4.5. Confidentiality & Integrity

Data that is transported over the protocol is deemed confidential. Therefore, a transport protocol suitable for SPPP MUST ensure confidentiality and integrity protection by providing encryption capabilities.

Additionally, a DRINKS protocol MUST NOT use an unreliable lower-layer transport protocol that does not provide confidentiality and integrity protection.

4.6. Near Real Time

Many use cases require near real-time responses from the Server. Therefore, a DRINKS transport protocol MUST support near-real-time response to requests submitted by the Client.

4.7. Request & Response Sizes

SPPP covers a range of use cases - from cases where provisioning a single public identifier will create very small request and response sizes to cases where millions of data records are submitted or retrieved in one transaction. Therefore, a transport protocol suitable for SPPP MUST support a great variety of request and response sizes.

A transport protocol MAY allow splitting large chunks of data into several smaller chunks.

4.8. Request and Response Correlation

A transport protocol suitable for SPPP MUST allow responses to be correlated with requests.

4.9. Request Acknowledgement

Data transported in the SPPP protocol is likely crucial for the operation of the communication network that is being provisioned.

Failed transactions can lead to situations where a subset of public identifiers (or even SSPs) might not be reachable, or situations

where the provisioning state of the network is inconsistent.

Therefore, a transport protocol for SPPP MUST provide a Response for each Request, so that a Client can identify whether a Request succeeded or failed.

4.10. Mandatory Transport

As of this writing of this revision, one transport protocol proposal has been provided in [I-D.ietf-drinks-sppp-over-soap].

This section will define a mandatory transport protocol to be compliant with this RFC.

5. XML Considerations

XML serves as the encoding format for SPPP, allowing complex hierarchical data to be expressed in a text format that can be read, saved, and manipulated with both traditional text tools and tools specific to XML.

XML is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document MUST be interpreted in the character case presented to develop a conforming implementation.

This section discusses a small number of XML-related considerations pertaining to SPPP.

5.1. Namespaces

All SPPP protocol elements are defined in the following namespace: urn:ietf:params:xml:ns:sppp:base:1

Namespace and schema definitions are used to identify both the base protocol schema and the schemas for managed objects.

5.2. Versioning

All XML instances SHOULD begin with an <?xml?> declaration to identify the version of XML that is being used, optionally identify use of the character encoding used, and optionally provide a hint to an XML parser that an external schema file is needed to validate the XML instance.

Conformant XML parsers recognize both UTF-8 (defined in [RFC3629]) and UTF-16 (defined in [RFC2781]); per [RFC2277] UTF-8 is the RECOMMENDED character encoding for use with SPPP.

Character encodings other than UTF-8 and UTF-16 are allowed by XML. UTF-8 is the default encoding assumed by XML in the absence of an "encoding" attribute or a byte order mark (BOM); thus, the "encoding" attribute in the XML declaration is OPTIONAL if UTF-8 encoding is used. SPPP clients and servers MUST accept a UTF-8 BOM if present, though emitting a UTF-8 BOM is NOT RECOMMENDED.

Example XML declarations:

```
<?xml?> version="1.0" encoding="UTF-8" standalone="no"?>
```

6. Request and Reply Model

An SPPP client interacts with an SPPP server by using one of the supported transport mechanisms to send one or more requests to the server and receive corresponding replies from the server. An SPPP request is wrapped within the <spppRequest> element while an SPPP reply is wrapped within an <spppReply> element. Furthermore, fully formed SPPP requests and replies are comprised of constructs required by the chosen transport technology, and the chosen envelope technology. The supported transport technology and envelope technology specifications will be defined in separate documents, and are not discussed here.

6.1. Request

An SPPP request object, common to any transport and envelope technology, is contained within the generic <spppRequest> element.

```
        <element name="spppRequest">
                 <complexType>
                          <sequence>
                                   <any/>
                          </sequence>
                 </complexType>
        </element>
```

Within any <spppRequest> element is the request object specific to the type of object(s) being operated on and the action(s) being performed on that object. For example, the addRteGroupRqst object, used to create Route Groups, that would be passed within an <spppRequest> is defined as follows:

```
        <element name="addRteGrpsRqst">
                                   <complexType>
            <sequence>
              <element name="basicRqst"
                type="spppb:BasicRqstType"/>
              <element name="rteGrp"
                type="spppb:RteGrpType"
                maxOccurs="unbounded"/>
            </sequence>
           </complexType>
        </element>
```

All update requests contain a BasicRqstType object. This object is defined as follows:

```
<complexType name="BasicRqstType">
 <sequence>
  <element name="clientTransId "
           type="spppb:TransIdType"/>
  <element name="minorVer"
           type="spppb:MinorVerType"/>
  <element name="ext"
           type="spppb:ExtAnyType"
           minOccurs="0"/>
 </sequence>
</complexType>




<simpleType name="TransIdType">
    <restriction base="string"/>
</simpleType>




<simpleType name="MinorVerType">
    <restriction base="unsignedLong"/>
</simpleType>
```

The data elements within the BasicRqstType object are primarily "house keeping" data elements. They are described as follows:

- o clientTransId: The client generated transaction ID that identifies this request for tracking purposes. This value is also echoed back to the client in the response. This value will not be checked for uniqueness.

- o minorVer: This identifies the minor version of the SPPP API that the client is attempting to use. This is used in conjunction with the major version identifier in the XML namespace. Refer to the Versioning section of this document for more detail.

- o ext: This is the standard extension element for this object. Refer to the Extensibility section of this document for more details.

6.2. Reply

An SPPP reply object, common to any transport and envelope technology, is contained within the generic <spppReply> element.

```
        <element name="spppReply">
                 <complexType>
                          <sequence>
                                   <any/>
                          </sequence>
                 </complexType>
        </element>
```

Within any <spppReply> element is the reply object containing the result of the request. All create, update, and delete operations result in a common response object structure, defined as follows:

```
       <element name="cmnRspns">
          <complexType>
             <sequence>
                <element name="rspns" type="spppb:BasicRspnsType"/>
             </sequence>
          </complexType>
       </element>
```

```
        <complexType name="BasicRspnsType">
          <sequence>
           <element name="clientTransId"
                        type="TransIdType"/>
           <element name="serverTransId"
                        type="TransIdType"/>
           <element name="resCode"
                        type="int"/>
           <element name="resMsg"
                        type="string"/>
           <element name="ext"
                        type="spppb:ExtAnyType"
                        minOccurs="0"/>
          </sequence>
                                </complexType>
```

The data elements within the BasicRspnseType object are described as follows:

o clientTransId: The echoed back client transaction ID that explicitly identifies this request for tracking purposes. This value is not guaranteed to be unique.

o serverTransId: The server transaction ID that identifies this request for tracking purposes. This value is guaranteed to be unique.

o resCode: The response code that explicitly identifies the result of the request. See the Response Code section for further details.

o resMsg: The human readable response message that accompanies the response code. See the Response Code section for further details.

o ext: This is the standard extension element for this object. Refer to the Extensibility section for more details.

7. Response Codes and Messages

This section contains an initial listing of response codes and their corresponding human readable text.

