DRINKS	J-F.M. Mule
Internet-Draft	CableLabs
Intended status: Standards Track	K.C. Cartwright
Expires: March 12, 2012	TNS
	S.A. Ali
	NeuStar
	A.M. Mayrhofer
	enum.at GmbH
	September 09, 2011

Session Peering Provisioning Protocol
draft-ietf-drinks-spprov-10

Abstract

This document defines a protocol for provisioning session establishment data into Session Data Registries and SIP Service Provider data stores. The provisioned data is typically used by various network elements for session peering.
This document describes the Session Peering Provisioning Protocol used by clients to provision registries. The document provides a set of guiding principles for the design of this protocol including extensibility and independent transport definitions, a basic data model and an XML Schema Document.

Status of this Memo

Copyright Notice

described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

*1. Introduction

*2. Terminology

*3. Protocol High Level Design

*3.1. Protocol Layering

*3.2. Protocol Data Model

*3.3. Time Value

*4. Transport Protocol Requirements

*4.1. Connection Oriented

*4.2. Request and Response Model

*4.3. Connection Lifetime

*4.4. Authentication

*4.5. Authorization

*4.6. Confidentiality and Integrity

*4.7. Near Real Time

*4.8. Request and Response Sizes

*4.9. Request and Response Correlation

*4.10. Request Acknowledgement

*4.11. Mandatory Transport

*5. Base Protocol Data Structures

*5.1. Request and Response Structures

*5.1.1. Update Request and Response Structures

*5.1.1.1. Update Request

*5.1.1.2. Update Response

*5.1.2. Query Request and Response Structures

*5.1.2.1. Query Request

*5.1.2.2. Query Response

*5.2. Response Codes and Messages

*5.3. Basic Object Type and Organization Identifiers

*6. Protocol Commands

*6.1. Add Destination Group Operation

*6.2. Get Destination Groups Operation

*6.3. Add Public Identifier Operation

*6.4. Get Public Identifiers Operation

*6.5. Add Route Group Operation

*6.6. Get Route Groups Operation

*6.7. Add Route Record Operation

*6.8. Get Route Records Operation

*6.9. Add Route Group Offer Operation

*6.10. Accept Route Group Offer Operation

*6.11. Reject Route Group Offer Operation

*6.12. Get Route Group Offers Operation

*6.13. Egress Route Operations

*6.14. Delete Operation

*7. SPPP Examples

*7.1. Add Destination Group

*7.2. Add Route Records

*7.3. Add Route Records -- URIType

*7.4. Add Route Group

*7.5. Add Public Identity -- Successful COR claim

*7.6. Add LRN

*7.7. Add TN Range

*7.8. Add TN Prefix

*7.9. Enable Peering -- Route Group Offer

*7.10. Enable Peering -- Route Group Offer Accept

*7.11. Add Egress Route

*7.12. Get Destination Group

*7.13. Get Public Identity

*7.14. Get Route Group Request

*7.15. Get Route Group Offers Request

*7.16. Get Egress Route

*7.17. Delete Destination Group

*7.18. Delete Public Identity

*7.19. Delete Route Group Request

*7.20. Delete Route Group Offers Request

*7.21. Delete Egress Route

*8. XML Considerations

*8.1. Namespaces

*8.2. Versioning and Character Encoding

*9. Security Considerations

*10. IANA Considerations

*11. Formal Specification

*12. Acknowledgments

*13. References

*13.1. Normative References

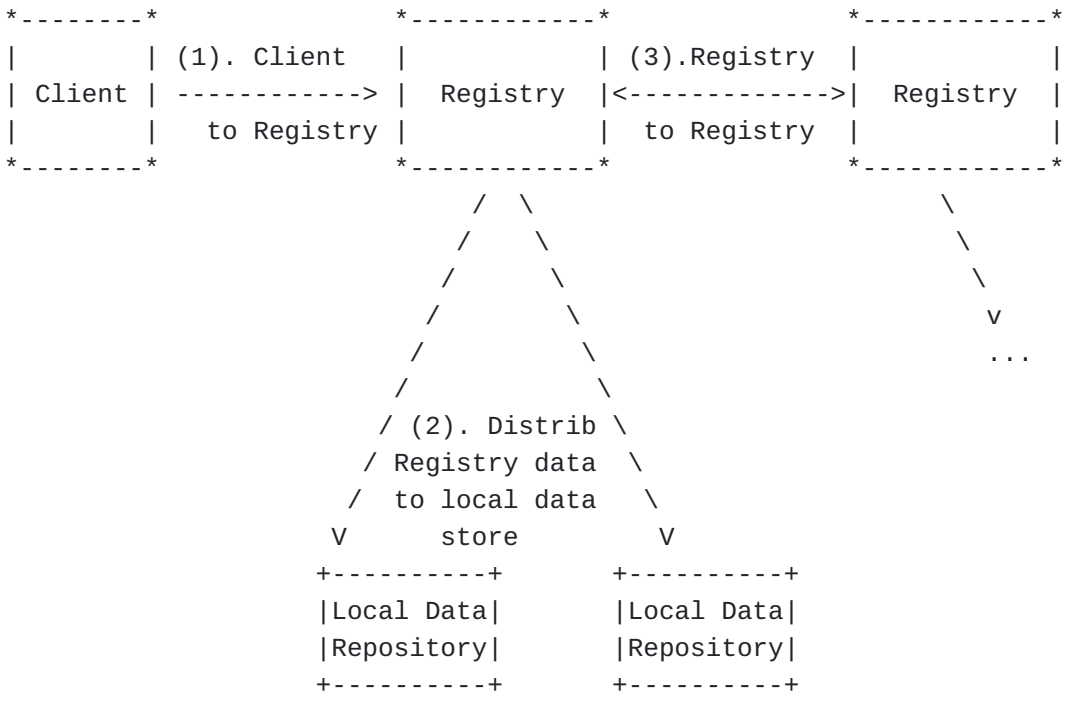
*13.2. Informative References

*Authors' Addresses

1. Introduction

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

Three types of provisioning flows have been described in the use case document: client to registry provisioning, registry to local data repository and registry to registry. This document addresses client to registry aspect to fulfill the need to provision Session Establishment Data (SED). The protocol that supports flow of messages to facilitate client to registry provisioning is referred to as Session Peering Provisioning Protocol (SPPP).
Please 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).



