DRINKS	J-F. Mule	
Internet-Draft	CableLabs	
Intended status: Standards Track	K. Cartwright	
Expires: April 25, 2011	TNS	
	S. Ali	
	NeuStar	
	A. Mayrhofer	
	enum.at GmbH	
	October 22, 2010	

TOC

Session Peering Provisioning Protocol
draft-ietf-drinks-spprov-03

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.

Table of Contents

---

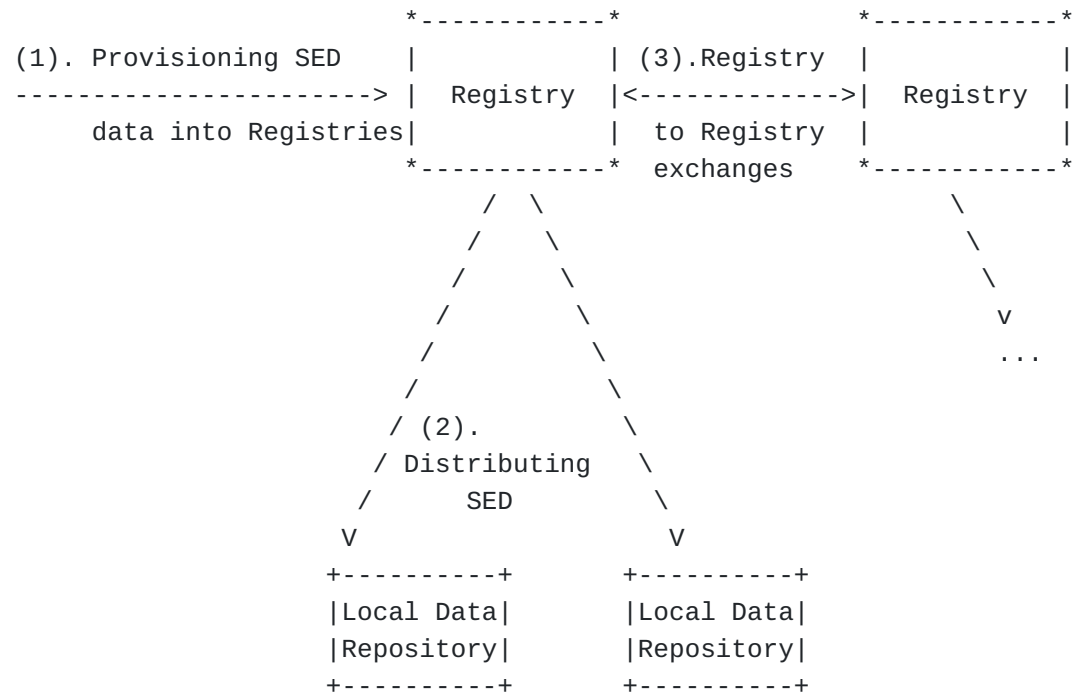
1. Introduction

TOC

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] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.). 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 "1" 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.



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] (Malas, D. and D. Meyer, "Session Peering for Multimedia Interconnect (SPEERMINT) Terminology," March 2009.) 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 Local Data Repository serves a Look-Up Function (LUF) to determine the target domain to assist in call routing (as described in [RFC5486] (Malas, D. and D. Meyer, "Session Peering for Multimedia Interconnect (SPEERMINT) Terminology," March 2009.)). 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. A Local Data Repository serves both a 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] (Cartwright, K., "SPPP Over SOAP and HTTP," June 2010.)) identifies a SOAP transport mechanism for SPPP.
This document is organized as follows:

*Section 2 (Terminology) provides the terminology;

*Section 3 (Protocol High Level Design) provides an overview of the SPPP protocol, including the layering approach, functional entities and data model;

*Section 4 (Transport Protocol Requirements) specifies requirements for SPPP transport protocols;

*Section 5 (Base Protocol Data Structures) describes the base protocol data structures including the request and response elements (Section 5.1 (Request and Response Structures)), the response codes and messages (Section 5.2 (Response Codes and Messages)) and the basic object type most first class objects extend from;

*Section 6 (Protocol Commands) and Section 7 (SPPP Examples) describe the main protocol commands and examples;

*Section 8 (XML Considerations) defines XML considerations that XML parsers must meet to conform to this specification;

*Section 11 (Formal Specification) normatively defines the SPPP protocol using its XML Schema Definition.

---

2. Terminology

TOC

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] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).
This document reuses terms from [RFC3261] (Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol," June 2002.), [RFC5486] (Malas, D. and D. Meyer, "Session Peering for Multimedia Interconnect (SPEERMINT) Terminology," March 2009.), use cases and requirements documented in [I-D.ietf-drinks-usecases-requirements] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.) and the ENUM Validation Architecture [RFC4725] (Mayrhofer, A. and B. Hoeneisen, "ENUM Validation Architecture," November 2006.).
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] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.)). 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] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.)).

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] (Mayrhofer, A. and B. Hoeneisen, "ENUM Validation Architecture," November 2006.). 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] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.), a Registrant is pictured as a SIP Service Provider in Figure 2.
A Registrant is identified by its name and an identifier in the data model.

Registrar: In this document, we also extend the definition of a Registrar from [RFC4725] (Mayrhofer, A. and B. Hoeneisen, "ENUM Validation Architecture," November 2006.). 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 and an identifier in the data model.

---

3. Protocol High Level Design

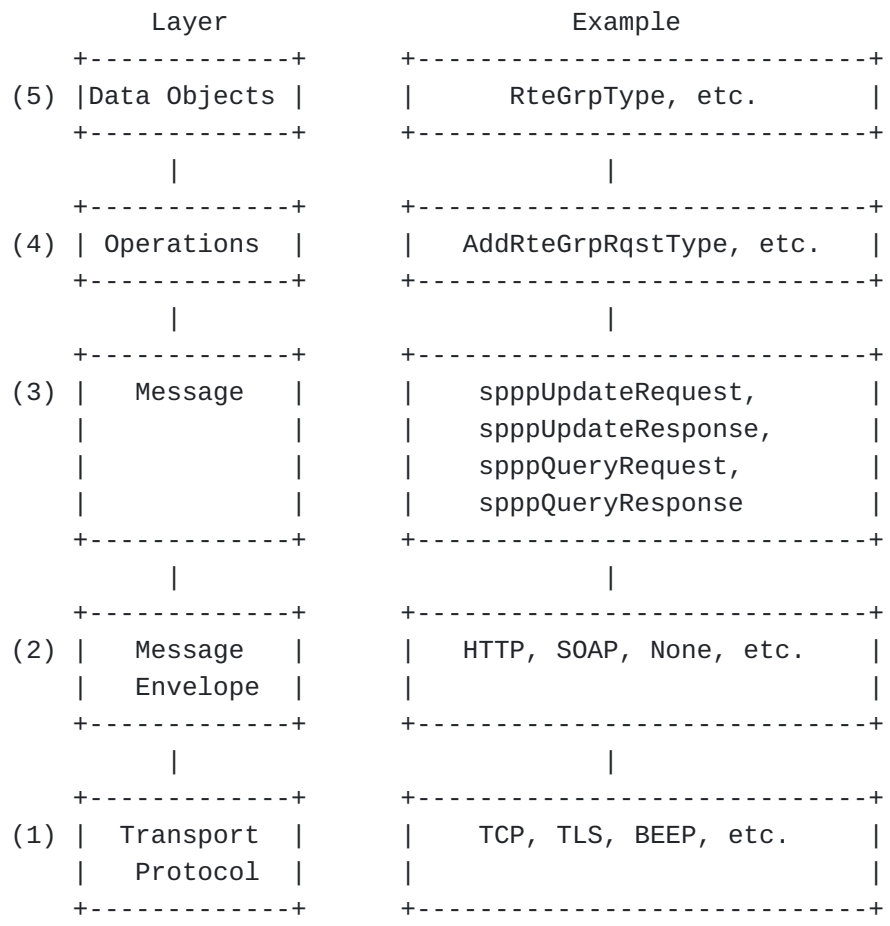
TOC

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.

TOC

3.1. Protocol 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.



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.