The response code numbering scheme generally adheres to the theory formalized in section 4.2.1 of [RFC2821]:

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

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

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

Result Code	Text
1000	Request Succeeded.
2001	Request syntax invalid.
2002	Request too large.
2003	Version not supported.
2103	Command invalid.
2104	Attribute value invalid: [ObjectTypeName]:[Object's rantId]:[Object's name]:{[Embedded ObjecTypeName]}:[attribute name]:[attribute value].
2105	Object does not exist: [ObjecTypeName]:[Object's rantId]:[Object's name].
2106	Object status or ownership does not allow for operation: [OperationName]:[ObjecTypeName]:[Object's rantId]:[Object's name].
2301	System temporarily unavailable.
2302	Unexpected internal system or server error.

Table 1: Response Codes Numbering Scheme and Messages

Some response messages are "parameterized" with one or more of the following parameters: "attribute name", "attribute value", "objectType-objectId", and "operation name".

The use of these parameters MUST adhere to the following rules:

- All parameters within a response message are mandatory and MUST be present. Parameters within a response message MUST NOT be left empty.

- Any value provided for the "attribute name" parameter MUST be an exact element name of the protocol data element that the response message is referring to. For example, valid values for "attribute name" are "destGrpName", "rteGrpName", etc.

- A value provided for the "command/request type" parameter MUST be an exact request object name that the response message is referring to. For example, a valid value for "request object name" is "delRteGrpsRqst".

- The value for "attribute value" MUST be the value of the data element to which the preceding "attribute name" refers.

- Result code 2104 SHOULD be used whenever an element value does not adhere to data validation rules.

- Result codes 2104 and 2105 MUST NOT be used interchangeably. Response code 2105 SHOULD be returned when the data element(s) used to uniquely identify a pre-existing object do not exist. If the data elements used to uniquely identify an object are malformed, then response code 2104 SHOULD be returned.

8. Protocol Commands

This section provides a preliminary list of SPPP protocol commands. At this early stage of the protocol development, the commands are only listed with a brief description.

8.1. Add Route Group Operation

As described in the introductory sections, a Route Group represents a combined grouping of Route Records that define route information, Destination Groups that contain a set of Public Identifiers with common routing information, and the list of peer organizations that have access to these public identifiers using this route information. It is this indirect linking of public identities to route information that significantly improves the scalability and manageability of the peering data. Additions and changes to routing information are reduced to a single operation on a Route Group, rather than millions of data updates to individual public identity records that individually contain their peering point data.

The addRteGrpsRqst operation creates or overwrites one or more Route Group objects. If a Route Group with the given name and registrant ID does not exist, then the server MUST create the Route Group. If a Route Group with the given name and registrant does exist, then the server MUST replace the current properties of the Route Group with the properties passed into the addRteGrpsRqst operation. The XSD declarations of the operation request object are as follows:

```
<element name="addRteGrpsRqst" type="spppb:AddRteGrpsRqstType"/>
      <complexType name="AddRteGrpsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="rteGrp" type="spppb:RteGrpType"
                                        maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
</complexType>
```

The element passed into the spppRequest element for this operation is the addRteGrpsRqst element. This element is of type AddRteGrpsRqstType, which extends BasicRqstType and contains one or more RteGrpType objects. Any limitation on the maximum number of RteGrpType objects that may be passed into this operation is a policy

decision and is not limited by the protocol. The RteGrpType object structure is defined as follows:

```
    <complexType name="RteGrpType">
      <sequence>
        <element name="base" type="spppb:BasicObjType"/>
        <element name="rteGrpName" type="spppb:ObjNameType"/>
        <element name="rteRec" type="spppb:RteRecType"
                        minOccurs="0" maxOccurs="unbounded"/>
        <element name="dgName" type="spppb:ObjNameType"
                        minOccurs="0" maxOccurs="unbounded"/>
        <element name="peeringOrg" type="spppb:OrgIdType"
                        minOccurs="0" maxOccurs="unbounded"/>
        <element name="sourceIdent" type="spppb:SourceIdentType"
                        minOccurs="0" maxOccurs="unbounded"/>
        <element name="isInSvc" type="boolean"/>
                <element name="ext" type="spppb:ExtAnyType"
s="0"/>
      </sequence>
    </complexType>
```

The RteGrpType object is composed of the following elements:

- o   base: As described in previous sections, most objects contain exactly one instance of BasicObjType which contains the ID of the registrant organization that owns this object and the ID of the registrar organization that provisioned this object.

- o   rteGrpName: The character string that contains the name of the Route Group. It uniquely identifies this object within the context of the registrant ID (a child element of the base element as described above).

- o   rteRec: Set of zero or more objects of type RteRecType that house the routing information, sometimes referred to as SED, that the RteGrpType object contains.

- o   dgName: Set of zero or more names of DestGrpType object instances. Each dgName name, in association with this Route Group's registrant ID, uniquely identifies a DestGrpType object instance whose public identities are reachable using the routing information housed in this Route Group.

- o   peeringOrg: Set of zero or more peering organization IDs that have accepted an offer to receive this Route Group's

information. The set of peering organizations in this list is not directly settable or modifiable using the addRteGrpsRqst operation. This set is instead controlled using the route offer and accept operations.

- sourceIdent: Set of zero or more SourceIdentType object instances. These objects, described further below, house the source identification schemes and identifiers that are applied at resolution time as part of source based routing algorithms for the Route Group.

- isInSvc: A boolean element that defines whether this Route Group is in service. The routing information contained in a Route Group that is in service is a candidate for inclusion in resolution responses for public identities residing in the Destination Group associated with this Route Group. The routing information contained in a Route Group that is not in service is not a candidate for inclusion is resolution responses.

- ext: Point of extensibility described in a previous section of this document.

As described above, the Route Group contains a set of RteRecType objects. The RteRecType object is an abstract type. The concrete types that use RteRecType as an extension base are NAPTRType, NSType, and URIType. The definitions of these types are included below. The NAPTRType object is comprised of the data elements necessary for a NAPTR that contains routing information the Route Group. The NSType object is comprised of the data elements necessary for a Name Server that points to another DNS server that contains the desired routing information. The URIType object is comprised of the data elements necessary to house a URI.

```
<complexType name="RteRecType" abstract="true">
  <sequence>
    <element name="priority" type="positiveInteger"
                                    default="100"/>
  </sequence>
</complexType>

<complexType name="NAPTRType">
  <complexContent>
    <extension base="spppb:RteRecType">
      <sequence>
        <element name="order" type="unsignedShort"/>
        <element name="pref" type="unsignedShort"/>
```

```
              <element name="flags" type="string" minOccurs="0"/>
              <element name="svcs" type="string"/>
              <element name="regx" type="spppb:RegexParamType"
                                              minOccurs="0"/>
              <element name="repl" type="string" minOccurs="0"/>
              <element name="ttl" type="positiveInteger"
                                              minOccurs="0"/>
              <element name="ext" type="spppb:ExtAnyType"
                                              minOccurs="0"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>

      <complexType name="NSType">
        <complexContent>
          <extension base="spppb:RteRecType">
            <sequence>
              <element name="hostName" type="string"/>
              <element name="ipAddr" type="spppb:IPAddrType"
                      minOccurs="0" maxOccurs="unbounded"/>
              <element name="ttl" type="positiveInteger"
                                              minOccurs="0"/>
              <element name="ext" type="spppb:ExtAnyType"
                                              minOccurs="0"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>

      <complexType name="IPAddrType">
        <sequence>
          <element name="addr" type="string"/>
          <element name="type" type="spppb:IPType"/>
          <element name="ext" type="spppb:ExtAnyType"
                                              minOccurs="0"/>
        </sequence>
      </complexType>

      <simpleType name="IPType">
        <restriction base="token">
          <enumeration value="IPv4"/>
          <enumeration value="IPv6"/>
        </restriction>
      </simpleType>

      <complexType name="URIType">
        <complexContent>
```

```
          <extension base="spppb:RteRecType">
            <sequence>
              <element name="ere" type="string" default="^(.*)$"/>
              <element name="uri" type="string"/>
              <element name="ext" type="spppb:ExtAnyType"
   minOccurs="0"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
```

The NAPTRType object is composed of the following elements:

- order: Order value in an ENUM NAPTR, relative to other NAPTRType objects in the same Route Group.