Three Registry Provisioning Flows

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

A "terminating" SIP Service Provider (SSP) provisions SED into the registry to be selectively shared with other peer SSPs. Subsequently, a registry may distribute the provisioned data into local data repositories used for look-up queries (identifier -> URI) or for lookup and location resolution (identifier -> URI -> ingress SBE of terminating SSP). In some cases, the registry may additionally offer a central query resolution service (not shown in the above figure).
A key requirement for the SPPP protocol is to be able to accommodate two basic deployment scenarios:

1. A resolution system returns a Look-Up Function (LUF) that comprises of the target domain to assist in call routing (as described in [RFC5486]). In this case, the querying entity may use other means to perform the Location Routing Function (LRF) which in turn helps determine the actual location of the Signaling Function in that domain.

2. A resolution system returns both a Look-Up function (LUF) and Location Routing Function (LRF) to locate the SED data fully.

In terms of protocol design, SPPP is agnostic to the transport. This document includes the description of the data model and the means to enable protocol operations within a request and response structure. To encourage interoperability, the protocol supports extensibility aspects.
Transport requirements are provided in this document to help with the selection of the optimum transport mechanism. ([I-D.ietf-drinks-sppp-over-soap]) identifies a SOAP transport mechanism for SPPP.
This document is organized as follows:

*Section 2 provides the terminology;

*Section 3 provides an overview of the SPPP, including the layering approach, functional entities and data model;

*Section 4 specifies requirements for SPPP transport protocols;

*Section 5 describes the base protocol data structures including the request and response elements (Section 5.1), the response codes and messages (Section 5.2) and the basic object type most first class objects extend from;

*Section 6 and Section 7 describe the main protocol commands and examples;

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

*Section 11 normatively defines the SPPP protocol using its XML Schema Definition.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
This document reuses terms from [RFC3261], [RFC5486], use cases and requirements documented in [I-D.ietf-drinks-usecases-requirements] and the ENUM Validation Architecture [RFC4725].
In addition, this document specifies the following additional terms:

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

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

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

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

A Registry acts as an SPPP server.

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

Within the confines of a Registry, a Registrant is uniquely identified by a well-known ID.

Registrar: In this document we extend the definition of a Registrar from [RFC4725]. A Registrar is an entity that performs provisioning operations on behalf of a Registrant by interacting with the Registry via SPPP operations. In other words the Registrar is the SPPP Client. The Registrar and Registrant roles are logically separate to allow, but not require, a single Registrar to perform provisioning operations on behalf of more than one Registrant.

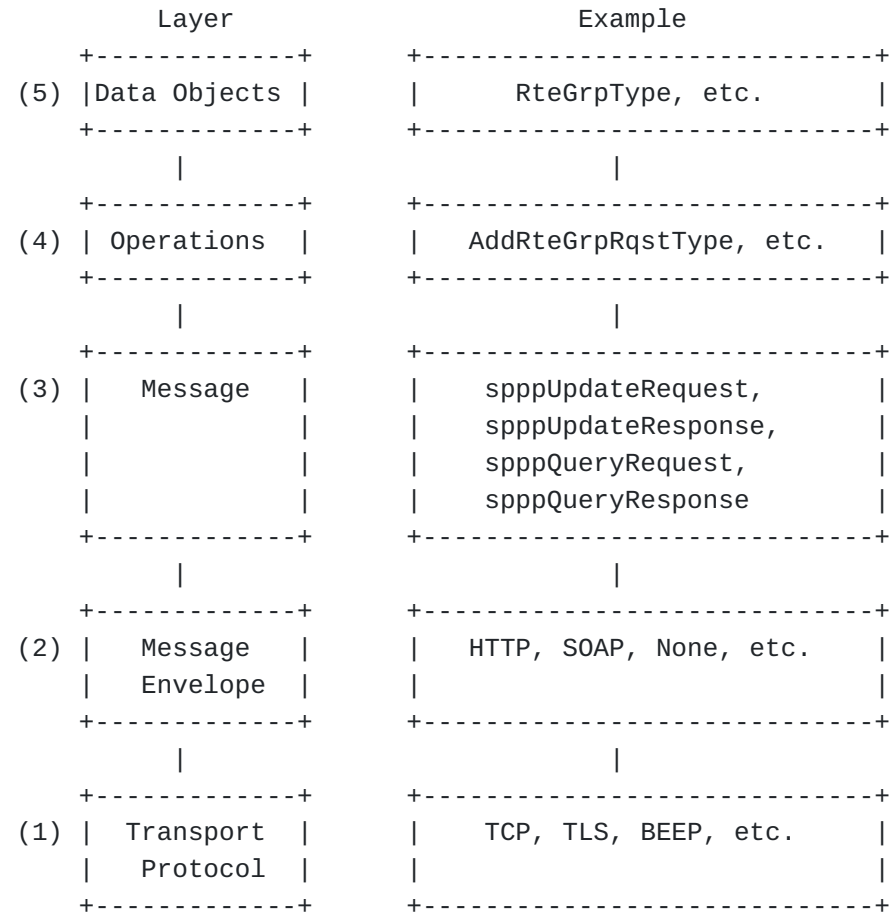
Peering Organization: A Peering Organization is an entity to which a Registrant's Route Groups are made visible using the operations of SPPP.

3. Protocol High Level Design

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

3.1. Protocol 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

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.

