DRINKS
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2012

J-F. Mule
CableLabs
K. Cartwright
TNS
S. Ali
NeuStar
A. Mayrhofer
enum.at GmbH
V. Bhatia
TNS

March 12, 2012

Session Peering Provisioning Framework (SPPF)
draft-ietf-drinks-spp-framework-01

Abstract

This document specifies the data model and the overall structure for a framework to provision session establishment data into Session Data Registries and SIP Service Provider data stores. The framework is called the Session Peering Provisioning Framework (SPPF). The provisioned data is typically used by network elements for session peering.

Status of this Memo

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

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

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

This Internet-Draft will expire on September 13, 2012.

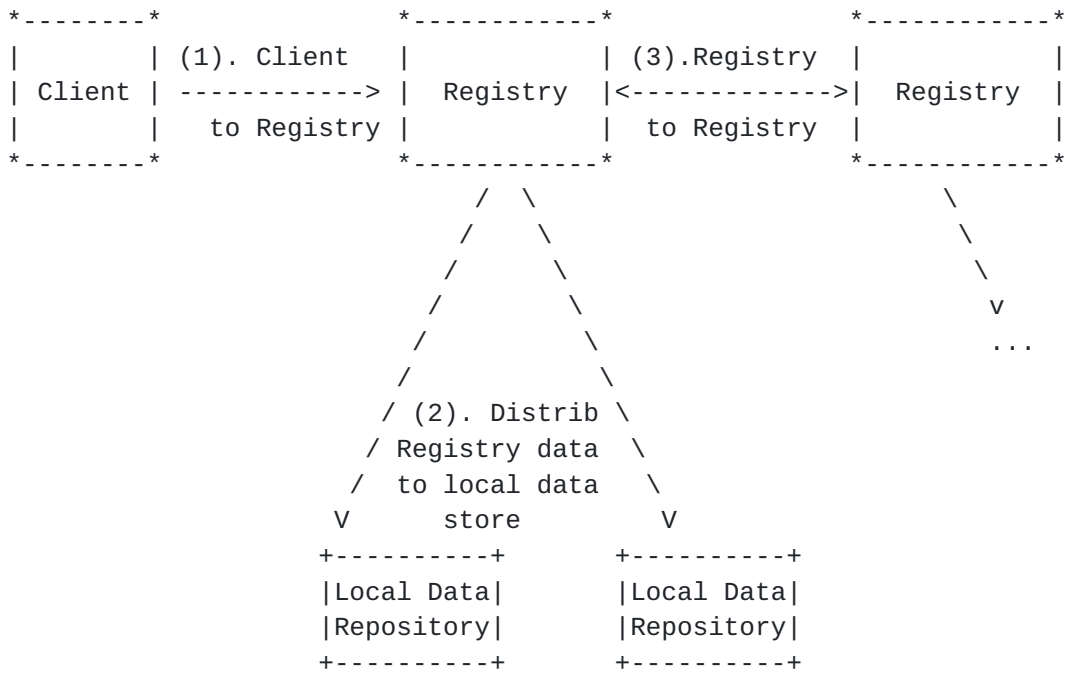
Table of Contents

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 framework for these registries. This framework 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 framework have been documented in [RFC6461]. 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 framework that supports flow of messages to facilitate client to registry provisioning is referred to as Session Peering Provisioning Framework (SPPF).

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

Figure 1

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

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

A key requirement for the SPPF 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 framework design, SPPF is agnostic to the transport protocol. This document includes the specification of the data model and identifies, but does not specify, the means to enable protocol operations within a request and response structure. That aspect of the specification has been delegated to the "protocol" specification for the framework. To encourage interoperability, the framework supports extensibility aspects.

Transport requirements are provided in this document to help with the selection of the optimum transport mechanism. The SPP Protocol over SOAP document identifies a protocol for SPPF that uses SOAP/HTTP as the transport mechanism.

This document is organized as follows:

- Section 2 provides the terminology;
- Section 3 provides an overview of SPPF, including the functional entities and data model;
- Section 4 specifies requirements for SPPF transport protocols;
- Section 5 describes the base framework data structures, the generic response types that MUST be supported by a conforming "protocol" specification, and the basic object type most first class objects extend from;
- Section 6 detailed description of the data model object specifications;
- Section 8 defines XML considerations that XML parsers must meet to conform to this specification;
- Section 11 normatively defines the SPPF 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 [RFC6461] and the ENUM Validation Architecture [RFC4725].

In addition, this document specifies the following additional terms:

SPPF: Session Peering Provisioning Framework, the framework used by a transport protocol to provision data into a Registry (see arrow labeled "1." in Figure 1 of [RFC6461]). 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 [RFC6461]).

Client: An application that supports an SPPF 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 SPPF 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 [RFC6461], 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 SPPF operations. In other words the Registrar is the SPPF 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 SPPF.

3. Framework High Level Design

This section introduces the structure of the data model and provides the information framework for the SPPF. The data model is defined along with all the objects manipulated by the protocol and their relationships.

3.1. Framework Data Model

The data model illustrated and described in Figure 2 defines the logical objects and the relationships between these objects that the SPPF protocol supports. SPPF defines the protocol operations through which an SPPF 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 [RFC6461].