- pref: Preference value in an ENUM NAPTR.

- svcs: ENUM service(s) that are served by the SBE. This field's value must be of the form specified in RFC 3761 (e.g., E2U+ pstn:sip+sip). The allowable values are a matter of policy and not limited by this protocol.

- regx: NAPTR's regular expression field. If this is not included then the Repl field must be included.

- repl: NAPTR replacement field, should only be provided if the Regex field is not provided, otherwise it will be ignored by the server.

- ttl: Number of seconds that an addressing server may cache this NAPTR.

- ext: Point of extensibility described in a previous section of this document.

The NSType object is composed of the following elements:

- hostName: Fully qualified host name of the name server.

- ipAddr: Zero or more objects of type IpAddrType. Each object holds an IP Address and the IP Address type, IPv4 or IP v6.

- ttl: Number of seconds that an addressing server may cache this Name Server.

o   ext: Point of extensibility described in a previous section of
     this document.

The URIType object is composed of the following elements:

o   ere: The POSIX Extended Regular Expression (ere) as defined in
     [RFC3986]

o   uri: the URI as defined in [RFC3986]

The RteGrpType object provides support for source-based routing via the source identity element. The source-based routing criteria provides the ability to specify zero or more of the following in association with a given Route Group: a regular expression that is matched against the resolution client IP address, a regular expression that is matched against the root domain name(s), and/or a regular expression that is matched against the calling party URI(s). The result will be that, after identifying the visible Route Groups whose associated Destination Group(s) contain the lookup key being queried, the resolution server will evaluate the characteristics of the Source URI, and Source IP address, and root domain of the lookup key being queried. The resolution server compares these criteria against source based routing criteria associated with the Route Groups. The routing information contained in Route Groups that have source based routing criteria will only be included in the resolution response if one or more of the criteria matches the source criteria from the resolution request.

```
   <complexType name="SourceIdentType">
     <sequence>
       <element name="sourceIdentRegex" type="string"/>
       <element name="sourceIdentScheme"
                                 type="spppb:SourceIdentSchemeType"/>
       <element name="ext" type="spppb:ExtAnyType"
                                           minOccurs="0"/>
     </sequence>
   </complexType>

   <simpleType name="SourceIdentSchemeType">
     <restriction base="token">
       <enumeration value="uri"/>
       <enumeration value="ip"/>
       <enumeration value="rootDomain"/>
     </restriction>
   </simpleType>
```

The SourceIdentType object is composed of the following data elements:

- o sourceIdentScheme: The source identification scheme that this source identification criteria applies to and that the associated sourceIdentRegex should be matched against.

- o sourceIdentRegex: The regular expression that should be used to test for a match against the portion of the resolution request that is dictated by the associated sourceIdentScheme.

- o ext: Point of extensibility described in a previous section of this document.

The result of the addRteGrpsRqst operation is the addRteGrpsRspns element defined below. As with all SPPP requests, the result is all-or-nothing. If more than one RteRecType is passed into this request, then they will either all succeed or all fail. In the case of failure, the failure response code(s) and message(s) will indicate the reason for the failure and the object(s) that caused the failure.

```
<element name="addRteGrpsRspns" type="spppb:BasicRspnsType"/>
```

The response codes that the addRteGrpsRqst operation can return are as follows:

- o 1000: Request Succeeded.

- o 2001: Request syntax invalid.

- o 2002: Request too large.

- o 2003: Version not supported.

- o 2103: Command invalid.

- o 2104: Attribute value invalid.

- o 2105: Object does not exist.

- o 2106: Object status or ownership does not allow for request.

- o 2301: System temporarily unavailable.

o 2302: Unexpected internal system or server error.

8.2. Get Route Groups Operation

The getRteGrpsRqst operation allows a client to get the properties of Route Group objects that a registrar organization is authorized to view. The server will attempt to find a Route Group object that has the registrant ID and route group name pair contained in each ObjKeyType object instance. If the set of ObjKeyType objects is empty then the server will return the list of Route Group objects that the querying client has the authority to view. If there are no matching Route Groups found then an empty result set will be returned.

The element passed into the spppRequest element for this operation is the getRteGrpsRqst element. This element is of type GetRteGrpsRqstType, which extends BasicRqstType and contains zero or more ObjKeyType objects. Any limitation on the maximum number of objects that may be passed into or returned by this operation is a policy decision and not limited by the protocol. The XSD declaration of the operation is as follows:

```
     <element name="getRteGrpsRqst" type="spppb:GetRteGrpsRqstType"/>

    <complexType name="GetRteGrpsRqstType">
      <complexContent>
        <extension base="spppb:BasicRqstType">
          <sequence>
            <element name="objectKey" type="spppb:ObjKeyType"
                             minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
```

The result of the getRteGrpsRqst operation returned in the spppResponse element is the getRteGrpsRspns element defined below. This object contains the resulting set of RteGrpType objects, or an empty set if there were no matches.

```
    <element name="getRteGrpsRspns"
type="spppb:GetRteGrpsRspnsType"/>

    <complexType name="GetRteGrpsRspnsType">
      <complexContent>
        <extension base="spppb:BasicRspnsType">
          <sequence>
            <element name="rteGrp" type="spppb:RteGrpType"
                 minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
```

The response codes that the getRteGrpsRqst operation can return are as follows:

- 1000: Request Succeeded.
- 2001: Request syntax invalid.
- 2002: Request too large.
- 2003: Version not supported.
- 2103: Command invalid.
- 2104: Attribute value invalid.
- 2301: System temporarily unavailable.
- 2302: Unexpected internal system or server error.