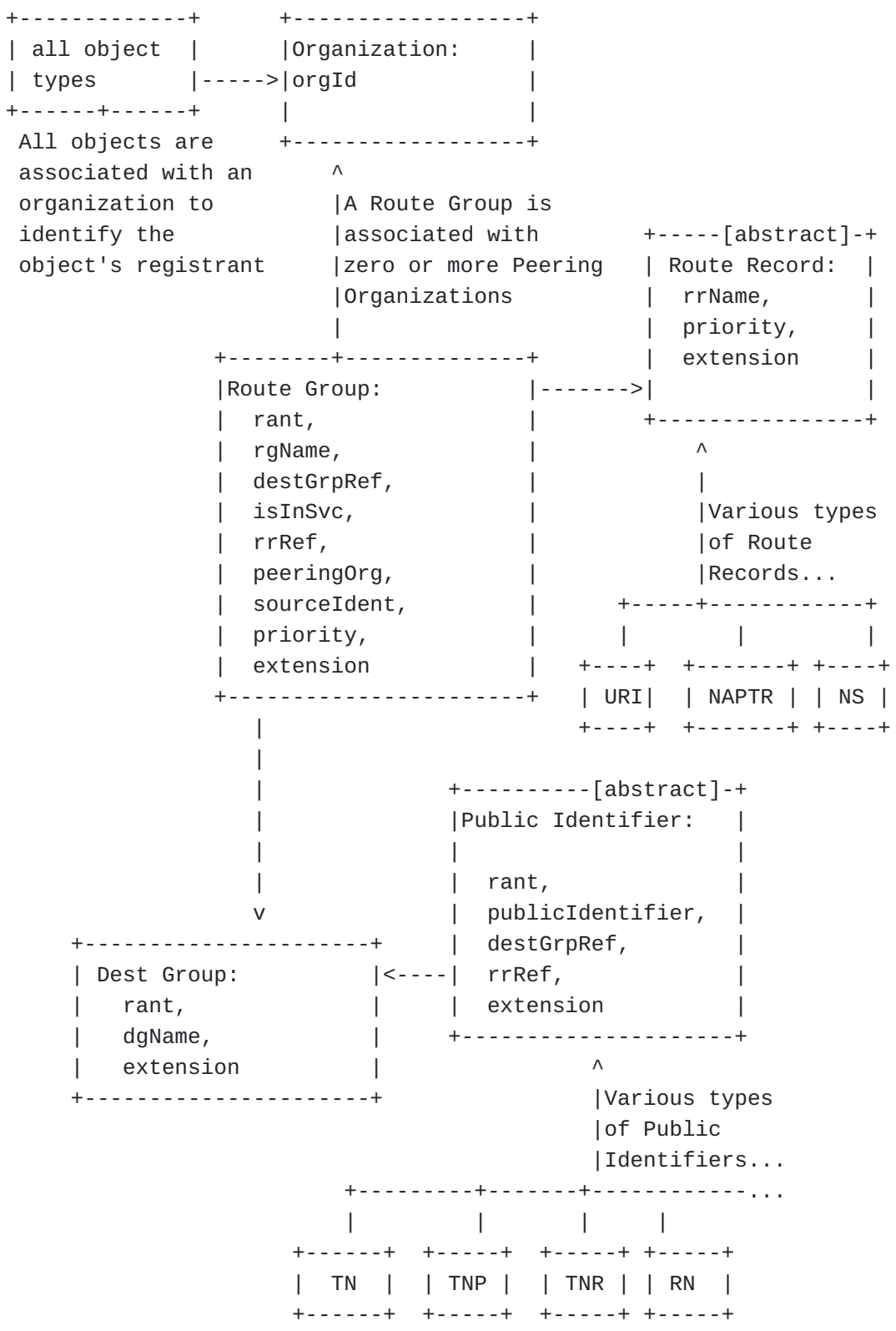
3. The message layer provides a simple, envelope-independent and transport-independent, SPPP wrapper for SPPP request and response messages.

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

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].



SPPP Data Model

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

*Public Identifier:
From a broad perspective a public identifier is a well-known attribute that is used as the key to perform resolution lookups. Within the context of SPPP, a public identifier object can be a telephone number, a range of telephone numbers, a PSTN Routing Number (RN), or a TN prefix.

An SPPP 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.

*Destination Group:
A named 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 their common routing information.

*Route Group:
A Route Group contains a set of Route Record references, a set of Destination Group references, and a set of peering organization identifiers. This is used to establish a three part relationships between a set of Public Identifiers, the routing information (SED) shared across the Public Identifiers, and the list of peering organizations whose query responses from the resolution system may include the routing information from a given route group. In addition, the sourceIdent element within a Route Group, in concert with the set of peering organization identifiers, enables fine-grained source based routing. For 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 Records are generally associated with a Route Group when the SED within is not specific to a Public Identifier.
To support the use cases defined in [I-D.ietf-drinks-usecases-requirements], SPPP 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 the role of a registrant or a peering organization. All SPPP objects are associated with an organization identifier to identify each object's registrant, while tracking the identity of the registrar that provisioned each SPPP object is left as a matter of policy for an SPPP implementation. A Route Group object is also associated with a set of zero or more organization identifiers that identify the peering organization(s) whose resolution query responses may include the routing information (SED) defined in the Route Records within that Route Group. A peering organization is an entity that the registrant intends to share the SED data with. A route group SPPP object is associated with a set of zero or more organization identifiers that identify the peering organizations whose resolution query responses may include the routing information (SED) defined in the route records within that route group.

3.3. Time Value

Some SPPP request and response messages include time value(s) defined as type xs:dateTime, a built-in W3C XML Schema Datatype. Use of unqualified local time value is discouraged as it can lead to interoperability issues. The value of time attribute MUST BE expressed in Coordinated Universal Time (UTC) format without the timezone digits. "2010-05-30T09:30:10Z" is an example of an acceptable time value for use in SPPP messages. "2010-05-30T06:30:10+3:00" is a valid UTC time, but it is not approved for use in SPPP messages.

4. Transport Protocol Requirements

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

4.1. Connection Oriented

The SPPP follows a model where a client establishes a connection to a server in order to further exchange SPPP messages over such point-to-point connection. A transport protocol for SPPP MUST therefore be connection oriented.

4.2. Request and Response Model

Provisioning operations in SPPP follow the request-response model, where a client sends a request message to initiate a transaction and the server responds with a response. Multiple subsequent request-response exchanges MAY be performed over a single persistent connection.
Therefore, a transport protocol for SPPP MUST follow the request-response model by allowing a response to be sent to the request initiator.

4.3. Connection Lifetime

Some use cases involve provisioning a single request to a network element. Connections supporting such provisioning requests might be short-lived, and may be established only on demand. Other use cases involve either provisioning a large dataset, or a constant stream of small updates, either of which would likely require long-lived connections.
Therefore, a protocol suitable for SPPP SHOULD be able to support both short-lived as well as long-lived connections.

4.4. Authentication

All SPPP objects are associated with a registrant identifier. SPPP Clients provisions SPPP objects on behalf of registrants. An authenticated SPP Client is a registrar. Therefore, the SPPP transport protocol MUST provide means for an SPPP server to authenticate an SPPP Client.