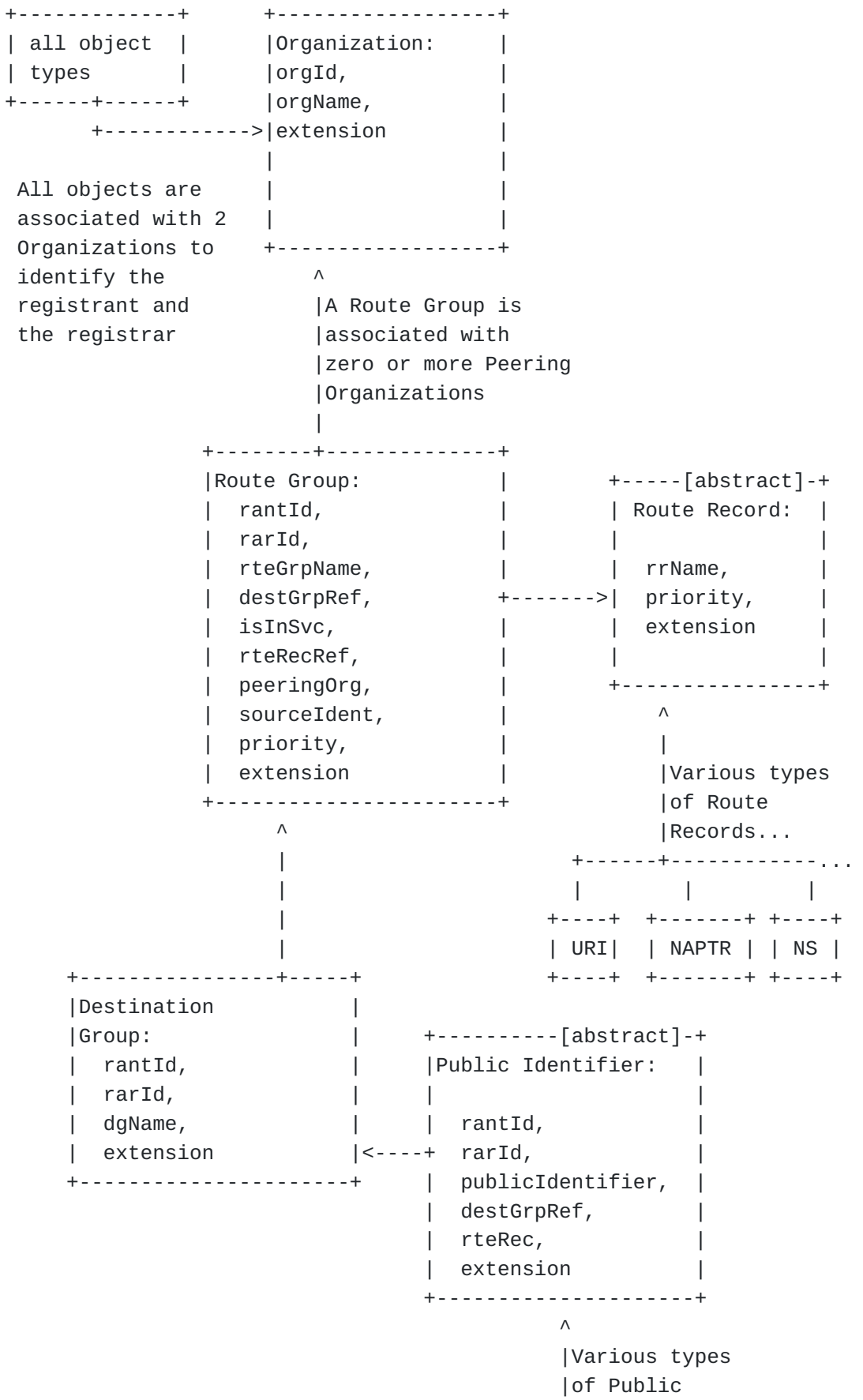
4. The operation layer defines the set of base SPPP actions that can be invoked for a given object data type using an SPPP message. Operations are encoded using XML encoded actions and objects.

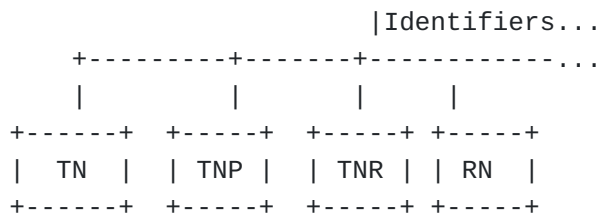
5. The data object layer defines the base set of SPPP data objects that can be included in update operations or returned in operation responses.

3.2. Protocol Data Model

TOC

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 Registrars may use the protocol for populating the Registry's data.
The logical structure presented below is consistent with the terminology and requirements defined in [I-D.ietf-drinks-usecases-requirements] (Channabasappa, S., “DRINKS Use cases and Protocol Requirements,” May 2010.).


all object types
Organization:
orgId,
orgName,
extension
All objects are associated with 2 Organizations to identify the registrant and the registrar
A Route Group is associated with zero or more Peering Organizations
Route Group:
rantId,
rarId,
rteGrpName,
destGrpRef,
isInSvc,
rteRecRef,
peeringOrg,
sourceIdent,
priority,
extension
[abstract]
Route Record:
rrName,
priority,
extension
Various types of Route Records...
URI
NAPTR
NS
Destination Group:
rantId,
rarId,
dgName,
extension
[abstract]
Public Identifier:
rantId,
rarId,
publicIdentifier,
destGrpRef,
rteRec,
extension
Various types of Public



SPPP Data Model

Figure 3

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

*Public Identifier:
A public identifier is a well known attribute that is used as the key to perform lookup functions. For the purposes of this document, a Public Identifier can be a telephone number, a range of telephone numbers, a PSTN Routing Number (RN), or perhaps another type of lookup key.

A Public Identifier is associated with a Destination Group to create a logical grouping of Public Identifiers that share a common set of Routes.

A TN 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 Route Records (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 TN Public Identifier.

*Telephone Number Range:
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 defined by tn=12125550000 and endTn=12125560000 means all the TNs from 12125550000 to 12125560000 inclusive are included.

*Destination Group:
A name collection of zero or more Public Identifiers that can be associated with one or more Route Groups for the purpose of facilitating the management of thier common routing information.

*Route Group:
A Route Group contains a set of references to Route Records, a set of Destination Group references, and a set of peering organization identifiers. This is used to establishe a three part

relationships between a set of Public Identifiers and their common routing information (SED), and the list of peering organizations whose query responses may include that routing information in their query responses. To support the use cases defined in [I-D.ietf-drinks-usecases-requirements], this document defines the following types of Route Records: NAPTRType, NSType, and URIType. The sourceIdent element within a Route Group, in concert with the set of peering organization identifiers enables fine grained source based routing. Further details about the Route Group and source based routing refer to the definitions and descriptions of the Route Group operations found later in this document.

*Route Record:
A Route Record contains the data that a resolution system returns in response to a successful query for a Public Identifier. Route Recoords are associated with a Route Group for SED that is not specific to a Public Identifier.
To support the use cases defined in [I-D.ietf-drinks-usecases-requirements] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.), 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.

*Organization:
An Organization is an entity that may fulfill any combination of three roles: Registrant, Registrar, and Peering Organization. All SPPP objects are associated with two organization identifiers to identify each object's registrant and the registrar. A Route Group object is also associated with a set of zero or more organization identifiers that identify the peering organizations whose query responses may include the routing information (SED) defined in the Route Records within that Route Group.

---

4. Transport Protocol Requirements

TOC

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] (Channabasappa, S., "DRINKS Use cases and Protocol Requirements," May 2010.). 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:

- 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.

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

- 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?

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

4.1. Connection Oriented

TOC

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 and Response Model

TOC

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

TOC

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

TOC

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 and Integrity

TOC

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

TOC

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 and Response Sizes

TOC

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

TOC

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

4.9. Request Acknowledgement

TOC

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

TOC

As of this writing of this revision, one transport protocol proposal has been provided in [I-D.ietf-drinks-sppp-over-soap] (Cartwright, K., “SPPP Over SOAP and HTTP,” June 2010.).

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

5. Base Protocol Data Structures

TOC

SPPP uses a common model and a common set of data structures for most of the supported operations and object types. This section describes these common data structures.

5.1. Request and Response Structures

TOC

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. There are two generalized types of operations that an SPPP client can submit to an SPPP server, updates and queries. The following two sub-sections describe the generalized data structures that are used for each of these two types of operations.

5.1.1. Update Request and Response Structures

TOC

An SPPP update request is wrapped within the <spppUpdateRequest> element while an SPPP update response is wrapped within an <spppUpdateResponse> element. The following two sub-sections describe these two elements.

5.1.1.1. Update Request

TOC

An SPPP update request object is contained within the generic <spppUpdateRequest> element.

```
<element name="spppUpdateRequest">
  <complexType>
    <sequence>
      <element name="clientTransId" type="spppb:TransIdType"
               minOccurs="0"/>
      <element name="minorVer" type="spppb:MinorVerType"
               minOccurs="0"/>
      <element name="rqst" type="spppb:BasicRqstType"
               maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

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

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

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

*clientTransId: Zero or one client generated transaction ID that, within the context of the SPPP client, identifies this request. This value can be used at the discretion of the SPPP client to track, log or correlate requests and their responses. This value is also echoed back to the client in the SPPP update response. An SPPP server will not check this value for uniqueness.

*minorVer: Zero or one minor version identifier, indicating 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 to identify the version of SPPP that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPP server for the given major version. The versions supported by a given SPPP server can be retrieved by the client using the SPPP server menu operation described later in the document.

*rqst: One or more BasicRqstType objects. These are the actions that the client is requesting the SPPP server perform. They are processed by the SPPP server in the order in which they are

included in the request. And with respect to handling error conditions, it is a matter of policy whether the objects are processed in a "stop and rollback" fashion or in a "stop and commit" fashion. In the "stop and rollback" scenario, the SPPP server would stop processing BasicRqstType object instances in the request at the first error and roll back any BasicRqstType object instances that had already been processed for that update request. In the "stop and commit" scenario the SPPP server would stop processing BasicRqstType object instances in the request at the first error but commit any BasicRqstType object instances that had already been processed for that update request.

All update request objects extend the base type BasicRqstType. This base type is defined as follows:

```
<complexType name="BasicRqstType" abstract="true">
  <sequence>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
```

The BasicRqstType object primarily acts as an abstract base type, and its only data element is described as follows:

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

5.1.1.2. Update Response

TOC

An SPPP update response object is contained within the generic <spppUpdateResponse> element.