8.3. Add Route Group Offers Operation

The list of peer organizations whose resolution responses can include the routing information contained in a given Route Group is controlled by the organization to which a Route Group object belongs, its registrant, and the peer organization that submits resolution requests, a data recipient or peering organization. The registrant offers access to a Route Group by submitting a Route Group Offer and the data recipient can then accept or reject that offer. Not until access to a Route Group has been offered and accepted will the data recipient's organization ID be included in the peeringOrg list in a Route Group object, and that Route Group's peering information become a candidate for inclusion in the responses to the resolution requests submitted by that data recipient. The addRteGrpOffersRqst operation

creates or overwrites one or more Route Group Offer objects. If a Route Group Offer for the given Route key (route name and registrant ID) and offeredToOrg ID does not exist, then the server creates the Route Group Offer object. If a such a Route Group Offer does exist, then the server replaces the current object with the new object. The XSD declarations of the operation request object are as follows:

```
<element name="addRteGrpOffersRqst"
                            type="spppb:AddRteGrpOffersRqstType"/>

<complexType name="AddRteGrpOffersRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrpOffer" type="spppb:RteGrpOfferType"
                                               maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The element passed into the spppRequest element for this operation is the addRteGrpOffersRqst element. This element is of type AddRteGrpOffersRqstType, which extends BasicRqstType and contains one or more RteGrpOfferType objects. Any limitation on the maximum number of objects that may be passed into or returned by this operation is a policy decision and not limited by the protocol. The XSD declaration of the operation is as follows:

```
    <complexType name="RteGrpOfferType">
      <sequence>
        <element name="base" type="spppb:BasicObjType"/>
        <element name="rteGrpOfferKey" type="spppb:RteGrpOfferKeyType"/>
        <element name="status" type="spppb:RteGrpOfferStatusType"/>
        <element name="offerDateTime" type="dateTime"/>
        <element name="acceptDateTime" type="dateTime" minOccurs="0"/>
        <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
      </sequence>
    </complexType>
       <complexType name="RteGrpOfferKeyType">
      <sequence>
        <element name="rteGrpKey" type="spppb:ObjKeyType"/>
        <element name="offeredTo" type="spppb:OrgIdType"/>
      </sequence>
    </complexType>

    <simpleType name="RteGrpOfferStatusType">
      <restriction base="token">
        <enumeration value="offered"/>
        <enumeration value="accepted"/>
      </restriction>
    </simpleType>
```

The RteGrpOfferType object is composed of the following elements:

- base: As described in previous sections, most objects contain exactly one instance of BasicObjType which contains the ID of the registrant organization that owns this object and the ID of the registrar organization that provisioned this object.

- rteGrpOfferKey: The object that identifies the route that is or has been offered and the organization that it is or has been offered to. The combination of these three data elements uniquely identify a Route Group Offer.

- status: The status of the offer, offered or accepted. This status is controlled by the server. It is automatically set to "offered" when ever a new Route Group Offer is added, and is automatically set to "accepted" if and when that offer is accepted. The value of the element is ignored when passed in by the client.

- offerDateTime: Date and time in GMT when the Route Group Offer was added.

o acceptDateTime: Date and time in GMT when the Route Group Offer was accepted.

The result of addRteGrpOffersRqst is the addRteGrpOffersRspns element defined below. As with all SPPP requests, the result is all-or-nothing. If more than one RteGrpOfferType is passed into this request, then they will either all succeed or all fail. In the case of failure, the failure response code(s) and message(s) will indicate the reason for the failure and the object(s) that caused the failure.

```
  <element name="addRteGrpOffersRspns" type="spppb:BasicRspnsType"/>
```

The response codes that the addRteGrpOffersRqst operation can return are as follows:

o 1000: Request Succeeded.

o 2001: Request syntax invalid.

o 2002: Request too large.

o 2003: Version not supported.

o 2103: Command invalid.

o 2104: Attribute value invalid.

o 2105: Object does not exist.

o 2106: Object status or ownership does not allow for request.

o 2301: System temporarily unavailable.

o 2302: Unexpected internal system or server error.

8.4. Accept Route Group Offers Operation

Not until access to a Route Group has been offered and accepted will the data recipient's organization ID be included in the peeringOrg list in that Route Group object, and that Route Group's peering information become a candidate for inclusion in the responses to the resolution requests submitted by that data recipient.The acceptRteGrpOffersRqst operation is called by, or on behalf of, the data recipient to accept one or more Route Group Offers that are pending in the "offered" status for the data recipient's organization

ID. If a Route Group Offer for the given Route Group Offer key (route name, route registrant ID, data recipient's organization ID) exists, then the server moves the Route Group Offer to the "accepted" status and adds that data recipient's organization ID into the list of peerOrgIds for that Route Group. If a such a Route Group Offer does not exist, then the server returns the appropriate error code 2105. The XSD declarations for the operation request object are as follows:

```
   <element name="acceptRteGrpOffersRqst"
                            type="spppb:AcceptRteGrpOffersRqstType"/>

   <complexType name="AcceptRteGrpOffersRqstType">
     <complexContent>
       <extension base="spppb:BasicRqstType">
         <sequence>
           <element name="rteGrpOfferKey"
              type="spppb:RteGrpOfferKeyType" maxOccurs="unbounded"/>
         </sequence>
       </extension>
     </complexContent>
   </complexType>
```

The element passed into the spppRequest element for this operation is the acceptRteGrpOffersRqst element. This element is of type AcceptRteGrpOffersRqstType, which extends BasicRqstType and contains one or more RteGrpOfferKeyType objects. Any limitation on the maximum number of objects that may be passed into or returned by this operation is a policy decision and not limited by the protocol.

The result of acceptRteGrpOffersRqst is the acceptRteGrpOffersRspns element defined below. As with all SPPP requests, the result is all-or-nothing. If more than one RteGrpOfferKeyType is passed into this request, then they will either all succeed or all fail. In the case of failure, the failure response code(s) and message(s) will indicate the reason for the failure and the object(s) that caused the failure.

```
   <element name="acceptRteGrpOffersRspns"
                                        type="spppb:BasicRspnsType"/>
```

The response codes that the acceptRteGrpOffersRspns operation can return are as follows:

- o 1000: Request Succeeded.
- o 2001: Request syntax invalid.
- o 2002: Request too large.
- o 2003: Version not supported.
- o 2103: Command invalid.
- o 2104: Attribute value invalid.
- o 2105: Object does not exist.
- o 2106: Object status or ownership does not allow for request.
- o 2301: System temporarily unavailable.
- o 2302: Unexpected internal system or server error.

8.5. Reject Route Group Offers Operation

Not until access to a Route Group has been offered and accepted will the data recipient's organization ID be included in the peeringOrg list in that Route Group object, and that Route Group's peering information become a candidate for inclusion in the responses to the resolution requests submitted by that data recipient. However, the data recipient that the Route Group has been offered to has the option of rejecting a Route Group Offer that has been offered but not accepted or that has been offered and accepted. The rejectRteGrpOffersRqst operation is used for these purposes and is called by, or on behalf of, the data recipient to accept one or more Route Group Offers that are pending in the "offered" status or the "accepted" status for the data recipient's organization ID. If a Route Group Offer for the given Route Group Offer key (route name, route registrant ID, data recipient's organization ID) exists in either the offered or accepted status, then the server deletes that Route Group Offer object , and, if appropriate, removes the data recipients organization ID from the list of peerOrgIds for that Route Group. If the Route Group Offer does not exist, then the server returns the appropriate error code 2105. The XSD declarations for the operation request object are as follows:

```
   <element name="rejectRteGrpOffersRqst"
                         type="spppb:RejectRteGrpOffersRqstType"/>

   <complexType name="RejectRteGrpOffersRqstType">
     <complexContent>
       <extension base="spppb:BasicRqstType">
         <sequence>
           <element name="rteGrpOfferKey"
              type="spppb:RteGrpOfferKeyType" maxOccurs="unbounded"/>
         </sequence>
       </extension>
     </complexContent>
   </complexType>
```

The element passed into the spppRequest element for this operation is the rejectRteGrpOffersRqst element. This element is of type RejectRteGrpOffersRqstType, which extends BasicRqstType and contains one or more RteGrpOfferKeyType objects. Any limitation on the maximum number of objects that may be passed into or returned by this operation is a policy decision and not limited by the protocol.

The result of rejectRteGrpOffersRqst is the rejectRteGrpOffersRspns element defined below. As with all SPPP requests, the result is all-or-nothing. If more than one RteGrpOfferKeyType is passed into this request, then they will either all succeed or all fail. In the case of failure, the failure response code(s) and message(s) will indicate the reason for the failure and the object(s) that caused the failure.