4.5. Authorization

After successful authentication of the SPPP client as a registrar the registry performs authorization checks to determine if the registrar is authorized to act on behalf of the Registrant whose identifier is included in the SPPP request. Refer to the Security Considerations section for further guidance.

4.6. Confidentiality and Integrity

In some deployments, the SPPP objects that an SPPP registry manages can be private in nature. As a result it MAY NOT be appropriate to for transmission in plain text over a connection to the SPPP registry. Therefore, the transport protocol SHOULD provide means for end-to-end encryption between the SPPP client and server.
For some SPPP implementations, it may be acceptable for the data to be transmitted in plain text, but the failure to detect a change in data after it leaves the SPPP client and before it is received at the server, either by accident or with a malicious intent, will adversely

affect the stability and integrity of the registry. Therefore, the transport protocol SHOULD provide means for data integrity protection.

4.7. Near Real Time

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

4.8. Request and Response Sizes

Use of SPPP may involve simple updates that may consist of small number of bytes, such as, update of a single public identifier. Other provisioning operations may constitute large number of datasets as in adding millions records to a registry. As a result, a suitable transport protocol for SPPP SHOULD accommodate datasets of various sizes.

4.9. Request and Response Correlation

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

4.10. Request Acknowledgement

Data transported in the SPPP is likely crucial for the operation of the communication network that is being provisioned. A SPPP client responsible for provisioning SED to the registry has a need to know if the submitted requests have been processed correctly.
Failed transactions can lead to situations where a subset of public identifiers or even SSPs might not be reachable, or 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.

4.11. Mandatory Transport

At the time of this writing, a choice of transport protocol has been provided in [I-D.ietf-drinks-sppp-over-soap]. To encourage interoperability, the SPPP server MUST provide support for this transport protocol. With time, it is possible that other transport layer choices may surface that agree with the requirements discussed above.

5. Base Protocol Data Structures

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

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

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

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:BasicUpdateRqstType"
                 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. SPPP server MUST echo back this value to the client in the corresponding response to the incoming request. 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 BasicUpdateRqstType 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 BasicUpdateRqstType object instances in the request at the first error and roll back any BasicUpdateRqstType object instances that had already been processed for that update request. In the "stop and commit" scenario the SPPP server would stop processing BasicUpdateRqstType object instances in the request at the first error but commit any BasicUpdateRqstType object instances that had already been processed for that update request.

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

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

The BasicUpdateRqstType 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

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:BasicUpdateRqstType"/>
          </sequence>
       </extension>
    </complexContent>
  </complexType>
```

An <spppUpdateResponse> contains the elements necessary for the SPPP client to precisely determine the overall 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. When included in the request, the SPPP server MUST return it in the corresponding response message.

*serverTransId: Exactly one server transaction ID that identifies this request for tracking purposes. This value MUST be unique for a given 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 has occurred. It indicates the error condition and the exact request object that contributed to the error. The response code will reflect the exact error. See 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

At times, on behalf of the registrant, the registrar may need to have access to SPPP objects that were previously provisioned in the registry. A few examples include logging, auditing, and pre-provisioning dependency checking. This query mechanism is limited to aid provisioning scenarios and should not be confused with query protocols provided as part of the resolution system (e.g. ENUM and SIP).
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

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

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 overall 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

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]:

- *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	**Result Message**	**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

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", "rgName", "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

This section introduces the basic object type that most first class objects derive from.
All first class objects extend the basic object type BasicObjType that contains the identifier of the registrant organization that owns 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="rant" type="spppb:OrgIdType"/>
    <element name="cDate" type="dateTime" minOccurs="0"/>
    <element name="mDate" type="dateTime" minOccurs="0"/>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>
```

The identifiers used for registrants (rant) 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

This section provides a description of each supported protocol command.

6.1. Add Destination Group Operation

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:BasicUpdateRqstType">
      <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 BasicUpdateRqstType 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 that contains the ID of the registrant organization that owns 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.

6.2. Get Destination Groups Operation

The getDestGrpsRqst operation allows an SPPP client to get the properties of Destination Group objects that a registrar is authorized to view on behalf of the registrant. 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 no ObjKeyType objects are found in the request then the server will return the list of all Destination Group objects in the registry. If no matching records can be located then an empty result set will be returned.
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.

6.3. Add Public Identifier Operation

A 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. SPPP supports the notion of the carrier-of-record as defined in [RFC5067]. Therefore, the registrant under whom the Public Identity is being created can optionally claim to be a carrier-of-record.
SPPP identifies two types of Public Identifiers: telephone numbers (TN), and the routing numbers (RN). SPPP provides structures to manage a single TN, a contiguous range of TNs, and a TN prefix.
The abstract XML schema type definition PubIDType is a generalization for the concrete the Public Identifier schema types. PubIDType element 'dgName' represents the name of the destination group that a given Public Identifier is a member of. Because a Destination Group is uniquely identified by its composite business key, which is comprised of its registrant ID, rantId, and its name, dgName, the Public Identity's containing Destination Group is identified by the Public Identity's dgName element and the Public Identity's registrant ID, rantId, element. 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 using the AddPubIdRqstType operation. To complete the add request, AddPubIdRqstType XML instance is populated into the <spppUpdateRequest> element. A Public Identifier may be provisioned as a member of a Destination Group or provisioned outside of a Destination Group. A Public Identifier that is provisioned as a member of a Destination Group is intended to be associated with its SED through the Route Group(s) that are associated with its containing Destination Group. A Public Identifier that is not provisioned as a member of a Destination Group is intended to be associated with its SED through the Route Records that are directly associated with the Public Identifier. If a Public Identifier being

added already exists then that Public Identifier will be replaced with the newly provisioned Public Identifier.
A telephone number is provisioned using the TNType, an extension of PubIDType. Each TNType object is uniquely identified by the combination of its <tn> element, and the unique key of its parent Destination Group (dgName and rantId). In other words a given telephone number string may exist within one or more Destination Groups, but must not exist more than once within a Destination Group. TNType is defined as follows:

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

TNType consists of the following attributes:

*tn: Telephone number to be added to the registry.

*rrRef: Optional reference to route records that are 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 [RFC5067]). 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 <corDate> 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.