```
<element name="spppUpdateResponse">
  <complexType>
    <sequence>
      <element name="overallResult" type="spppb:ResultCodeType"/>
      <element name="rqstObjResult" type="spppb:RqstObjResultCodeType"
          minOccurs="0" maxOccurs="unbounded"/>
      <element name="clientTransId" type="spppb:TransIdType"
          minOccurs="0"/>
      <element name="serverTransId" type="spppb:TransIdType"/>
    </sequence>
  </complexType>
</element>

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

<complexType name="RqstObjResultCodeType">
  <complexContent>
     <extension base="spppb:ResultCodeType">
        <sequence>
           <element name="rqstObj" type="spppb:BasicRqstType"/>
        </sequence>
     </extension>
  </complexContent>
</complexType>
```

An <spppUpdateResponse> contains the elements necessary for the SPPP client to precisely determine the overal result of the request, and if an error occurred, it provides information about the specific object, data element, or condition caused the error.
The data elements within the SPPP update response are described as follows:

*clientTransId: Zero or one client transaction ID. This value is simply an echo of the client transaction ID that SPPP client passed into the SPPP update request.

*serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value is guaranteed to be unique for a given SPPP server.

*overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See the Response Code section for further details.

*rqstObjResult: An optional response code, response message, and BasicRqstObject triplet. This element will be present only if an object level error condition occurs, and indicates exactly which error condition occurred and exactly which request object that was passed in caused the error condition. The contained BasicRqstObject is simply an echo of the request object instance that caused the error, while the response code and message indicate the error condition for this object. See the Response Code section for further details.

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

5.1.2. Query Request and Response Structures

TOC

An SPPP query request is wrapped within the <spppQueryRequest> element while an SPPP query response is wrapped within an <spppQueryResponse> element. The following two sub-sections describe these two element structures.

5.1.2.1. Query Request

TOC

An SPPP query request object is contained within the generic <spppQueryRequest> element.

```
<element name="spppQueryRequest">
  <complexType>
    <sequence>
      <element name="minorVer" type="spppb:MinorVerType"
          minOccurs="0"/>
      <element name="rqst" type="spppb:BasicQueryRqstType"/>
    </sequence>
  </complexType>
</element>
```

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

- *minorVer: Zero or one minor version identifier, indicating 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 to identify the version of SPPP that the client is using. If the element is not present, the server assumes that the client is using the latest minor version supported by the SPPP server for the given major version. The versions supported by a given SPPP server can be retrieved by the client using the SPPP server menu operation described later in the document.

- *rqst: One BasicQueryRqstType objects. This is the query that the client is requesting the SPPP server perform.

All query request objects extend the base type BasicQueryRqstType. This base type is defined as follows:

```
<complexType name="BasicQueryRqstType" abstract="true">
  <sequence>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
```

The BasicQueryRqstType object primarily acts as an abstract base type, and its only data element is described as follows:

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

5.1.2.2. Query Response

TOC

An SPPP query response object is contained within the generic <spppQueryResponse> element.

```
<element name="spppQueryResponse">
  <complexType>
    <sequence>
      <element name="overallResult" type="spppb:ResultCodeType"/>
      <element name="resultSet" type="spppb:BasicObjType"
         minOccurs="0" maxOccurs=" unbounded"/>
    </sequence>
  </complexType>
</element>
```

An <spppQueryResponse> contains the elements necessary for the SPPP client to precisely determine the overal result of the query, and if an error occurred, exactly what condition caused the error.
The data elements within the SPPP query response are described as follows:

*overallResult: Exactly one response code and message pair that explicitly identifies the result of the request. See the Response Code section for further details.

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

5.2. Response Codes and Messages

TOC

This section contains the 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 [RFC5321] (Klensin, J., “Simple Mail Transfer Protocol,” October 2008.):

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

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

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

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

Result Code	Text	Overall or Object Level
1000	Request Succeeded.	Overall Response Code
2001	Request syntax invalid.	Overall Response Code
2002	Request too large.	Overall Response Code
2003	Version not supported.	Overall Response Code
2103	Command invalid.	Overall Response Code
2301	System temporarily unavailable.	Overall Response Code
2302	Unexpected internal system or server error.	Overall Response Code
2104	Attribute value invalid. AttrName: [AttributeName] AttrVal: [AttributeValue]	Object Level Response Code
2105	Object does not exist. AttrName: [AttributeName] AttrVal: [AttributeValue]	Object Level Response Code
2106	Object status or ownership does not allow for operation. AttrName: [AttributeName] AttrVal: [AttributeValue]	Object Level Response Code

Table 1: Response Codes Numbering Scheme and Messages

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

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

- All parameters within a response message are mandatory and MUST be present.

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

- The value for "AttributeValue" MUST be the value of the data element to which the preceding "AttributeName" 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 by an update operation 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.

5.3. Basic Object Type and Organization Identifiers

TOC

This section introduces the basic object type that most first class objects derive from.
All first class objects extend the basic object type BasicObjType which contains the identifier of the registrant organization that owns this object, the identifier of the registrar organization that provisioned this object, the date and time that the object was created by the server, and the date and time that the object was last modified.

```
<complexType name="BasicObjType" abstract="true">
  <sequence>
    <element name="rantId" type="spppb:OrgIdType"/>
    <element name="rarId" type="spppb:OrgIdType"/>
    <element name="crtDate" type="dateTime" minOccurs="0"/>
    <element name="modDate" type="dateTime" minOccurs="0"/>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
```

The identifiers used for registrants (rantId), registrars (rarId) and peering organizations (peeringOrg) are instances of OrgIdType. The OrgIdType is defined as a string and all OrgIdType instances SHOULD follow the textual convention: "namespace:value" (for example "iana-en: 32473"). See the IANA Consideration section for more details.

---

6. Protocol Commands

TOC

This section provides a description of each supported protocol command.

6.1. Add Route Group Operation

TOC

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 identifiers to their 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 or Route Record , rather than millions of data updates to individual public identifier records that individually contain their peering data.
The AddRteGrpRqstType operation creates or overwrites a Route Group object. If a Route Group with the given name and registrant ID (which together comprise the unique key or a Route Group) does not exist, then the server MUST create the Route Group. If a Route Group with the given name and registrant ID does exist, then the server MUST replace the current properties of the Route Group with the properties passed into the AddRteGrpRqstType operation. The XSD declarations of the AddRteGrpRqstType operation request object are as follows:

```
<complexType name="AddRteGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrp" type="spppb:RteGrpType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The element passed into the spppUpdateRequest element for this operation is an instance of AddRteGrpRqstType, which extends BasicRqstType and contains one RteGrpType object. The RteGrpType object structure is defined as follows:

```
<complexType name="RteGrpType">
  <complexContent>
   <extension base="spppb:BasicObjType">
     <sequence>
       <element name="rteGrpName" type="spppb:ObjNameType"/>
       <element name="rteRecRef" type="spppb:RteRecRefType"
         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="priority" type="unsignedShort"/>
       <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
     </sequence>
   </extension>
  </complexContent>
</complexType>

<complexType name="RteRecRefType">
  <sequence>
    <element name="rteRec" type="spppb:ObjKeyType"/>
    <element name="priority" type="unsignedShort"/>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
```

The RteGrpType object is composed of the following elements:

*base: All first class objects extend BasicObjType which contains the ID of the registrant organization that owns this object, the ID of the registrar organization that provisioned this object, the date and time that the object was created by the server, and the date and time that the object was last modified. If the client passes in either the created date or the modification date, the server will ignore them. The server sets these two date/time values.

*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).

*rteRecRef: Set of zero or more objects of type RteRecRefType that house the unique keys of the Route Records that the the RteGrpType object refers to and their relative priority within the context of a given route group. The associated Route Records contain the routing information, sometimes called SED, associated with this Roue Group.

*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 identifiers are reachable using the routing information housed in this Route Group. An intended side affect of this is that a Route Group cannot provide routing information for a Destination Group belonging to another registrant.

*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 in resolution responses.

*priority: Zero or one priority value that can be used to provide a relative value weighting of one Route Group over another. The manner in which this value is used, perhaps in conjunction with other factors, is a matter of policy.

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

As described above, the Route Group contains a set of references to route record objects. A route record object is based on an abstract type: RteRecType. The concrete types that use RteRecType as an

extension base are NAPTRType, NSType, and URIType. The definitions of these types are included the Route Record section of this document. The RteGrpType object provides support for source-based routing via the peeringOrg data element and more granular source base routing via the source identity element. The source identity element 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 and whose peeringOrg list contains the querying organizations organization ID, 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 then compares these criteria against the source identity 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. The Source Identity data element is of type SourceIdentType, whose structure is defined as follows:

```
<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>
```

The SourceIdentType object is composed of the following data elements:

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

*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.

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

As with the responses to all update operations, the result of the AddRteGrpRqstType operation is contained in the generic spppUpdateResponse data structure described in an earlier sections of this document. For a detailed description of the spppUpdateResponse data structure refer to that section of the document.

6.2. Get Route Groups Operation

TOC

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 spppQueryRequest element for this operation is an instance 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:

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

As described in an earlier section of this document, the result of any spppQueryRequest operation is an spppQueryResponse element that contains the overall response code and the query result set, if any.

Refer to that section of the document for a detailed description of the spppQueryResponse element.