```
    <element name="rejectRteGrpOffersRspns"
                             type="spppb:BasicRspnsType"/>
```

The response codes that the rejectRteGrpOffersRspns operation can return are as follows:

o   1000: Request Succeeded.

o   2001: Request syntax invalid.

o   2002: Request too large.

o   2003: Version not supported.

o   2103: Command invalid.

- o 2104: Attribute value invalid.
- o 2105: Object does not exist.
- o 2106: Object status or ownership does not allow for request.
- o 2301: System temporarily unavailable.
- o 2302: Unexpected internal system or server error.

8.6. Get Route Group Offers Operation

The getRteGrpOffersRqst operation allows a client to get the properties of zero or more Route Group Offer objects that that registrar is authorized to view. The server will attempt to find Route Group Offer objects that has all the properties specified in the criteria passed into the operation. If no criteria is passed in then the server will return the list of Route Group Offer objects that the querying client has the authority to view. If there are no matching Route Group Offers found then an empty result set will be returned.

The element passed into the spppRequest element for this operation is the getRteGrpOffersRqst element. This element is of type GetRteGrpOffersRqstType, which extends BasicRqstType and contains the criteria that the returnedRoute Group Offer objects must match. Any limitation on the maximum number of objects that may be passed into or returned by this operation is a policy decision and not limited by the protocol. The XSD declaration of the operation is as follows:

```
      <element name="getRteGrpOffersRqst"
                                   type="spppb:GetRteGrpOffersRqstType"/>

      <complexType name="GetRteGrpOffersRqstType">
        <complexContent>
          <extension base="spppb:BasicRqstType">
            <sequence>
              <element name="offeredByPeers" type="boolean"
                                                          minOccurs="0"/>
              <element name="offeredToPeers" type="boolean"
                                                          minOccurs="0"/>
              <element name="status" type="spppb:RteGrpOfferStatusType"
                                                          minOccurs="0"/>
              <element name="peeringOrg" type="spppb:OrgIdType"
                                    minOccurs="0" maxOccurs="unbounded"/>
              <element name="rteGrpOfferKey"
  type="spppb:RteGrpOfferKeyType" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
```

The GetRteGrpOffersRqstType object is composed of the following elements:

o offeredByPeers: Zero or one boolean value that, if true, indicates that only offers that are offered by peering organizations to the querying registrant should be included in the result set. If this value is false, the offers by peering organizations to the querying registrant should not be included in the result set. The result set is also subject to other query criteria in the request.

o offeredToPeers: Zero or one boolean value that, if true, indicates that only offers that are offered to peering organizations by the querying registrant should be included in the result set. If this value is false, the offers to peering organizations by the querying registrant should not be included in the result set. The result set is also subject to other query criteria in the request.

o 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.

o peeringOrg: Zero or more organization IDs. Only offers that are offered to or 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.

o rteGrpOfferKey: Zero or more Route 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.

The result of the getRteGrpOffersRqst operation returned in the spppResponse element is the getRteGrpOffersRspns element defined below. This object contains the resulting set of RteGrpOfferType objects, or an empty set if there were no matches.

```
   <element name="getRteGrpOffersRspns"
                           type="spppb:GetRteGrpOffersRspnsType"/>

   <complexType name="GetRteGrpOffersRspnsType">
     <complexContent>
       <extension base="spppb:BasicRspnsType">
         <sequence>
           <element name="rteGrpOffer" type="spppb:RteGrpOfferType"
                             minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
       </extension>
     </complexContent>
   </complexType>
```

The response codes that the getRteGrpOffersRqst operation can return are as follows:

o 1000: Request Succeeded.

o 2001: Request syntax invalid.

o 2002: Request too large.

o 2003: Version not supported.

o 2103: Command invalid.

o 2104: Attribute value invalid.

- o 2301: System temporarily unavailable.

- o 2302: Unexpected internal system or server error.

8.7. Public Identifier Operations

Public Identifier is a well-known attribute that is used as the search key to find the routes associated with it. There are three types of public identifiers defined in this document: TNType for the telephone number, EmailType for the email address, and RNType for PSTN routing number. Further, TNRangeType is used to add a range of telephone numbers.

8.7.1. Add Public Identifier

addPubIdsRqst operation is used to create or overwrite one or more public identifier(s). When activating a new public identifier that can be reached using a common set of routes, it is often associated with a well-known destination group. In some cases, such as the email public identifier, the routing information is unique, and therefore, addPubIdsRqst allows the public identifier to be directly associated with a route record.

PubIdType in the schema represents the public identifier and it is defined as an abstract type. TNType, EmailType, and RNType, the concrete types of PubIdType, are inputs to 'addPubIdRqst' operation. The declaration of 'addPubIdsRqst' is as follows:

```
<element name="addPubIdsRqst" type="spppb:AddPubIdsRqstType"/>
<complexType name="AddPubIdsRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

For the 'addPubIdsRqst' operation to succeed, each public identifier should be associated with at least a valid destination group or a valid route type as defined within the PubIdType definition. If not, the provisioning server will deem the request a failure and return an appropriate failure code in the response.

TNType is a concrete public identifier that extends PubIdType definition. If the entity provisioning the telephone number is the carrier of record [see RFC 5067], then it SHOULD include the 'corClaim' element with a value 'true'. If the SPPP server records disagree with the COR claim of the provisioning entity, an appropriate failure response MUST be returned.

```
<complexType name="PubIdType" abstract="true">
  <sequence>
    <element name="base" type="spppb:BasicObjType"/>
    <element name="dgName" type="spppb:ObjNameType" minOccurs="0"/>
    <element name="rteRec" type="spppb:RteRecType" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="TNType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="tn" type="string"/>
        <element name="corClaim" type="spppb:CORInfoType"
          minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

For added flexibility, there is support to add a range of telephone numbers and associate them with a destination group. TNRType extends TNType and adds the 'endTn' attribute to mark the end of the range. In the TNRType context, the extended 'tn' attribute is used for the starting TN of a given telephone number range.

```
<complexType name="TNRType">
  <complexContent>
    <extension base="spppb:TNType">
      <sequence>
        <element name="endTn" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The element passed into the spppRequest element for this operation is the addPubIdsRqst element. This element is of type AddPubIdsRqstType, which extends BasicRqstType and contians one or more PubIdType objects. Any limitation on the maximum number of PubIdType objects that may be passed into this operatoin is a policy decision and is not limited by the protocol.

The response from the server is returned in addPubIdsRspns element. If more than one public identifiers are passed in the addPubIdsRqst, then a failure to add one will result in the failure of addPubIdsRqst operation. If the 'transactional' attribute is set to 'true' in the root element spppRequest and more than one operation request elements are included, then a failure of any one operation will result in the overall failure of spppRequest. In the case of a failure, the response code(s) and message(s) will indicate the reason of failure.

```
<element name="addRteGrpsRspns" type="spppb:BasicRspnsType"/>
```

The response codes that the addRteGrpsRqst operation can return are as follows:

o   1000: Request Succeeded.

o   2001: Request syntax invalid.

o   2002: Request too large.

o   2003: Version not supported.

o   2103: Command invalid.

o   2104: Attribute value invalid.

o   2105: Object does not exist.

o   2106: Object status or ownership does not allow for request.

o   2301: System temporarily unavailable.

o   2302: Unexpected internal system or server error.