A routing number is provisioned using the RNType, an extension of PubIDType. 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. Each RNType object is uniquely identified by the combination of its <rn> element, and the unique key of its parent Destination Group (dgName and rantId). In other words a given routing number string may exist within one or more Destination Groups, but must not exist more than once within a Destination Group. RNType is defined as follows:

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

RNType has the following attributes:

*rn: Routing Number used as the search key

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

TNRType structure is used to provision a contiguous range of telephone numbers. The object definition requires a starting TN and an ending TN that together define the span of the TN range. Use of TNRType is particularly useful when expressing a TN range that does not include all the TNs within a TN block or prefix. The TNRType definition accommodates the open number plan as well such that the TNs that fall between the start and end TN range may include TNs with different length variance. Whether the registry can accommodate the open number plan semantics is a matter of policy and is beyond the scope of this document. Each TNRType object is uniquely identified by the combination of its <startTn> and <endTn> elements, and the unique key of its parent Destination Group (dgName and rantId). In other words a given TN Range may exist within one or more Destination Groups, but must not exist

more than once within a Destination Group. TNRType object structure definition 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>
      </extension>
    </complexContent>
  </complexType>
```

TNRType has 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

In some cases, it is useful to describe a set of TNs with the help of the first few digits of the telephone number, also referred to as the telephone number prefix or a block. A given TN prefix may include TNs with different length variance in support of open number plan. Once again, whether the registry supports the open number plan semantics is a matter of policy and it is beyond the scope of this document. The TNPType data structure is used to provision a TN prefix. Each TNPType object is uniquely identified by the combination of its <tnPrefix> element, and the unique key of its parent Destination Group (dgName and rantId). TNPType is defined 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>
```

TNPType consists of the following attributes:

*tnPrefix: The telephone number prefix

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

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

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

6.4. Get Public Identifiers Operation

The SPPP client can use the GetPubIdsRqstType in the <spppQueryRequest> structure to obtain information about one or more <pi> objects. If no matching Public Identifiers are found, then an empty result set is returned.
GetPubIdsRqstType object structure is as follows:

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

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

6.5. Add Route Group Operation

As described in the introductory sections, a Route Group represents a combined grouping of Route Records that define route information, Destination Groups that contain a set of Public Identifiers with common routing information, and the list of peer organizations that have access to these public identifiers using this route information. It is this indirect linking of public 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:BasicUpdateRqstType">
        <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 BasicUpdateRqstType and contains one RteGrpType object. The RteGrpType object structure is defined as follows:

```
  <complexType name="RteGrpType">
    <complexContent>
     <extension base="spppb:BasicObjType">
       <sequence>
         <element name="rgName" type="spppb:ObjNameType"/>
         <element name="rrRef" 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="rrKey" 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 that contains the ID of the registrant organization that owns 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.

*rgName: 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).

*rrRef: Set of zero or more objects of type RteRecRefType that house the unique keys of the Route Records that 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 Route 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.

6.6. Get Route Groups Operation

The getRteGrpsRqst operation allows an SPPP client to get the properties of Route Group objects that the registrar is authorized to view on behalf of the registrant. 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 no ObjKeyType objects are found in the request then the server will return the list of all Route Group objects that belongs to the registrant. 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 BasicUpdateRqstType 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.

6.7. Add Route Record Operation

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 serve 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 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:BasicUpdateRqstType">
        <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 BasicUpdateRqstType 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 that contains the ID of the registrant organization that owns 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.

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 DNS name server that points to another DNS server that contains the desired routing information. The NSType is relevant only when the resolution protocol is ENUM. 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="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="IPAddrType">
  <sequence>
    <element name="addr" type="string"/>
    <element name="type" type="spppb:IPType"/>
    <element name="ext" type="spppb:ExtAnyType" minOccurs="0"/>
  </sequence>
</complexType>

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

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

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

The NAPTRType object is composed of the following elements:

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

*svcs: ENUM service(s) that are served by the SBE. This field's value must be of the form specified in [RFC6116] (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 the server will ignore it

*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 DNS 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].

*uri: the URI as defined in [RFC3986]. 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.

6.8. Get Route Records Operation

The getRteRecsRqst operation allows an SPPP client to get the properties of Route Record objects that a registrar is authorized to view on behalf of the registrant. 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 no ObjKeyType objects are found in the request then the server will return the list of all Route Record that belongs to the registrant. 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 BasicUpdateRqstType 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.

6.9. Add Route Group Offer Operation

The list of peer organizations whose resolution responses can include the routing information contained in a given Route Group is controlled by the organization to which a Route Group object belongs (its registrant), and the peer organization that submits resolution requests (a data recipient, 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 <offeredTo> Org 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:BasicUpdateRqstType">
      <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 BasicUpdateRqstType 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 that contains the ID of the registrant organization that owns 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. The server controls the status. 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 UTC when the Route Group Offer was added.

*acceptDateTime: Date and time in UTC 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.

6.10. Accept Route Group Offer Operation

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:BasicUpdateRqstType">
      <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 BasicUpdateRqstType 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.

6.11. Reject Route Group Offer Operation

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 recipient's 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:BasicUpdateRqstType">
      <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 BasicUpdateRqstType 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.

6.12. Get Route Group Offers Operation

The getRteGrpOffersRqst operation allows an SPPP client to get the properties of zero or more Route Group Offer objects that registrar is

authorized to view on behalf of the registrant. 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 belongs to the registrant. 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="offeredBy" type="spppb:OrgIdType"
            minOccurs="0" maxOccurs="unbounded"/>
          <element name="offeredTo" type="spppb:OrgIdType"
            minOccurs="0" maxOccurs="unbounded"/>
          <element name="status" type="spppb:RteGrpOfferStatusType"
            minOccurs="0"/>
          <element name="rteGrpOfferKey"
            type="spppb:RteGrpOfferKeyType" minOccurs="0"
            maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

The GetRteGrpOffersRqstType object is composed of the following elements:

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

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

*status: The status of the offer, offered or accepted. Only offers in the specified status should be included in the result set. If

this element is not present then the status of the offer should not be considered in the query. The result set is also subject to other query criteria in the request.

*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.

6.13. Egress Route Operations

In a high-availability environment, the originating SSP likely has more than one egress paths to the ingress SBE of the target SSP. If the originating SSP wants to exercise greater control and choose a specific egress SBE to be associated to the target ingress SBE, it can do so using the AddEgrRteRqstType object.
Lets assume that the target SSP has offered to share one or more ingress route information and that the originating SSP has accepted the offer. In order to add the egress route to the registry, the originating SSP uses a valid regular expression to rewrite ingress route in order to include the egress SBE information. Also, more than one egress route can be associated with a given ingress route in support of fault-tolerant configurations. The supporting SPPP structure provides a way to include route precedence information to help manage traffic to more than one outbound egress SBE.
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 that contains the ID of the registrant organization that owns 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:BasicUpdateRqstType">
      <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>
```