6.3. Add Destination Group Operation

TOC

As described in the introductory sections, a Destination Group represents a set of Public Identifiers with common routing information. The AddDestGrpRqstType operation creates or overwrites a Destination Group object. If a Destination Group with the given name and registrant ID (which together comprise the unique key for a Destination Group) does not exist, then the server MUST create the Destination Group. If a Destination Group with the given name and registrant ID does exist, then the server MUST replace the current properties of the Destination Group with the properties passed into the AddDestGrpsRqstType operation. The XSD declarations of the operation request object are as follows:

```
<complexType name="AddDestGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="destGrp" type="spppb:DestGrpType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The element passed into the spppUpdateRequest element for this operation is an element of type AddDestGrpRqsttype, which extends BasicRqstType and contains a DestGrpType object. The DestGrpType object structure is defined as follows:

```
<complexType name="DestGrpType">
  <complexContent>
   <extension base="spppb:BasicObjType">
    <sequence>
       <element name="dgName" type="spppb:ObjNameType"/>
    </sequence>
   </extension>
  </complexContent>
</complexType>
```

The DestGrpType object is composed of the following elements:

- *base: All first class objects extend BasicObjType which contains the ID of the registrant organization that owns this object, the ID of the registrar organization that provisioned this object, the date and time that the object was created by the server, and the date and time that the object was last modified. If the client passed in either the created date or the modification date, the server will ignore them. The server sets these two date/time values.

- *dgName: The character string that contains the name of the Destination Group. This uniquely identifies this object within the context of the registrant ID (a child element of the base element as described above).

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

As with the responses to all update operations, the result of the AddDestGrpRqstType operation is contained in the generic spppUpdateResponse data structure described in an earlier sections of this document. For a detailed description of the spppUpdateResponse data structure refer to that section of the document.

6.4. Get Destination Groups Operation

TOC

The getDestGrpsRqst operation allows a client to get the properties of Destination Group objects that a registrar organization is authorized to view. The server will attempt to find a Destination Group object that has the registrant ID and destination group name pair contained in each ObjKeyType object instance. If there are no matching Destination Groups found then an empty result set will be returned. If the set of ObjKeyType objects passed in is empty then the server will return the list of Destination Group objects that the querying registrar has the authority to view.
The element passed into the spppQueryRequest element for this operation is an instance of type GetDestGrpsRqstType, which extends BasicQueryRqstType 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:

```
<complexType name="GetDestGrpsRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

As described in an earlier section of this document, the result of any spppQueryRequest operation is an spppQueryResponse element that contains the overall response code and the query result set, if any. Refer to that section of the document for a detailed description of the spppQueryResponse element.

6.5. Add Route Group Offer Operation

TOC

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, also know as a peering organization). The registrant offers access to a Route Group by submitting a Route Group Offer. 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 AddRteGrpOffersRqstType operation creates or overwrites one or more Route Group Offer objects. If a Route Group Offer for the given Route Group object key and the 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:

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

The element passed into the spppUpdateRequest element for this operation is an instance of AddRteGrpOfferRqstType, which extends BasicRqstType and contains a RteGrpOfferType object. The XSD declaration of the RteGrpOfferType is as follows:

```
<complexType name="RteGrpOfferType">
 <complexContent>
   <extension base="spppb:BasicObjType">
     <sequence>
       <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>
   </extension>
  </complexContent>
</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: All first class objects extend BasicObjType which contains the ID of the registrant organization that owns this object, the ID of the registrar organization that provisioned this object, the date and time that the object was created by the server, and the date and time that the object was last modified. If the client passed in either the created date or the modification date, the will ignore them. The server sets these two date/time values.

*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.

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

As with the responses to all update operations, the result of the AddRteGrpOfferRqstType operation is contained in the generic spppUpdateResponse data structure described in an earlier sections of this document. For a detailed description of the spppUpdateResponse data structure refer to that section of the document.

6.6. Accept Route Group Offer Operation

TOC

Not until access to a Route Group has been offered and accepted will the data recipient's organization ID will it 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 AcceptRteGrpOffersRqstType operation is called by, or on behalf of, the data recipient to accept a Route Group Offer that is 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:

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

The element passed into the spppUpdateRequest element for this operation is an instance of AcceptRteGrpOffersRqstType, which extends BasicRqstType and contains a RteGrpOfferKeyType object.
As with the responses to all update operations, the result of the AcceptRteGrpOfferRqstType operation is contained in the generic spppUpdateResponse data structure described in an earlier sections of this document. For a detailed description of the spppUpdateResponse data structure refer to that section of the document.

6.7. Reject Route Group Offer Operation

TOC

The data recipient to which a Route Group has been offered has the option of rejecting a Route Group Offer. Furthermore, that offer may be rejected, regardless of whether or not it has been previously accepted. The RejectRteGrpOffersRqstType operation is used for these purposes and is called by, or on behalf of, the data recipient to accept a Route Group Offer that is pending in the "offered" status or is in 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 peeringOrg IDs 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:

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

The element passed into the spppUpdateRequest element for this operation is an instance of RejectRteGrpOffersRqstType, which extends BasicRqstType and contains a RteGrpOfferKeyType object.
As with the responses to all update operations, the result of the RejectRteGrpOfferRqstType operation is contained in the generic spppUpdateResponse data structure described in an earlier sections of this document. For a detailed description of the spppUpdateResponse data structure refer to that section of the document.

6.8. Get Route Group Offers Operation

TOC

The getRteGrpOffersRqst operation allows a client to get the properties of zero or more Route Group Offer objects that registrar is authorized to view. The server will attempt to find Route Group Offer objects that have 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 spppQueryRequest element for this operation is an instance of GetRteGrpOffersRqstType, which extends BasicQueryRqstType and contains the criteria that the returned Route Group Offer objects must match. Any limitation on the maximum number of objects that may be 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="GetRteGrpOffersRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <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:

*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.

*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.

*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.

*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.

*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.

As described in an earlier section of this document, the result of any spppQueryRequest operation is an spppQueryResponse element that contains the overall response code and the query result set, if any. Refer to that section of the document for a detailed description of the spppQueryResponse element.

6.9. Public Identifier Operations

TOC

Public Identifier is the search key used for locating the session establishment data (SED). In many cases, a Public Identifier is attributed to the end user who has a retail relationship with the service provider or registrant organization. In SPPP, SED can be provisioned by the registrant, or by the registrar on behalf of the registrant. Also, SPPP supports the notion of the carrier-of-record as defined in RFC 5067. Therefore, the entity adding the Public Identity in the Registry can optionally claim to be a carrier-of-record.
SPPP identifies three types of Public Identifiers: telephone number (TN), email address, and the routing number (RN). SPPP provides structures to manage a single TN, a contiguous range of TNs, and a TN prefix.
The XML schema type definition PubIDType is the generalization of the Public Identifier. PubIDType is an abstract type. In agreement with the data model, PubIDType member 'dgName' represents the name of the destination group that a given Public Identifier is associated to. The PubIDType object structure is defined as follows:

```
<complexType name="PubIdType" abstract="true">
  <complexContent>
   <extension base="spppb:BasicObjType">
     <sequence>
       <element name="dgName" type="spppb:ObjNameType" minOccurs="0"/>
     </sequence>
   </extension>
  </complexContent>
</complexType>
```

A registrant can add a Public Identifier with the help of a BasicRqstType called AddPubIdRqstType. To complete the add request, AddPubIdRqstType XML instance is added to <spppUpdateRequest> root element. If there is a conflict and a Public Identifier already exists

in the Registry, the old entry will be replaced with the newly provisioned entry. For the add or update operation, the destination group name is a mandatory parameter. Not including a valid destination group name in the update request will cause the Registry to return an appropriate error.
Telephone number is identified by TNType, an extension of PubIDType. TNType is composed of the following attributes:

- *tn: Telephone number to be added to the Registry.

- *rteRecRef: Optional reference to the route record that is directly associated with the TN Public Identifier. Following the SPPP data model, the route record could be a protocol agnostic URIType or another type.

- *corInfo: corInfo is an optional parameter of type CORInfoType that allows the registrant organization to set forth a claim to be the carrier-of-record [see RFC 5067]. This is done by setting the value of <corClaim> element of the CORInfoType object structure to "true". The other two parameters of the CORInfoType, <cor> and <corDateTime> are set by the Registry to describe the outcome of the carrier-of-record claim by the registrant. In general, inclusion of <corInfo> parameter is useful if the Registry has the authority information, such as, the number portability data, etc., in order to qualify whether the registrant claim can be satisfied. If the carrier-of-record claim disagrees with the authority data in the Registry, whether the TN add operation fails or not is a matter of policy and it is beyond the scope of this document. In the response message <spppUpdateResponse>, the SPPP Server must include the <cor> parameter of the <corInfo> element to let the registrant know the outcome of the claim.

TNType object definition is as follows:

```
<complexType name="TNType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="tn" type="string"/>
        <element name="rteRecRef" type="spppb:RteRecRefType"
           minOccurs="0" maxOccurs="unbounded"/>
        <element name="corInfo" type="spppb:CORInfoType"
           minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Routing number is identified by RNType. SSPs that possess the number portability data may be able to leverage the RN search key to discover the ingress routes for session establishment. Therefore, the registrant organization can add the RN and associate it with the appropriate destination group to share the route information.
RNType is composed of the following attributes:

*rn: Routing Number used as the search key

*corInfo: Optional <corInfo> element of type CORInfoType.