8.7.2. Get Public Identifier

The getPubIdsRqst can be used by an authorized entity to obtain the properties of one or more public identifiers. In case of an authorization failure or if no matching public identifiers are found, an appropriate failure code will be returned.

To make a successful query, getPubIdsRqst element is set within the spppRequest root element. getPubIdsRqst is of type GetPubIdsRqstType, which extends from the common BasicRqstType.

```
      <element name="getPubIdsRqst" type="spppb:GetPubIdsRqstType"/>
      <complexType name="AddPubIdsRqstType">
        <complexContent>
          <extension base="spppb:BasicRqstType">
            <sequence>
              <element name="pi" type="spppb:PubIdType"
                maxOccurs="unbounded"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
```

The result of the getPubIdsRqst operation returned in the spppResponse element is the getPubIdsRspns element of type GetPubIdsRspnsType. If the matching record is found, getPubIdsRspns element will include one or more pi elements with destination group name and/or the route record associations.

```
    <element name="getPubIdsRspns" type="spppb:GetPubIdsRspnsType"/>
    <complexType name="GetPubIdsRspnsType">
      <complexContent>
        <extension base="spppb:BasicRspnsType">
          <sequence>
            <element name="pi" type="spppb:PubIdType" minOccurs="0"
              maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
```

The response codes that the addRteGrpsRqst operation can return are

as follows:

- o 1000: Request Succeeded.
- o 2001: Request syntax invalid.
- o 2002: Request too large.
- o 2003: Version not supported.
- o 2103: Command invalid.
- o 2104: Attribute value invalid.
- o 2105: Object does not exist.
- o 2106: Object status or ownership does not allow for request.
- o 2301: System temporarily unavailable.
- o 2302: Unexpected internal system or server error.

8.7.3. Delete Public Identifier

In order to remove the public identifier, an authorized entity can use the delPubIdsRqst operation. If the entity that issued the command is not authorized to perform this operation or if the public identifier doesn't exist, an appropriate error code will be returned in the response.

delPubIdsRqst element is set in the root spppRequest element. delPubIdsRqst element is of type DelPubIdsRqstType as shown below:

```
<element name="getPubIdsRqst" type="spppb:GetPubIdsRqstType"/>
<complexType name="DelPubIdsRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The result of the delPubIdsRqst operation returned in the spppResponse element is the getPubIdsRspns element of type GetPubIdsRspnsType.