6.14. Delete Operation

In order to remove an object from the registry, an authorized entity can send the <spppUpdateRequest> to the registry with a corresponding delete BasicUpdateRqstType object. Each 'Add' operation in SPPP has a corresponding 'Del' operation, which is used to delete the respective object type from the registry. If the entity that issued the command is not authorized to perform this operation an appropriate error code will be returned in the <spppUpdateRespnonse> message.
As an example, DelPubIdRqstType is used to delete Public Identifiers
The DelPubIdsRqstType object definition is shown below:

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

When an object is deleted, any references to that object must of course also be removed as the SPPP server implementation fulfills the deletion request. Furthermore, the deletion of a composite object must also result in the deletion of the objects it contains. As a result, the following rules apply to the deletion of SPPP object types:

*Destination Groups: When a destination group is deleted all public identifiers within that destination group must also be automatically deleted by the SPPP implementation as part of fulfilling the deletion request. And any references between that destination group and any route group must be automatically removed by the SPPP implementation as part of fulfilling the deletion request.

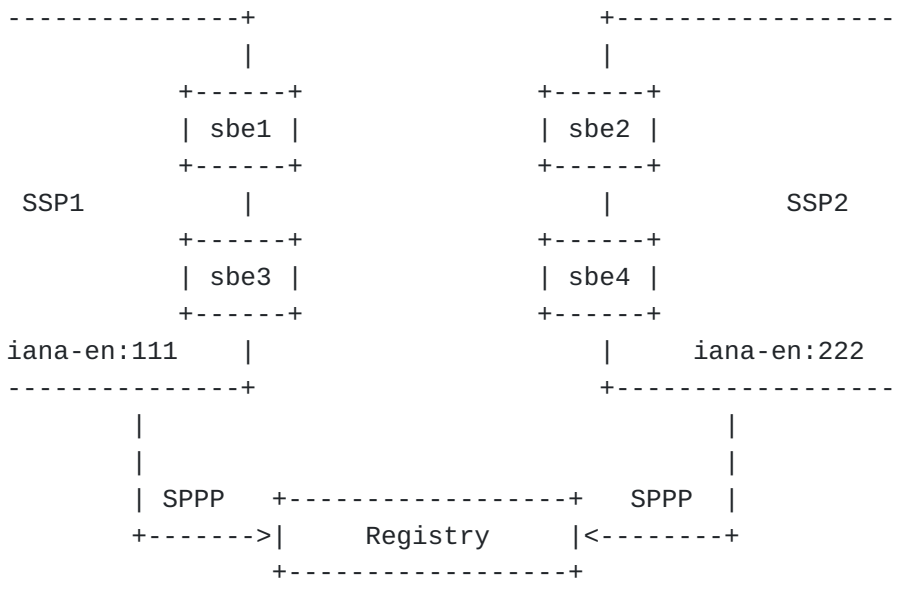
*Route Groups: When a route group is deleted any references between that route group and any destination group must be automatically removed by the SPPP implementation as part of fulfilling the deletion request. Similarly any references between that route group and any route records must be removed by the SPPP implementation as part of fulfilling the deletion request. Furthermore, route group offers relating that route group must also be deleted as part of fulfilling the deletion request.

*Route Records: When a route record is deleted any references between that route record and any route group must be removed by the SPPP implementation as part of fulfilling the deletion request.

*Public Identifiers: When a public identifier is deleted any references between that public identifier and its containing destination group must be removed by the SPPP implementation as part of fulfilling the deletion request. And any route records contained directly within that Public Identifier must be deleted by the SPPP implementation as part of fulfilling the deletion request.

7. SPPP Examples

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 is left out. The SPPP 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 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

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:rant>iana-en:222</ns1:rant>
      <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_5555</clientTransId>
  <serverTransId>tx_id_12346</serverTransId>
  <overallResult>
    <code>1000</code>
    <msg>success</msg>
  </overallResult>
</spppUpdateResponse>
```

7.2. Add Route Records

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">
      <rant>iana-en:222</rant>
      <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

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 xsi:type="ns1:URIType">
      <rant>iana-en:222</rant>
      <rrName>RTE_SSP2_SBE4</rrName>
      <ere>^(.*)$</ere>
      <uri>sip:\1;npdi@sbe4.ssp2.example.com</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

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>
      <rant>iana-en:222</rant>
      <rgName>RTE_GRP_SSP2_1</rgName>
      <rrRef>
        <rrKey>
          <rant>iana-en:222</rant>
          <name>RTE_SSP2_SBE2</name>
        </rrKey>
        <priority>100</priority>
      </rrRef>
      <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>
```

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 xsi:type="ns1:TNType">
      <rant>iana-en:222</rant>
      <cDate>2010-05-30T09:30:10Z</cDate>
      <dgName>DEST_GRP_SSP2_1</dgName>
      <tn>+12025556666</tn>
      <corInfo>
        <corClaim>true</corClaim>
      </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 is processed successfully. In the response message, the registry sets the value of <cor> to "true" in order to confirm SSP2 claim as the carrier of record and the <corDate> reflects the time when the carrier of record claim is processed.

```
<?xml version="1.0" encoding="UTF-8"?>
<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 xsi:type="ns1:TNType">
        <rant>iana-en:222</rant>
        <cDate>2010-05-30T09:30:10Z</cDate>
        <dgName>DEST_GRP_SSP2_1</dgName>
        <tn>+12025556666</tn>
        <corInfo>
          <corClaim>true</corClaim>
          <cor>true</cor>
          <corDate>2010-05-30T09:30:11Z</corDate>
        </corInfo>
      </pi>
    </rqstObj>
  </rqstObjResult>
</spppUpdateResponse>
```

7.6. Add LRN

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">
      <rant>iana-en:222</rant>
      <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