RNType object information is as follows:

```
<complexType name="RNType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="rn" type="string" default="true"/>
        <element name="corInfo" type="spppb:CORInfoType"
             minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

A contiguous range of TNs is added with the help of TNRType structure. The object definition requires a starting TN and the ending TN that describes the TN range. In addition, TNRType includes an optional "prefix" attribute to indicate that the given TN range qualifies for the Open Number Plan (ONP). In order to correctly expand the number range that qualifies for Open Number Plan, the Registry must have the

required data related to the national significant number length for the TNs in the TN range object. Further, <spppUpdateRequest> is transactional in nature, therefore, if the Registry encounters an error in adding even a single TN of a TNRType object, the whole request will be deemed a failure. In other words, the partial success case is not supported.
TNRType is composed of the following attributes:

*startTn: Starting TN in the TN range

*endTn: The last TN in the TN range

*corInfo: Optional <corInfo> element of type CORInfoType

*prefix: Optional attribute, when set to "true", indicates that the Open Number Plan applies to a given TN Range

TNPType object structure is as follows:

```
<complexType name="TNRType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="startTn" type="string"/>
        <element name="endTn" type="string"/>
        <element name="corInfo" type="spppb:CORInfoType"
            minOccurs="0"/>
      </sequence>
      <attribute name="prefix" type="boolean" default="false">
         </attribute>
    </extension>
  </complexContent>
</complexType>
```

In some cases, it is useful to describe a set of TNs with the help of the first few digits of the telephone numbers identified as telephone number prefix. In many instances, the numbering authority assigns TN prefix, also referred to as a TN block, to a well-known telephony service provider. In SPPP, the TNPType structure describes a TN prefix and it consists of the following attributes:

*tnPrefix: The telephone number prefix

*corInfo: Optional <corInfo> element of type CORInfoType.

TNPType object structure is as follows:

```
<complexType name="TNPType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="tnPrefix" type="string"/>
        <element name="corInfo" type="spppb:CORInfoType"
          minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The object structure of AddPubIdRqstType used to add Public Identifiers is as follows

```
<complexType name="AddPubIdRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The client can set the GetPubIdsRqstType in the <spppQueryRequest> structure to obtain information about one or more <pi> objects that were successfully provisioned earlier and that the calling entity is privileged to see. If the GetPubIdsRqstType object does not include <pi> data, then all authorized Public Identity data will be returned by the Registry in the response. If no matching Public Identifiers are found, then an empty result set will be returned.
GetPubIdsRqstType object structure is as follows:

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

As described in an earlier section of this document, the result of any spppQueryRequest operation is a spppQueryResponse that contains the response code and the query result set, if any.

6.10. Egress Route Operations

TOC

The egress route add operation allows a call originating SSP to define a preferred egress route in an attempt to reach the ingress SBE of the target SSP. The need arises when there is a choice of egress SBE and an SSP wants to exercise greater control in deciding how to route the outbound session establishment request.

As a first step, it is assumed that the target SSP has offered to share the route group that consists of the ingress route information to the SBE(s) and the originating SSP has accepted the offer. Next, the originating SSP can add the egress route in the Registry, with appropriate regular expression, to rewrite ingress route information from the target SSP and include the egress SBE information. In high-availability configurations, the originating SSP will likely add a secondary egress route object re-writing the same ingress route from the target SSP with a secondary choice of egress SBE as a backup. In this case, the backup egress route definition will carry the higher integer value for the "pref" parameter to indicate a lower priority.

An egress route is identified by type EgrRteType and its object structure is shown below:

```
<complexType name="EgrRteType">
  <complexContent>
   <extension base="spppb:BasicObjType">
     <sequence>
       <element name="egrRteName" type="spppb:ObjNameType"/>
       <element name="pref" type="unsignedShort"/>
       <element name="regxRewriteRule" type="spppb:RegexParamType"/>
       <element name="ingrRteRec" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
       <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
     </sequence>
   </extension>
  </complexContent>
</complexType>
```

The EgrRteType object is composed of the following elements:

*base: All first class objects extend BasicObjType which contains the ID of the registrant organization that owns this object, the ID of the registrar organization that provisioned this object, the date and time that the object was created by the server, and the date and time that the object was last modified. If the client passes in either the created date or the modification date, the server will ignore them. The server sets these two date/time values.

*egrRteName: The name of the egress route.

*pref: The preference of this egress route relative to other egress routes that may get selected when responding to a resolution request.

*regxRewriteRule: The regular expression re-write rule that should be applied to the regular expression of the ingress NAPTR(s) that belong to the ingress route.

*ingrRteRec: The ingress route records that the egress route should be used for.

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

The AddEgrRteRqstType request is used to create or overwrite an egress route.

```
<complexType name="AddEgrRteRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="egrRte" type="spppb:EgrRteType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

An instance of AddEgrRtesRqstType is added in the spppUpdateRequest element in order to send a valid request to the server. Any limitation on the maximum number of AddEgrRteRqstType instances is a matter of policy and is not limited by the specification.
The response from the server is returned in addEgrRteRspns element, which is defined as the element of type BasicRspnsType.
The GetEgrRtesRqstType is used by an authorized entity to fetch the well-known egress route data.

```
<complexType name="GetEgrRtesRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.11. Add Route Record Operation

TOC

As described in the introductory sections, a Route Group represents a combined grouping of Route Records that define route information. However, Route Records need not be created to just server a single Route Group. Route Records can be created and managed to serve multiple Route Groups. As a result, a change to the properties of a network node, for example, that is used for multiple routes, would necessitate just a single update operation to change the properties of that node. The change would then be reflected in all the Route Groups whose route record set contains a reference to that node.

The AddRteRecRqstType operation creates or overwrites a Route Record object. If a Route Record with the given name and registrant ID (which together comprise the unique key or a Route Record) does not exist, then the server MUST create the Route Record. If a Route Record with the given name and registrant ID does exist, then the server MUST replace the current properties of the Route Record with the properties passed into the AddRteRecRqstType operation. The XSD declarations of the AddRteRecRqstType operation request object are as follows:

```
<complexType name="AddRteRecRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteRec" type="spppb:RteRecType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The element passed into the spppUpdateRequest element for this operation is an instance of AddRteRecRqstType, which extends BasicRqstType and contains one RteRecType object. The RteRecType object structure is defined as follows:

```
<complexType name="RteRecType" abstract="true">
  <complexContent>
    <extension base="spppb:BasicObjType">
       <sequence>
         <element name="rrName" type="spppb:ObjNameType"/>
         <element name="priority" type="unsignedShort" minOccurs="0"/>
       </sequence>
    </extension>
  </complexContent>
</complexType>
```

The RteRecType object is composed of the following elements:

*base: All first class objects extend BasicObjType which contains the ID of the registrant organization that owns this object, the ID of the registrar organization that provisioned this object, the date and time that the object was created by the server, and the date and time that the object was last modified. If the client passes in either the created date or the modification date, the server will ignore them. The server sets these two date/time values.

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

*priority: Zero or one priority value that can be used to provide a relative value weighting of one Route Record over another. The manner in which this value is used, perhaps in conjunction with other factors, is a matter of policy.

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

As described above, route records are based on an abstract type: RteRecType. 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 for a 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.
The data provisioned in a Registry can be leveraged for many purposes and queried using various protocols including SIP, ENUM and others. It is for this reason that a route record type offers a choice of URI and DNS resource record types. URIType fulfills the need for both SIP and ENUM protocols. When a given URIType is associated to a destination group, the user part of the replacement string <uri> that may require the Public Identifier cannot be preset. As a SIP Redirect, the resolution server will apply <ere> pattern on the input Public Identifier in the query and process the replacement string by substituting any back reference(s) in the <uri> to arrive at the final URI that is returned in the SIP Contact header. For an ENUM query, the resolution server will simply return the value of the <ere> and <uri> members of the URIType in the NAPTR REGEX parameter.

```
<complexType name="RteRecType" abstract="true">
  <sequence>
    <element name="rrName" type="spppb:ObjNameType"/>
    <element name="priority" type="unsignedShort"/>
  </sequence>
</complexType>
<complexType name="NAPTRType">
  <complexContent>
    <extension base="spppb:RteRecType">
      <sequence>
        <element name="order" 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="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>
```

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.

*svcs: ENUM service(s) that are served by the SBE. This field's value must be of the form specified in [RFC3761] (Faltstrom, P. and M. Mealling, “The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM),” April 2004.) (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.

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

The URIType object is composed of the following elements:

*ere: The POSIX Extended Regular Expression (ere) as defined in [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," January 2005.).

*uri: the URI as defined in [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," January 2005.). In some cases, this will serve as the replacement string and it will be left to the resolution server to arrive at the final usable URI.

As with the responses to all update operations, the result of the AddRteRecRqstType operation is contained in the generic spppUpdateResponse data structure described in an earlier sections of this document. For a detailed description of the spppUpdateResponse data structure refer to that section of the document.