```
        <element name="delPubIdsRspns" type="spppb:BasicRspnsType"/>
```

8.8. Egress Route Operations

8.8.1. Add Egress Route

8.8.2. Get Egress Route

8.8.3. Delete Egress Route

9. Security Considerations

The transport protocol section contains some security properties that the transport protocol must provide so that authenticated endpoints can exchange data confidentially and with integrity protection.

More details will be provided in a future revision of this document.

10. IANA Considerations

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

Two URI assignments are requested.

Registration request for the SPPP XML namespace:
urn:ietf:params:xml:ns:sppp:base:1
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

Registration request for the XML schema:
URI: urn:ietf:params:xml:schema:sppp:1
Registrant Contact: IESG
XML: See the "Formal Specification" section of this document (Section 11).

11. Formal Specification

This section provides the draft XML Schema Definition for the SPPP protocol. Please read Section 3.4 for known issues.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns:spppb="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:ietf:params:xml:ns:sppp:base:1"
  elementFormDefault="qualified" xml:lang="EN">
  <annotation>
    <documentation> ------------------ Object Type Definitions
      -------------- </documentation>
  </annotation>
  <complexType name="RteGrpType">
    <sequence>
      <element name="base" type="spppb:BasicObjType"/>
      <element name="rteGrpName" type="spppb:ObjNameType"/>
      <element name="rteRec" type="spppb:RteRecType" minOccurs="0"
        maxOccurs="unbounded"/>
      <element name="dgName" type="spppb:ObjNameType" minOccurs="0"
        maxOccurs="unbounded"/>
      <element name="peeringOrg" type="spppb:OrgIdType" minOccurs="0"
        maxOccurs="unbounded"/>
      <element name="sourceIdent" type="spppb:SourceIdentType"
        minOccurs="0" maxOccurs="unbounded"/>
      <element name="isInSvc" type="boolean"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <complexType name="DestGrpType">
    <sequence>
      <element name="base" type="spppb:BasicObjType"/>
      <element name="dgName" type="spppb:ObjNameType"/>
    </sequence>
  </complexType>
  <complexType name="PubIdType" abstract="true">
    <sequence>
      <element name="base" type="spppb:BasicObjType"/>
      <element name="dgName" type="spppb:ObjNameType" minOccurs="0"/>
      <element name="rteRec" type="spppb:RteRecType" minOccurs="0"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="EmailType">
```

```
    <complexContent>
      <extension base="spppb:PubIdType">
        <sequence>
          <element name="email" type="string"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="TNType">
    <complexContent>
      <extension base="spppb:PubIdType">
        <sequence>
          <element name="tn" type="string"/>
          <element name="corClaim" type="spppb:CORInfoType"
            minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="TNRType">
    <complexContent>
      <extension base="spppb:TNType">
        <sequence>
          <element name="endTn" type="string"/>
          <element name="corClaim" type="spppb:CORInfoType"
            minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="RNType">
    <complexContent>
      <extension base="spppb:PubIdType">
        <sequence>
          <element name="rn" type="string" default="true"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="NAPTRType">
    <complexContent>
      <extension base="spppb:RteRecType">
        <sequence>
          <element name="order" type="unsignedShort"/>
          <element name="pref" type="unsignedShort"/>
          <element name="flags" type="string" minOccurs="0"/>
          <element name="svcs" type="string"/>
          <element name="regx" type="spppb:RegexParamType"
```

```
            minOccurs="0"/>
          <element name="repl" type="string" minOccurs="0"/>
          <element name="ttl" type="positiveInteger" minOccurs="0"/>
          <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="NSType">
    <complexContent>
      <extension base="spppb:RteRecType">
        <sequence>
          <element name="hostName" type="string"/>
          <element name="ipAddr" type="spppb:IPAddrType" minOccurs="0"
            maxOccurs="unbounded"/>
          <element name="ttl" type="positiveInteger" minOccurs="0"/>
          <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="URIType">
    <complexContent>
      <extension base="spppb:RteRecType">
        <sequence>
          <element name="ere" type="string" default="^(.*)$"/>
          <element name="uri" type="string"/>
          <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="RteGrpOfferType">
    <sequence>
      <element name="base" type="spppb:BasicObjType"/>
      <element name="rteGrpOfferKey" type="spppb:RteGrpOfferKeyType"/>
      <element name="status" type="spppb:RteGrpOfferStatusType"/>
      <element name="offerDateTime" type="dateTime"/>
      <element name="acceptDateTime" type="dateTime" minOccurs="0"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <complexType name="EgrRteType">
    <sequence>
      <element name="base" type="spppb:BasicObjType"/>
      <element name="egrRteName" type="spppb:ObjNameType"/>
      <element name="pref" type="unsignedShort"/>
      <element name="svcs" type="string"/>
```

```
      <element name="regxRewriteRule" type="spppb:RegexParamType"/>
      <element name="ingressRte" type="spppb:ObjKeyType" minOccurs="0"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <annotation>
    <documentation> ------------------ Abstract Object and Element
      Type Definitions -------------- </documentation>
  </annotation>
  <complexType name="BasicObjType">
    <sequence>
      <element name="rantId" type="spppb:OrgIdType"/>
      <element name="rarId" type="spppb:OrgIdType"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <complexType name="RteRecType" abstract="true">
    <sequence>
      <element name="priority" type="positiveInteger" default="100"/>
    </sequence>
  </complexType>
  <complexType name="RegexParamType">
    <sequence>
      <element name="ere" type="string" default="^(.*)$"/>
      <element name="repl" type="string"/>
    </sequence>
  </complexType>
  <simpleType name="OrgIdType">
    <restriction base="string"/>
  </simpleType>
  <simpleType name="ObjNameType">
    <restriction base="string"/>
  </simpleType>
  <simpleType name="TransIdType">
    <restriction base="string"/>
  </simpleType>
  <simpleType name="MinorVerType">
    <restriction base="unsignedLong"/>
  </simpleType>
  <complexType name="ObjKeyType">
    <sequence>
      <element name="rantId" type="spppb:OrgIdType"/>
      <element name="name" type="spppb:ObjNameType"/>
    </sequence>
  </complexType>
  <complexType name="RteGrpOfferKeyType">
    <sequence>
      <element name="rteGrpKey" type="spppb:ObjKeyType"/>
```

```
      <element name="offeredTo" type="spppb:OrgIdType"/>
    </sequence>
  </complexType>
  <complexType name="BasicRqstType">
    <sequence>
      <element name="clientTransId" type="spppb:TransIdType"
        minOccurs="0"/>
      <element name="minorVer" type="spppb:MinorVerType" minOccurs="0"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <complexType name="BasicRspnsType">
    <sequence>
      <element name="clientTransId" type="spppb:TransIdType"
        minOccurs="0"/>
      <element name="serverTransId" type="spppb:TransIdType"/>
      <element name="resCode" type="int"/>
      <element name="resMsg" type="string"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
    <attribute name="clientTransId" type="spppb:TransIdType"/>
    <attribute name="serverTransId" type="spppb:TransIdType"/>
  </complexType>
  <complexType name="IPAddrType">
    <sequence>
      <element name="addr" type="string"/>
      <element name="type" type="spppb:IPType"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <simpleType name="IPType">
    <restriction base="token">
      <enumeration value="IPv4"/>
      <enumeration value="IPv6"/>
    </restriction>
  </simpleType>
  <complexType name="SourceIdentType">
    <sequence>
      <element name="sourceIdentLabel" type="string"/>
      <element name="sourceIdentScheme"
        type="spppb:SourceIdentSchemeType"/>
      <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
    </sequence>
  </complexType>
  <simpleType name="SourceIdentSchemeType">
    <restriction base="token">
      <enumeration value="uri"/>
      <enumeration value="ip"/>
```

```
      <enumeration value="rootDomain"/>
    </restriction>
  </simpleType>
  <complexType name="CORInfoType">
    <sequence>
      <element name="corClaim" type="boolean" default="true"/>
      <element name="corClaimApproved" type="boolean" default="false"
      />
    </sequence>
  </complexType>
  <complexType name="SvcMenuType">
    <sequence>
      <element name="serverStatus" type="spppb:ServerStatusType"/>
      <element name="majMinVersion" type="string"
        maxOccurs="unbounded"/>
      <element name="objURI" type="anyURI" maxOccurs="unbounded"/>
      <element name="extURI" type="anyURI" minOccurs="0"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <simpleType name="ServerStatusType">
    <restriction base="token">
      <enumeration value="inService"/>
      <enumeration value="outOfService"/>
    </restriction>
  </simpleType>
  <simpleType name="RteGrpOfferStatusType">
    <restriction base="token">
      <enumeration value="offered"/>
      <enumeration value="accepted"/>
    </restriction>
  </simpleType>
  <complexType name="ExtAnyType">
    <sequence>
      <any namespace="##other" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <annotation>
    <documentation> -------------- Operation Request and Response
      Object Type Definitions ------------ </documentation>
  </annotation>
  <complexType name="AddRteGrpsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="rteGrp" type="spppb:RteGrpType"
            maxOccurs="unbounded"/>
        </sequence>
```

```
      </extension>
    </complexContent>
  </complexType>
  <complexType name="DelRteGrpsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="objectKey" type="spppb:ObjKeyType"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetRteGrpsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="objectKey" type="spppb:ObjKeyType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetRteGrpsRspnsType">
    <complexContent>
      <extension base="spppb:BasicRspnsType">
        <sequence>