Next, SSP2 activates a block of ten thousand TNs and associate it to a destination 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:AddPubIdRqstType">
    <pi xsi:type="ns1:TNRType">
      <rant>iana-en:222</rant>
      <dgName>DEST_GRP_SSP2_1</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 Prefix

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

```
<?xml version="1.0" encoding="UTF-8"?>
<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 xsi:type="ns1:TNPType">
      <rant>iana-en:222</rant>
      <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>
```

7.9. Enable Peering -- Route Group Offer

In order for SSP1 to complete session establishment for a destination TN where the target subscriber has a retail relationship with SSP2, it first requires an asynchronous bi-directional handshake to show mutual consent. To start the process, SSP2 initiates the peering handshake by offering SSP1 access to its 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>
      <rant>iana-en:222</rant>
      <rteGrpOfferKey>
        <rteGrpKey>
          <rant>iana-en:222</rant>
          <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>
```

7.10. 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>
        <rant>iana-en:222</rant>
        <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>
```

7.11. 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>
      <rant>iana-en:111</rant>
      <egrRteName>EGR_RTE_01</egrRteName>
      <pref>50</pref>
      <regxRewriteRule>
        <ere>^(.*@)(.*)$</ere>
        <repl>\1\2?route=sbe1.ssp1.example.com</repl>
      </regxRewriteRule>
      <ingrRteRec>
        <rant>iana-en:222</rant>
        <name>SSP2_RTE_REC_3</name>
      </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>
```

7.12. Get Destination Group

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>
      <rant>iana-en:222</rant>
      <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">
    <rant>iana-en:222</rant>
    <dgName>DEST_GRP_SSP2_1</dgName>
  </resultSet>
</spppQueryResponse>
```

7.13. Get Public Identity

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">
      <rant>iana-en:222</rant>
      <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">
    <rant>iana-en:222</rant>
    <dgName>DEST_GRP_1</dgName>
    <tn>+12025556666</tn>
    <corInfo>
      <corClaim>true</corClaim>
      <cor>true</cor>
      <corDate>2010-05-30T09:30:10Z</corDate>
    </corInfo>
  </resultSet>
</spppQueryResponse>
```

7.14. 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>
      <rant>iana-en:222</rant>
      <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">
    <rant>iana-en:222</rant>
    <rgName>RTE_GRP_SSP2_1</rgName>
    <rrRef>
      <rrKey>
        <rant>iana-en:222</rant>
        <name>RTE_SSP2_SBE2</name>
      </rrKey>
      <priority>100</priority>
    </rrRef>
    <rrRef>
      <rrKey>
        <rant>iana-en:222</rant>
        <name>RTE_SSP2_SBE4</name>
      </rrKey>
      <priority>101</priority>
    </rrRef>
    <dgName>DEST_GRP_SSP2_1</dgName>
    <isInSvc>true</isInSvc>
    <priority>10</priority>
  </resultSet>
</spppQueryResponse>
```

7.15. Get Route Group Offers Request

SSP2 fetches the last provisioned route group 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">
    <offeredTo>iana-en:111</offeredTo>
  </rqst>
</spppQueryRequest>
```

Registry processes 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">
    <rant>iana-en:222</rant>
    <rteGrpOfferKey>
      <rteGrpKey>
        <rant>iana-en:222</rant>
        <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>
```

7.16. Get Egress Route

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

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
    <objKey>
      <rant>iana-en:111</rant>
      <name>EGR_RTE_01</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:EgrRteType">
    <rant>iana-en:111</rant>
    <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>
      <rant>iana-en:222</rant>
      <name>RTE_GRP_SSP2_1</name>
    </ingressRte>
  </resultSet>
</spppQueryResponse>
```

7.17. Delete Destination Group

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>
      <rant>iana-en:222</rant>
      <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>
```

7.18. Delete Public Identity

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">
      <rant>iana-en:222</rant>
      <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>
```

7.19. Delete Route Group Request

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>
      <rant>iana-en:222</rant>
      <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>
```

7.20. Delete Route Group Offers Request

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>
        <rant>iana-en:222</rant>
        <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>
```

7.21. Delete Egress Route

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>
      <rant>iana-en:111</rant>
      <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

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

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

8.2. Versioning and Character Encoding

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

Conformant XML parsers recognize both UTF-8 (defined in [RFC3629]) and UTF-16 (defined in [RFC2781]); per [RFC2277] UTF-8 is the RECOMMENDED character encoding for use with SPPP.
Character encodings other than UTF-8 and UTF-16 are allowed by XML. UTF-8 is the default encoding assumed by XML in the absence of an "encoding" attribute or a byte order mark (BOM); thus, the "encoding" attribute in the XML declaration is OPTIONAL if UTF-8 encoding is used. SPPP clients and servers MUST accept a UTF-8 BOM if present, though emitting a UTF-8 BOM is NOT RECOMMENDED.
Example XML declarations:

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

9. Security Considerations

Many SPPP implementations manage data that is considered confidential and critical. Furthermore, SPPP implementations can support provisioning activities for multiple registrars and registrants. As a result any SPPP implementation must address the requirements for confidentiality, authentication, and authorization.
With respect to confidentiality and authentication, the transport protocol requirements section of this document contains security properties that the transport protocol must provide so that authenticated endpoints can exchange data confidentially and with integrity protection. Refer to that section and the resulting transport protocol specification document for the specific solutions to authentication and confidentiality.
With respect to authorization, the SPPP server implementation must define and implement a set of authorization rules that precisely address (1) which registrars will be authorized to create/modify/delete each SPPP object type for given registrant(s) and (2) which registrars will be authorized to view/get each SPPP object type for given registrant(s). These authorization rules are a matter of policy and are not specified within the context of SPPP. However, any SPPP implementation must specify these authorization rules in order to function in a reliable and safe manner.

In some situations, it may be required to protect against denial of involvement (see [RFC4949]) and tackle non-repudiation concerns in regards to SPPP messages. This type of protection is useful to satisfy authenticity concerns related to SPPP messages beyond the end-to-end connection integrity, confidentiality, and authentication protection that the transport layer provides. This is an optional feature and some SPPP implementations MAY provide support for it.
It is not uncommon for the logging systems to document on-the-wire messages for various purposes, such as, debug, audit, and tracking. At the minimum, the various support and administration staff will have access to these logs. Also, if an unprivileged user gains access to the SPPP deployments and/or support systems, it will have access to the information that is potentially deemed confidential. To manage information disclosure concerns beyond the transport level, SPPP implementations MAY provide support for encryption at the SPPP object level.
Anti-replay protection ensures that a given SPPP object replayed at a later time doesn't affect the integrity of the system. SPPP provides at least one mechanism to fight against replay attacks. Use of the optional client transaction identifier allows the SPPP client to correlate the request message with the response and to be sure that it is not a replay of a server response from earlier exchanges. Use of unique values for the client transaction identifier is highly encouraged to avoid chance matches to a potential replay message.
The SPPP client or registrar can be a separate entity acting on behalf of the registrant in facilitating provisioning transactions to the registry. Further, the transport layer provides end-to-end connection protection between SPPP client and the SPPP server. Therefore, man-in-the-middle attack is a possibility that may affect the integrity of the data that belongs to the registrant and/or expose peer data to unintended actors in case well-established peering relationships already exist.

10. IANA Considerations

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

Registration request for the SPPP XML namespace:
urn:ietf:params:xml:ns:sppp:base:1
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.
Registration request for the XML schema:
URI: urn:ietf:params:xml:schema:sppp:1
Registrant Contact: IESG
XML: See the "Formal Specification" section of this document (Section 11).

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

This section provides the draft XML Schema Definition for SPPP.