6.12. Get Route Records Operation

TOC

The getRteRecsRqst operation allows a client to get the properties of Route Record objects that a registrar organization is authorized to view. The server will attempt to find a Route Record object that has the registrant ID and route record 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 Record objects that the querying client has the authority to view. If there are no matching Route Record found then an empty result set will be returned.
The element passed into the spppQueryRequest element for this operation is an instance of type GetRteRecsRqstType, 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:

```
<complexType name="GetRteRecsRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

As described in an earlier section of this document, the result of any spppQueryRequest operation is an spppQueryResponse element that contains the overall response code and the query result set, if any. Refer to that section of the document for a detailed description of the spppQueryResponse element.

6.13. Delete Operation

TOC

In order to remove an object from the Registry, an authorized entity can send the <spppUpdateRequest> to the Registry with a corresponding delete BasicRqstType object. 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 <spppUpdateRespnonse> message.
As an example, DelPubIdRqstType aids in identifying the Public Identifier that is used to delete a Public Identifier from the Registry. DelPubIdsRqstType object definition is shown below:

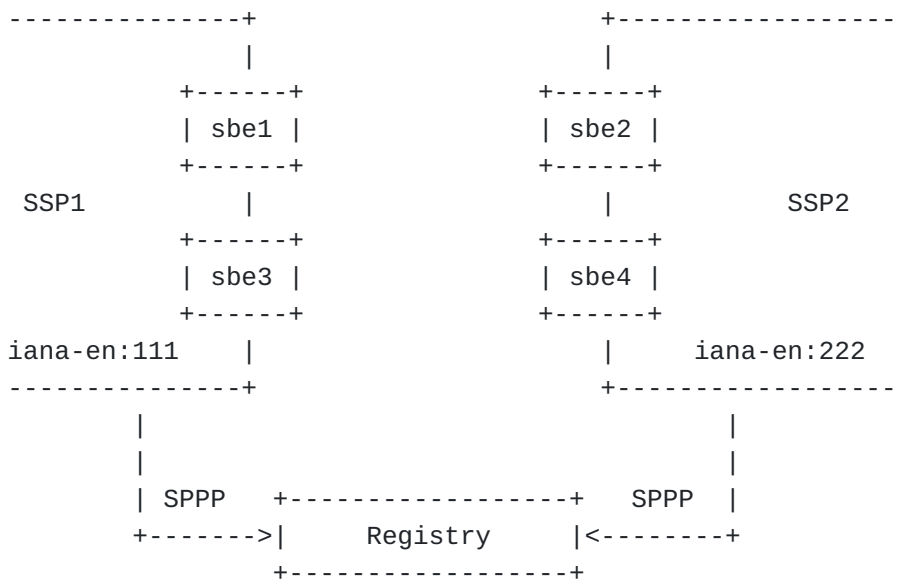
```
<complexType name="DelPubIdRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Similarly, each 'Add' operation in the SP protocol has a corresponding 'Del' operation used to delete the respective object type from the Registry.

7. SPPP Examples

TOC

This section shows XML message exchange between two SIP Service Providers (SSP) and a Registry. For the sake of simplicity, the transport wrapper for the SPPP protocol is left out. The SPPP protocol messages in this section are valid XML instances that conform to the SPPP schema version within this document.
In this sample use case scenario, SSP1 and SSP2 provision resource data in the registry and use SPPP constructs to selectively share the route groups. In the figure below, SSP2 has two ingress SBE instances that are associated with the public identities that SSP2 has the retail relationship with. Also, the two SBE instances for SSP1 are used to show how to use SPPP protocol to associate route preferences for the destination ingress routes and exercise greater control on outbound traffic to the peer's ingress SBEs.



7.1. Add Destination Group

TOC

SSP2 adds a destination group to the Registry for use later. The SSP2 SPPP client sets a unique transaction identifier 'tx_7777' for tracking purposes. The name of the destination group is set to DEST_GRP_SSP2_1

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
 xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <clientTransId>txid-5555</clientTransId>
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddDestGrpRqstType">
    <destGrp>
      <ns1:rantId>iana-en:222</ns1:rantId>
      <ns1:rarId>iana-en:222</ns1:rarId>
      <dgName>DEST_GRP_SSP2_1</dgName>
    </destGrp>
  </rqst>
</spppUpdateRequest>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
 xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <clientTransId>tx_7777</clientTransId>
  <serverTransId>tx_id_12346</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
</spppUpdateResponse>
```

7.2. Add Route Records

TOC

SSP2 adds an ingress routes in the Registry.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddRteRecRqstType">
    <rteRec xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:NAPTRType">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <ns1:rrName>RTE_SSP2_SBE2</ns1:rrName>
      <order>10</order>
      <flags>u</flags>
      <svcs>E2U+sip</svcs>
      <regx>
        <ere>^(.*)$</ere>
        <repl>sip:\1@sbe2.ssp2.example.com</repl>
      </regx>
    </rteRec>
  </rqst>
</spppUpdateRequest>
```

The Registry returns a success response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_11145</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.3. Add Route Records -- URIType

TOC

SSP2 adds another ingress routes in the Registry and makes use of URIType

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddRteRecRqstType">
    <rteRec xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:URIType">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <ns1:rrName>RTE_SSP2_SBE4</ns1:rrName>
      <ns1:ere>^(.*)$</ns1:ere>
      <ns1:uri>sip:\1;npdi@sbe4.ssp2.example.com</ns1:uri>
    </rteRec>
  </rqst>
</spppUpdateRequest>
```

The Registry returns a success response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_11145</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.4. Add Route Group

TOC

SSP2 creates the grouping of the ingress routes and choses higher precedence for RTE_SSP2_SBE2 by setting a lower number for the "priority" attribute, a protocol agnostic precedence indicator.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddRteGrpRqstType">
    <rteGrp>
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <rteGrpName>RTE_GRP_SSP2_1</rteGrpName>
      <ns1:rteRecRef>
        <ns1:rteRec>
          <ns1:rantId>iana-en:222</ns1:rantId>
          <ns1:name>RTE_SSP2_SBE2</ns1:name>
        </ns1:rteRec>
        <ns1:priority>100</ns1:priority>
      </ns1:rteRecRef>
      <dgName>DEST_GRP_SSP2_1</dgName>
      <isInSvc>true</isInSvc>
      <ns1:priority>10</ns1:priority>
    </rteGrp>
  </rqst>
</spppUpdateRequest>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12345</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

TOC

7.5. Add Public Identity -- Successful COR claim

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <clientTransId>txid-5577</clientTransId>
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddPubIdRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:TNType">
      <ns1:rantId>iana-en:222</ns1:rantId>
      <ns1:rarId>iana-en:222</ns1:rarId>
      <ns1:crtDate>2010-05-30T09:30:10Z</ns1:crtDate>
      <ns1:dgName>DEST_GRP_SSP2_1</ns1:dgName>
      <tn>+12025556666</tn>
      <ns1:corInfo>
        <ns1:corClaim>true</ns1:corClaim>
      </ns1:corInfo>
    </pi>
  </rqst>
</spppUpdateRequest>
```

Assuming that the Registry has access to TN authority data and it performs the required checks to verify that SSP2 is in fact the service provider of record for the given TN, the request was processed successfully. <cor> element confirms SSP2 claim to be the carrier of record has been accepted and the processing time is reflected by <corDateTime> data element.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <clientTransId>txid-5577</clientTransId>
  <serverTransId>tx_id_12345</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
  <rqstObjResult>
    <code>1000</code>
    <msg>success</msg>
    <rqstObj xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:AddPubIdRqstType">
        <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
          xsi:type="ns1:TNType">
          <ns1:rantId>iana-en:222</ns1:rantId>
          <ns1:rarId>iana-en:222</ns1:rarId>
          <ns1:crtDate>2010-05-30T09:30:10Z</ns1:crtDate>
          <ns1:dgName>DEST_GRP_SSP2_1</ns1:dgName>
          <tn>+12025556666</tn>
          <ns1:corInfo>
            <ns1:corClaim>true</ns1:corClaim>
            <ns1:cor>true</ns1:cor>
            <ns1:corDateTime>2006-05-04T18:13:51.0Z
                                    </ns1:corDateTime>
          </ns1:corInfo>
        </pi>
    </rqstObj>
  </rqstObjResult>
</spppUpdateResponse>
```

7.6. Add LRN

TOC

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddPubIdRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:RNType">
      <rantId>rantId0</rantId>
      <rarId>rarId0</rarId>
      <ns1:dgName>DEST_GRP_SSP2_1</ns1:dgName>
      <rn>2025550000</rn>
    </pi>
  </rqst>
</spppUpdateRequest>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12345</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.7. Add TN Range

TOC

Next, SSP2 activates a block of ten thousand TNs and associate it to a destination group. Since the 'prefix' public identity attribute is not set to 'true', this means that the TNs belong to a closed number plan.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddPubIdRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:TNRType">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <ns1:dgName>DEST_GRP_SSP2_1</ns1:dgName>
      <startTn>+12026660000</startTn>
      <endTn>+12026669999</endTn>
    </pi>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12244498</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.8. Add TN Range with Open Number Plan support

TOC

In this case, open number plan refers to TN length variance. Inclusion of "prefix" attribute of TNRType with its value set to true indicates that the start TN range identified by the <tn> element is not necessarily a subscriber number and the Registry will have to consult the number plan data for the respective country to know how to expand the number range. <endTn> attribute marks the end of the TN range.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddPubIdRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:TNRType" prefix="true">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <ns1:dgName>DEST_GRP_SSP2_1</ns1:dgName>
      <startTn>+4312315566</startTn>
      <endTn>+4312315567</endTn>
    </pi>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12255598</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.9. Add TN Prefix

TOC

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddPubIdRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:TNPType">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <ns1:dgName>DEST_GRP_SSP2_1</ns1:dgName>
      <tnPrefix>+1202777</tnPrefix>
    </pi>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12387698</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.10. Enable Peering -- Route Group Offer

TOC

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddRteGrpOfferRqstType">
    <rteGrpOffer>
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <rteGrpOfferKey>
        <rteGrpKey>
          <rantId>iana-en:222</rantId>
          <name>RTE_GRP_SSP2_1</name>
        </rteGrpKey>
        <offeredTo>iana-en:111</offeredTo>
      </rteGrpOfferKey>
      <status>offered</status>
      <offerDateTime>2006-05-04T18:13:51.0Z</offerDateTime>
    </rteGrpOffer>
  </rqst>
</spppUpdateRequest>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12277798</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

TOC

7.11. Enable Peering -- Route Group Offer Accept

SSP1 responds to the offer from SSP2 and agrees to have visibility to SSP2 ingress routes.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AcceptRteGrpOfferRqstType">
    <rteGrpOfferKey>
      <rteGrpKey>
        <rantId>iana-en:222</rantId>
        <name>RTE_GRP_SSP2_1</name>
      </rteGrpKey>
      <offeredTo>iana-en:111</offeredTo>
    </rteGrpOfferKey>
  </rqst>
</spppUpdateRequest>
```