          <element name="rteGrp" type="spppb:RteGrpType" minOccurs="0"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AddDestGroupsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="destGrp" type="spppb:DestGrpType"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="DelDestGroupsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="objectKey" type="spppb:ObjKeyType"
```

```
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetDestGroupsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="objectKey" type="spppb:ObjKeyType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetDestGroupsRspnsType">
    <complexContent>
      <extension base="spppb:BasicRspnsType">
        <sequence>
          <element name="destGrp" type="spppb:DestGrpType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AddPubIdsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="pi" type="spppb:PubIdType"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="DelPubIdsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="pi" type="spppb:PubIdType"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetPubIdsRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
```

```
        <sequence>
          <element name="pi" type="spppb:PubIdType" minOccurs="0"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetPubIdsRspnsType">
    <complexContent>
      <extension base="spppb:BasicRspnsType">
        <sequence>
          <element name="pi" type="spppb:PubIdType" minOccurs="0"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AddRteGrpOffersRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="rteGrpOffer" type="spppb:RteGrpOfferType"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="DelRteGrpOffersRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="rteGrpOfferKey"
            type="spppb:RteGrpOfferKeyType" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="AcceptRteGrpOffersRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="rteGrpOfferKey"
            type="spppb:RteGrpOfferKeyType" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="RejectRteGrpOffersRqstType">
```

```
      <complexContent>
        <extension base="spppb:BasicRqstType">
          <sequence>
            <element name="rteGrpOfferKey"
              type="spppb:RteGrpOfferKeyType" maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="GetRteGrpOffersRqstType">
      <complexContent>
        <extension base="spppb:BasicRqstType">
          <sequence>
            <element name="offeredByPeers" type="boolean" minOccurs="0"/>
            <element name="offeredToPeers" type="boolean" minOccurs="0"/>
            <element name="status" type="spppb:RteGrpOfferStatusType"
              minOccurs="0"/>
            <element name="peeringOrg" type="spppb:OrgIdType"
              minOccurs="0" maxOccurs="unbounded"/>
            <element name="rteGrpOfferKey"
              type="spppb:RteGrpOfferKeyType" minOccurs="0"
              maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="GetRteGrpOffersRspnsType">
      <complexContent>
        <extension base="spppb:BasicRspnsType">
          <sequence>
            <element name="rteGrpOffer" type="spppb:RteGrpOfferType"
              minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="AddEgrRtesRqstType">
      <complexContent>
        <extension base="spppb:BasicRqstType">
          <sequence>
            <element name="egrRte" type="spppb:EgrRteType"
              maxOccurs="unbounded"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name="DelEgrRtesRqstType">
      <complexContent>
```

```
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="objectKey" type="spppb:ObjKeyType"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetEgrRtesRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType">
        <sequence>
          <element name="objectKey" type="spppb:ObjKeyType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetEgrRtesRspnsType">
    <complexContent>
      <extension base="spppb:BasicRspnsType">
        <sequence>
          <element name="rteGrp" type="spppb:RteGrpType" minOccurs="0"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GetSvcMenuRqstType">
    <complexContent>
      <extension base="spppb:BasicRqstType"/>
    </complexContent>
  </complexType>
  <complexType name="GetSvcMenuRspnsType">
    <complexContent>
      <extension base="spppb:BasicRspnsType">
        <sequence>
          <element name="svcMenu" type="spppb:SvcMenuType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <annotation>
    <documentation> -------------- Operation Request and Response
      Element Definitions ------------ </documentation>
  </annotation>
  <annotation>
    <documentation> -------------- Manage Route Groups
```

```
        </documentation>
      </annotation>
      <element name="addRteGrpsRqst" type="spppb:AddRteGrpsRqstType"/>
      <element name="delRteGrpsRqst" type="spppb:DelRteGrpsRqstType"/>
      <element name="getRteGrpsRqst" type="spppb:GetRteGrpsRqstType"/>
      <element name="addRteGrpsRspns" type="spppb:BasicRspnsType"/>
      <element name="delRteGrpsRspns" type="spppb:BasicRspnsType"/>
      <element name="getRteGrpsRspns" type="spppb:GetRteGrpsRspnsType"/>
      <annotation>
        <documentation> -------------- Manage Destination Groups
        </documentation>
      </annotation>
      <element name="addDestGroupsRqst" type="spppb:AddDestGroupsRqstType"/>
      <element name="delDestGroupsRqst" type="spppb:DelDestGroupsRqstType"/>
      <element name="getDestGroupsRqst" type="spppb:GetDestGroupsRqstType"/>
      <element name="addDestGroupsRspns" type="spppb:BasicRspnsType"/>
      <element name="delDestGroupsRspns" type="spppb:BasicRspnsType"/>
      <element name="getDestGroupsRspns"
        type="spppb:GetDestGroupsRspnsType"/>
      <annotation>
        <documentation> -------------- Manage Public Identifiers
        </documentation>
      </annotation>
      <element name="addPubIdsRqst" type="spppb:AddPubIdsRqstType"/>
      <element name="delPubIdsRqst" type="spppb:DelPubIdsRqstType"/>
      <element name="getPubIdsRqst" type="spppb:GetPubIdsRqstType"/>
      <element name="addPubIdsRspns" type="spppb:BasicRspnsType"/>
      <element name="delPubIdsRspns" type="spppb:BasicRspnsType"/>
      <element name="getPubIdsRspns" type="spppb:GetPubIdsRspnsType"/>
      <annotation>
        <documentation> -------------- Manage Route Group Offers
        </documentation>
      </annotation>
      <element name="addRteGrpOffersRqst"
        type="spppb:AddRteGrpOffersRqstType"/>
      <element name="delRteGrpOffersRqst"
        type="spppb:DelRteGrpOffersRqstType"/>
      <element name="acceptRteGrpOffersRqst"
        type="spppb:AcceptRteGrpOffersRqstType"/>
      <element name="rejectRteGrpOffersRqst"
        type="spppb:RejectRteGrpOffersRqstType"/>
      <element name="getRteGrpOffersRqst"
        type="spppb:GetRteGrpOffersRqstType"/>
      <element name="addRteGrpOffersRspns" type="spppb:BasicRspnsType"/>
      <element name="delRteGrpOffersRspns" type="spppb:BasicRspnsType"/>
      <element name="acceptRteGrpOffersRspns" type="spppb:BasicRspnsType"/>
      <element name="rejectRteGrpOffersRspns" type="spppb:BasicRspnsType"/>
      <element name="getRteGrpOffersRspns"
```

```
    type="spppb:GetRteGrpOffersRspnsType"/>
  <annotation>
    <documentation> -------------- Manage Egress Routes
    </documentation>
  </annotation>
  <element name="addEgrRtesRqst" type="spppb:AddEgrRtesRqstType"/>
  <element name="delEgrRtesRqst" type="spppb:DelEgrRtesRqstType"/>
  <element name="getEgrRtesRqst" type="spppb:GetEgrRtesRqstType"/>
  <element name="addEgrRtesRspns" type="spppb:BasicRspnsType"/>
  <element name="delEgrRtesRspns" type="spppb:BasicRspnsType"/>
  <element name="getEgrRtesRspns" type="spppb:GetEgrRtesRspnsType"/>
  <annotation>
    <documentation> -------------- Misc Operations </documentation>
  </annotation>
  <element name="getSvcMenuRqst" type="spppb:GetSvcMenuRqstType"/>
  <element name="getSvcMenuRspns" type="spppb:GetSvcMenuRspnsType"/>
  <annotation>
    <documentation> -------- Generic Request and Response Definitions
      --------------- </documentation>
  </annotation>
  <element name="spppRequest">
    <complexType>
      <sequence>
        <any maxOccurs="unbounded"/>
      </sequence>
      <attribute name="transactional" type="boolean" use="optional"/>
    </complexType>
  </element>
  <element name="spppResponse">
    <complexType>
      <sequence>
        <any maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

12. Specification Extensibility

The protocol defined in this specification is extensible. This section explains how to extend the protocol and what procedures are necessary to follow in order to ensure proper extensions.

13. Acknowledgments

This document is a result of various discussions held in the DRINKS working group and within the DRINKS protocol design team, which is comprised of the following individuals, in alphabetical order: Deborah A Guyton (Telcordia), Sumanth Channabasappa (CableLabs), Jean-Francois Mule (CableLabs), Kenneth Cartwright (TNSI), Manjul Maharishi (TNSI), David Schwartz (XConnect), and the co-chairs Richard Shockey and Alexander Mayrhofer (enum.at GmbH).

The authors of this document thank the following individuals for their advice, reviews and comments during the development of this protocol: Lisa Dusseault, "YOUR NAME HERE" -- send comments to drinks list.

14. References

14.1. Normative References

[I-D.ietf-drinks-sppp-over-soap]
Cartwright, K., "SPPP Over SOAP and HTTP", draft-ietf-drinks-sppp-over-soap-00 (work in progress), June 2010.

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

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2781] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

14.2. Informative References

[I-D.ietf-drinks-usecases-requirements]
Channabasappa, S., "DRINKS Use cases and Protocol Requirements", draft-ietf-drinks-usecases-requirements-03 (work in progress), May 2010.

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3761] Faltstrom, P. and M. Mealling, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", RFC 3761, April 2004.

[RFC4725] Mayrhofer, A. and B. Hoeneisen, "ENUM Validation

Architecture", RFC 4725, November 2006.

[RFC5486] Malas, D. and D. Meyer, "Session Peering for Multimedia Interconnect (SPEERMINT) Terminology", RFC 5486, March 2009.

Authors' Addresses

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

Email: jfm@cablelabs.com

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

Email: kcartwright@tnsi.com

Syed Wasim Ali
NeuStar
46000 Center Oak Plaza
Sterling, VA 20166
USA

Email: syed.ali@neustar.biz

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

Email: alexander.mayrhofer@enum.at