```
<?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="rgName" type="spppb:ObjNameType"/>
         <element name="rrRef" 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="TNType">
  <complexContent>
    <extension base="spppb:PubIdType">
      <sequence>
        <element name="tn" type="string"/>
        <element name="rrRef" 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>
    </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"/>
        <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"/>
         <element name="priority" type="unsignedShort" minOccurs="0"/>
       </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="rant" type="spppb:OrgIdType"/>
    <element name="cDate" type="dateTime" minOccurs="0"/>
    <element name="mDate" 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="rant" 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="rrKey" 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="corDate" 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:BasicUpdateRqstType"/>
        </sequence>
     </extension>
  </complexContent>
</complexType>

<complexType name="BasicUpdateRqstType" 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:BasicUpdateRqstType">
      <sequence>
        <element name="rteGrp" type="spppb:RteGrpType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelRteGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <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:BasicUpdateRqstType">
      <sequence>
        <element name="rteRec" type="spppb:RteRecType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelRteRecRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <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:BasicUpdateRqstType">
      <sequence>
        <element name="destGrp" type="spppb:DestGrpType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelDestGrpRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <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:BasicUpdateRqstType">
      <sequence>
        <element name="pi" type="spppb:PubIdType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelPubIdRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <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"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AddRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <sequence>
        <element name="rteGrpOffer" type="spppb:RteGrpOfferType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelRteGrpOfferRqstType">
  <complexContent>
```

```
    <extension base="spppb:BasicUpdateRqstType">
      <sequence>
        <element name="rteGrpOfferKey"
                   type="spppb:RteGrpOfferKeyType" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AcceptRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <sequence>
        <element name="rteGrpOfferKey"
                  type="spppb:RteGrpOfferKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="RejectRteGrpOfferRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <sequence>
        <element name="rteGrpOfferKey"
                  type="spppb:RteGrpOfferKeyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="GetRteGrpOffersRqstType">
  <complexContent>
    <extension base="spppb:BasicQueryRqstType">
      <sequence>
        <element name="offeredBy" type="spppb:OrgIdType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="offeredTo" type="spppb:OrgIdType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="status" type="spppb:RteGrpOfferStatusType"
          minOccurs="0"/>
        <element name="rteGrpOfferKey"
          type="spppb:RteGrpOfferKeyType" minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="AddEgrRteRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <sequence>
```

```
        <element name="egrRte" type="spppb:EgrRteType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="DelEgrRteRqstType">
  <complexContent>
    <extension base="spppb:BasicUpdateRqstType">
      <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:BasicUpdateRqstType"
          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. Acknowledgments

This document is a result of various discussions held in the DRINKS working group and within the DRINKS protocol design team, which is comprised of the following individuals, in alphabetical order: Alexander Mayrhofer, Deborah A Guyton, David Schwartz, Lisa Dusseault,

Manjul Maharishi, Mickael Marrache, Otmar Lendl, Richard Shockey, Samuel Melloul, and Sumanth Channabasappa.

13. References

13.1. Normative References

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC2277]	Alvestrand, H.T., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.
[RFC3629]	Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
[RFC3688]	Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
[RFC3986]	Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
[RFC5067]	Lind, S. and P. Pfautz, "Infrastructure ENUM Requirements", RFC 5067, November 2007.
[RFC4949]	Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
[I-D.ietf-drinks-sppp-over-soap]	Cartwright, K and V Bhatia, "SPPP Over SOAP and HTTP", Internet-Draft draft-ietf-drinks-sppp-over-soap-07, November 2011.

13.2. Informative References

[RFC5321]	Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.
[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.
[RFC6116]	Bradner, S., Conroy, L. and K. Fujiwara, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", RFC 6116, March 2011.
[RFC4725]	Mayrhofer, A. and B. Hoeneisen, "ENUM Validation Architecture", RFC 4725, November 2006.
[RFC5486]	Malas, D. and D. Meyer, "Session Peering for Multimedia Interconnect (SPEERMINT) Terminology", RFC 5486, March 2009.
[RFC2781]	Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000.

[I-D.ietf-drinks-usecases-requirements]	Channabasappa, S, "Data for Reachability of Inter/tra-NetworK SIP (DRINKS) Use cases and Protocol Requirements", Internet-Draft draft-ietf-drinks-usecases-requirements-06, August 2011.

Authors' Addresses

Jean-Francois Mule Mule CableLabs 858 Coal Creek Circle Louisville, CO 80027 USA EMail: jfm@cablelabs.com

Kenneth Cartwright Cartwright TNS 1939 Roland Clarke Place Reston, VA 20191 USA EMail: kcartwright@tnsi.com

Syed Wasim Ali Ali NeuStar 46000 Center Oak Plaza
Sterling, VA 20166 USA EMail: syed.ali@neustar.biz

Alexander Mayrhofer Mayrhofer enum.at GmbH Karlsplatz 1/9
Wien, A-1010 Austria EMail: alexander.mayrhofer@enum.at