Registry confirms that the request has been processed successfully. From this point forward, if SSP1 looks up a public identity through the query resolution server, where the public identity is part of the destination group by way of "RTE_GRP_SSP2_1" route association, SSP2 ingress SBE information will be shared with SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <serverTransId>tx_id_12333798</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
</spppUpdateResponse>
```

TOC

7.12. Add Egress Route

SSP1 wants to prioritize all outbound traffic to routes associated with "RTE_GRP_SSP2_1" route group through "sbe1.ssp1.example.com".

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <clientTransId>tx_9000</clientTransId>
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:AddEgrRteRqstType">
    <egrRte>
      <rantId>iana-en:111</rantId>
      <rarId/>
      <egrRteName>EGR_RTE_01</egrRteName>
      <pref>50</pref>
      <regxRewriteRule>
        <ere>^(.*@)(.*)$</ere>
        <repl>\1\2?route=sbe1.ssp1.example.com</repl>
      </regxRewriteRule>
      <ns1:ingrRteRec>
        <ns1:rantId>iana-en:222</ns1:rantId>
        <ns1:name>SSP2_RTE_REC_3</ns1:name>
      </ns1:ingrRteRec>
    </egrRte>
  </rqst>
</spppUpdateRequest>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd"
  xmlns="urn:ietf:params:xml:ns:sppp:base:1">
  <clientTransId>tx_9000</clientTransId>
  <serverTransId>tx_id_12388898</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Request successful</msg>
  </overallResult>
</spppUpdateResponse>
```

7.13. Get Destination Group

TOC

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:GetDestGrpsRqstType">
    <objKey>
      <rantId>iana-en:222</rantId>
      <name>DEST_GRP_SSP2_1</name>
    </objKey>
  </rqst>
</spppQueryRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
  <resultSet xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:DestGrpType">
    <rantId>iana-en:222</rantId>
    <rarId>iana-en:222</rarId>
    <dgName>DEST_GRP_SSP2_1</dgName>
  </resultSet>
</spppQueryResponse>
```

7.14. Get Public Identity

TOC

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:GetPubIdsRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:TNType">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <tn>+12025556666</tn>
    </pi>
  </rqst>
</spppQueryRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
  <resultSet xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:TNType">
    <rantId>iana-en:222</rantId>
    <rarId>iana-en:222</rarId>
    <ns1:dgName>DEST_GRP_1</ns1:dgName>
    <tn>+12025556666</tn>
    <ns1:corInfo>
      <ns1:corClaim>true</ns1:corClaim>
      <ns1:cor>true</ns1:cor>
      <ns1:corDateTime>2010-05-30T09:30:10Z</ns1:corDateTime>
    </ns1:corInfo>
  </resultSet>
</spppQueryResponse>
```

TOC

7.15. Get Route Group Request

SSP2 obtains the last provisioned record for the route group RTE_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:GetRteGrpsRqstType">
    <objKey>
      <rantId>iana-en:222</rantId>
      <name>RTE_GRP_SSP2_1</name>
    </objKey>
  </rqst>
</spppQueryRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
  <resultSet xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:RteGrpType">
    <rantId>iana-en:222</rantId>
    <rarId>iana-en:222</rarId>
    <rteGrpName>RTE_GRP_SSP2_1</rteGrpName>
    <ns1:rteRecRef>
      <ns1:rteRec>
        <ns1:rantId>iana-en:222</ns1:rantId>
        <ns1:name>RTE_SSP2_SBE2</ns1:name>
      </ns1:rteRec>
      <ns1:priority>100</ns1:priority>
    </ns1:rteRecRef>
    <ns1:rteRecRef>
      <ns1:rteRec>
        <ns1:rantId>iana-en:222</ns1:rantId>
        <ns1:name>RTE_SSP2_SBE4</ns1:name>
      </ns1:rteRec>
      <ns1:priority>101</ns1:priority>
    </ns1:rteRecRef>
    <dgName>DEST_GRP_SSP2_1</dgName>
    <isInSvc>true</isInSvc>
    <ns1:priority>10</ns1:priority>
  </resultSet>
</spppQueryResponse>
```

7.16. Get Route Group Offers Request

TOC

SSP2 choses to fetch the last provisioned route group sharing offer to the <peeringOrg> SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:GetRteGrpOffersRqstType">
    <ns1:offeredToPeers>true</ns1:offeredToPeers>
    <ns1:peeringOrg>ssp1</ns1:peeringOrg>
  </rqst>
</spppQueryRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
  <resultSet xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:RteGrpOfferType">
    <rantId>iana-en:222</rantId>
    <rarId>iana-en:222</rarId>
    <rteGrpOfferKey>
      <rteGrpKey>
        <rantId>iana-en:222</rantId>
        <name>RTE_GRP_SSP2_1</name>
      </rteGrpKey>
      <offeredTo>iana-en:111</offeredTo>
    </rteGrpOfferKey>
    <status>offered</status>
    <offerDateTime>2006-05-04T18:13:51.0Z</offerDateTime>
  </resultSet>
</spppQueryResponse>
```

TOC

7.17. Get Egree Route

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:GetEgrRtesRqstType">
    <ns1:objKey>
      <rantId>iana-en:111</rantId>
      <name>EGR_RTE_01</name>
    </ns1:objKey>
  </rqst>
</spppQueryRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppQueryResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
  <resultSet xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:EgrRteType">
    <rantId>iana-en:111</rantId>
    <rarId>iana-en:111</rarId>
    <egrRteName>EGR_RTE_01</egrRteName>
    <pref>50</pref>
    <svcs>E2U+sip</svcs>
    <regxRewriteRule>
      <ere>^(.*)$</ere>
      <repl>sip:\1@sbe1.ssp1.example.com</repl>
    </regxRewriteRule>
    <ingressRte>
      <rantId>iana-en:222</rantId>
      <name>RTE_GRP_SSP2_1</name>
    </ingressRte>
  </resultSet>
</spppQueryResponse>
```

7.18. Delete Destination Group

TOC

SSP2 initiates a request to delete the destination group DEST_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:DelDestGrpRqstType">
    <objKey>
      <rantId>iana-en:222</rantId>
      <name>DEST_GRP_SSP2_1</name>
    </objKey>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <serverTransId>txid-982543123</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Success</msg>
  </overallResult>
</spppUpdateResponse>
```

7.19. Delete Public Identity

TOC

SSP2 choses to de-activate the TN and remove it from the Registry.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:DelPubIdRqstType">
    <pi xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
      xsi:type="ns1:TNType">
      <rantId>iana-en:222</rantId>
      <rarId>iana-en:222</rarId>
      <tn>+12025556666</tn>
    </pi>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <serverTransId>txid-98298273123</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
</spppUpdateResponse>
```

7.20. Delete Route Group Request

TOC

SSP2 removes the route group called RTE_GRP_SSP2_1.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:DelRteGrpRqstType">
    <objKey>
      <rantId>iana-en:222</rantId>
      <name>RTE_GRP_SSP2_1</name>
    </objKey>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <serverTransId>txid-982543123</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>msg</msg>
  </overallResult>
</spppUpdateResponse>
```

7.21. Delete Route Group Offers Request

TOC

SSP2 no longer wants to share route group RTE_GRP_SSP2_1 with SSP1.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:DelRteGrpOfferRqstType">
    <rteGrpOfferKey>
      <rteGrpKey>
        <rantId>iana-en:222</rantId>
        <name>RTE_GRP_SSP2_1</name>
      </rteGrpKey>
      <offeredTo>iana-en:111</offeredTo>
    </rteGrpOfferKey>
  </rqst>
</spppUpdateRequest>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <serverTransId>txid-982543123</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Success</msg>
  </overallResult>
</spppUpdateResponse>
```

7.22. Delete Egress Route

TOC

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

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateRequest xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <rqst xmlns:ns1="urn:ietf:params:xml:ns:sppp:base:1"
    xsi:type="ns1:DelEgrRteRqstType">
    <objKey>
      <rantId>iana-en:111</rantId>
      <name>EGR_RTE_01</name>
    </objKey>
  </rqst>
</spppUpdateRequest>
```

Registry completes the request successfully and returns a favorable response.

```
<?xml version="1.0" encoding="UTF-8"?>
<spppUpdateResponse xmlns="urn:ietf:params:xml:ns:sppp:base:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:sppp:base:1 sppp.xsd">
  <serverTransId>txid-982543123</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>Success</msg>
  </overallResult>
</spppUpdateResponse>
```

---

8. XML Considerations

TOC

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.

8.1. Namespaces

TOC

All SPPP protocol elements are defined in the namespaces in te IANA Considerations section and in the Formal Protocol Specification section of this document.

8.2. Versioning and Character Encoding

TOC

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] (Yergeau, F., “UTF-8, a transformation format of ISO 10646,” November 2003.)) and UTF-16 (defined in [RFC2781] (Hoffman, P. and F. Yergeau, “UTF-16, an encoding of ISO 10646,” February 2000.)); per [RFC2277] (Alvestrand, H., “IETF Policy on Character Sets and Languages,” January 1998.) 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"?>

9. Security Considerations

TOC

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.

TOC

10. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [RFC3688] (Mealling, M., “The IETF XML Registry,” January 2004.).
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 (Formal Specification)).
IANA is requested to create a new SPPP registry for Organization Identifiers that will indicate valid strings to be used for well-known enterprise namespaces.
This document makes the following assignments for the OrgIdType namespaces:

Namespace	OrgIdType namespace string
IANA Enterprise Numbers	iana-en

---

11. Formal Specification

TOC

This section provides the draft XML Schema Definition for the SPPP protocol.

```
<?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">
    <complexContent>
     <extension base="spppb:BasicObjType">
       <sequence>
         <element name="rteGrpName" type="spppb:ObjNameType"/>
         <element name="rteRecRef" type="spppb:RteRecRefType"
           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="priority" type="unsignedShort"/>
         <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
       </sequence>
     </extension>
    </complexContent>
  </complexType>
  <complexType name="DestGrpType">
    <complexContent>
     <extension base="spppb:BasicObjType">
      <sequence>
         <element name="dgName" type="spppb:ObjNameType"/>
      </sequence>
     </extension>
    </complexContent>
  </complexType>
  <complexType name="PubIdType" abstract="true">
    <complexContent>
     <extension base="spppb:BasicObjType">
       <sequence>
         <element name="dgName" type="spppb:ObjNameType" minOccurs="0"/>
       </sequence>
     </extension>
    </complexContent>
```

```
</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="rteRecRef" type="spppb:RteRecRefType"
           minOccurs="0" maxOccurs="unbounded"/>
        <element name="corInfo" type="spppb:CORInfoType"
           minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="TNRType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="startTn" type="string"/>
        <element name="endTn" type="string"/>
        <element name="corInfo" type="spppb:CORInfoType"
            minOccurs="0"/>
      </sequence>
      <attribute name="prefix" type="boolean" default="false">
         </attribute>
    </extension>
  </complexContent>
</complexType>
<complexType name="TNPType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="tnPrefix" type="string"/>
        <element name="corInfo" 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"/>
        <element name="corInfo" type="spppb:CORInfoType"
           minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="RteRecType" abstract="true">
  <complexContent>
    <extension base="spppb:BasicObjType">
       <sequence>
         <element name="rrName" type="spppb:ObjNameType"/>
       </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="NAPTRType">
  <complexContent>
    <extension base="spppb:RteRecType">
      <sequence>
        <element name="order" 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">
 <complexContent>
   <extension base="spppb:BasicObjType">
     <sequence>
       <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>
   </extension>
  </complexContent>
</complexType>
<complexType name="EgrRteType">
  <complexContent>
   <extension base="spppb:BasicObjType">
     <sequence>
       <element name="egrRteName" type="spppb:ObjNameType"/>
       <element name="pref" type="unsignedShort"/>
       <element name="regxRewriteRule" type="spppb:RegexParamType"/>
       <element name="ingrRteRec" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
       <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
     </sequence>
   </extension>
  </complexContent>
</complexType>
<annotation>
  <documentation> ------------------ Abstract Object and Element
    Type Definitions -------------- </documentation>
</annotation>
<complexType name="BasicObjType" abstract="true">
  <sequence>
    <element name="rantId" type="spppb:OrgIdType"/>
    <element name="rarId" type="spppb:OrgIdType"/>
    <element name="crtDate" type="dateTime" minOccurs="0"/>
    <element name="modDate" type="dateTime" minOccurs="0"/>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
```

```
  </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="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="RteRecRefType">
  <sequence>
    <element name="rteRec" type="spppb:ObjKeyType"/>
    <element name="priority" type="unsignedShort"/>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
<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="cor" type="boolean" default="false"
       minOccurs="0"/>
    <element name="corDateTime" type="dateTime" minOccurs="0"/>
  </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>
<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>
<complexType name="ExtAnyType">
```

```
  <sequence>
    <any namespace="##other" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<annotation>
  <documentation> -------------- Operation Request and Response
    Object Type Definitions ------------ </documentation>
</annotation>
<complexType name="ResultCodeType">
  <sequence>
     <element name="code" type="int"/>
     <element name="msg" type="string"/>
  </sequence>
</complexType>
<complexType name="RqstObjResultCodeType">
  <complexContent>
     <extension base="spppb:ResultCodeType">
        <sequence>
           <element name="rqstObj" type="spppb:BasicRqstType"/>
        </sequence>
     </extension>
  </complexContent>
</complexType>

<complexType name="BasicRqstType" abstract="true">
  <sequence>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
<complexType name="BasicQueryRqstType" abstract="true">
  <sequence>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>

<complexType name="AddRteGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrp" type="spppb:RteGrpType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelRteGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"/>
```

```
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetRteGrpsRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AddRteRecRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteRec" type="spppb:RteRecType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelRteRecRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetRteRecsRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="AddDestGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="destGrp" type="spppb:DestGrpType"/>
      </sequence>
```

```
    </extension>
  </complexContent>
</complexType>
<complexType name="DelDestGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetDestGrpsRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AddPubIdRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelPubIdRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetPubIdsRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType" minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
```

```
</complexType>
<complexType name="AddRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrpOffer" type="spppb:RteGrpOfferType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrpOfferKey"
                   type="spppb:RteGrpOfferKeyType" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AcceptRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrpOfferKey"
                  type="spppb:RteGrpOfferKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="RejectRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="rteGrpOfferKey"
                  type="spppb:RteGrpOfferKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetRteGrpOffersRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <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="AddEgrRteRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="egrRte" type="spppb:EgrRteType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelEgrRteRqstType">
  <complexContent>
    <extension base="spppb:BasicRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetEgrRtesRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="objKey" type="spppb:ObjKeyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<annotation>
  <documentation> -------- Generic Request and Response Definitions
    --------------- </documentation>
</annotation>
<element name="spppUpdateRequest">
  <complexType>
    <sequence>
      <element name="clientTransId" type="spppb:TransIdType"
          minOccurs="0"/>
      <element name="minorVer" type="spppb:MinorVerType"
          minOccurs="0"/>
```

```
      <element name="rqst" type="spppb:BasicRqstType"
          maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
<element name="spppUpdateResponse">
  <complexType>
    <sequence>
      <element name="clientTransId" type="spppb:TransIdType"
          minOccurs="0"/>
      <element name="serverTransId" type="spppb:TransIdType"/>
      <element name="overallResult" type="spppb:ResultCodeType"/>
      <element name="rqstObjResult"
          type="spppb:RqstObjResultCodeType" minOccurs="0"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
<element name="spppQueryRequest">
  <complexType>
    <sequence>
      <element name="minorVer" type="spppb:MinorVerType"
          minOccurs="0"/>
      <element name="rqst" type="spppb:BasicQueryRqstType"/>
    </sequence>
  </complexType>
</element>
<element name="spppQueryResponse">
  <complexType>
    <sequence>
      <element name="overallResult" type="spppb:ResultCodeType"/>
      <element name="resultSet" type="spppb:BasicObjType"
         minOccurs="0" maxOccurs=" unbounded"/>
    </sequence>
  </complexType>
</element>
<element name="spppServerStatusRequest">
  <complexType>
    <sequence>
      <element name="minorVer" type="spppb:MinorVerType"
         minOccurs="0"/>
    </sequence>
  </complexType>
</element>
<element name="spppServerStatusResponse">
  <complexType>
    <sequence>
      <element name="overallResult" type="spppb:ResultCodeType"/>
      <element name="svcMenu" type="spppb:SvcMenuType"/>
```

```
        </sequence>
      </complexType>
    </element>
  </schema>
```

12. Specification Extensibility

TOC

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

TOC

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: Alexander Mayrhofer, Deborah A Guyton, David Schwartz, Lisa Dusseault, Manjul Maharishi, Otmar Lendl, Richard Shockey and Sumanth Channabasappa.

14. References

TOC

14.1. Normative References

TOC

[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 (TXT).
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2277]	Alvestrand, H., "IETF Policy on Character Sets and Languages," BCP 18, RFC 2277, January 1998 (TXT, HTML, XML).
[RFC3629]	Yergeau, F., "UTF-8, a transformation format of ISO 10646," STD 63, RFC 3629, November 2003 (TXT).
[RFC3688]	

	Mealling, M., “The IETF XML Registry,” BCP 81, RFC 3688, January 2004 (TXT).
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005 (TXT, HTML, XML).

14.2. Informative References

TOC

[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 (TXT).
[RFC2781]	Hoffman, P. and F. Yergeau, “UTF-16, an encoding of ISO 10646,” RFC 2781, February 2000 (TXT).
[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 (TXT).
[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 (TXT).
[RFC4725]	Mayrhofer, A. and B. Hoeneisen, “ENUM Validation Architecture,” RFC 4725, November 2006 (TXT).
[RFC5321]	Klensin, J., “Simple Mail Transfer Protocol,” RFC 5321, October 2008 (TXT).
[RFC5486]	Malas, D. and D. Meyer, “Session Peering for Multimedia Interconnect (SPEERMINT) Terminology,” RFC 5486, March 2009 (TXT).

Authors' Addresses

TOC

	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