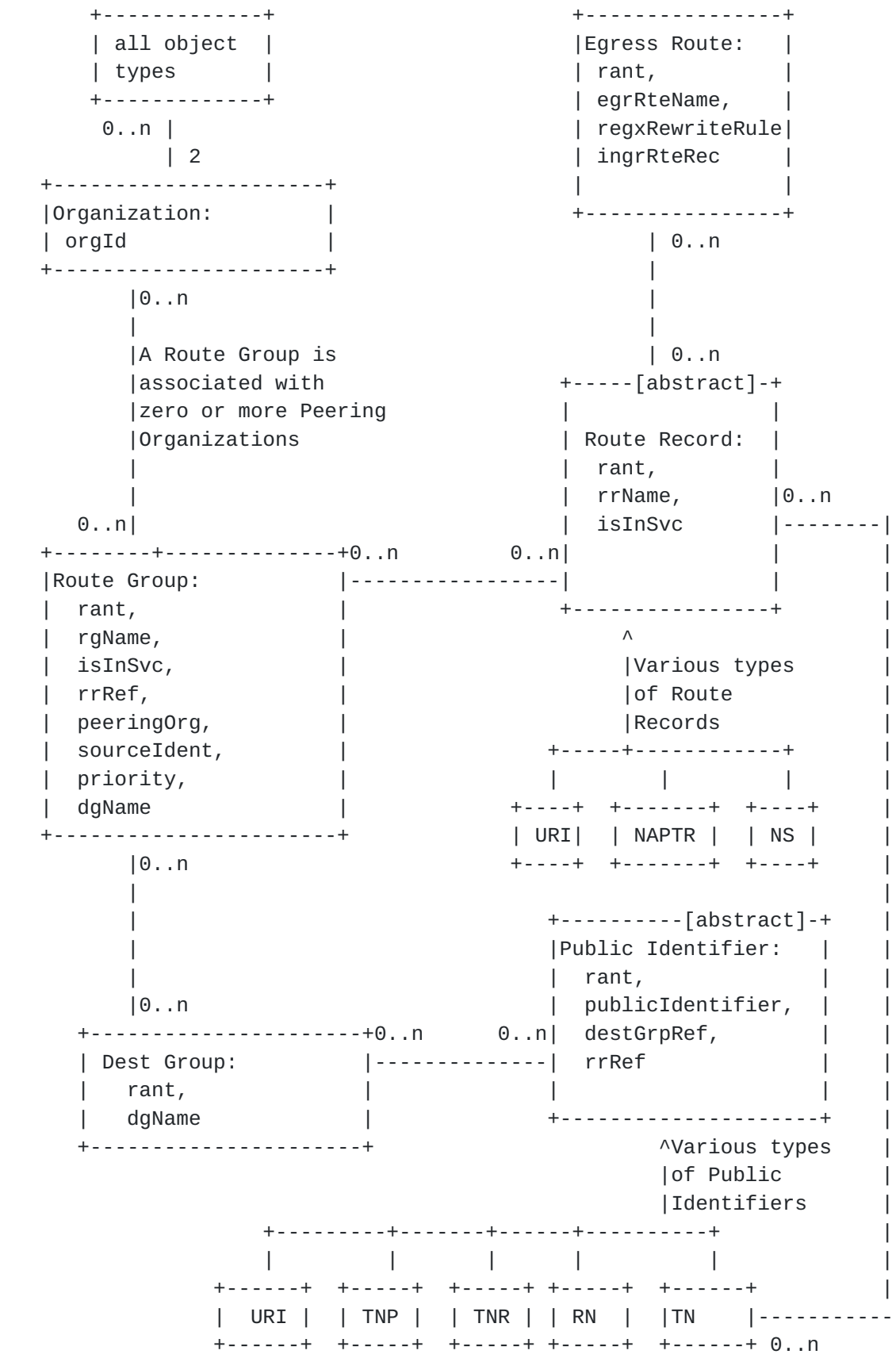

all object types
0..n
2
Organization: orgId
0..n
A Route Group is associated with zero or more Peering Organizations
0..n
Route Group: rant, rgName, isInSvc, rrRef, peeringOrg, sourceIdent, priority, dgName
0..n
Dest Group: rant, dgName
0..n
0..n
URI
TNP
TNR
Egress Route: rant, egrRteName, regxRewriteRule ingrRteRec
0..n
0..n
[abstract]
Route Record: rant, rrName, isInSvc
0..n
0..n
Various types of Route Records
URI
NAPTR
NS
[abstract]
Public Identifier: rant, publicIdentifier, destGrpRef, rrRef
^Various types of Public Identifiers
URI
TNP
TNR
RN
TN
0..n

Figure 2

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 SPPF, a public identifier object can be a Telephone Number (TN), a range of Telephone Numbers, a PSTN Routing Number (RN), a TN prefix, or a URI.

  An SPPF 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 [RFC6461], SPPF framework defines three type of Route Records: URIRteRecType, NAPTRType, and NSType. These Route Records extend the abstract type RteRecType and inherit the common attribute 'priority' that is meant for setting precedence across the route records defined within a Route Group in a protocol agnostic fashion.

- Organization:
  An Organization is an entity that may fulfill any combination of three roles: Registrant, Registrar, and Peering Organization. All objects in SPPF framework are associated with two organization identifiers to identify each object's registrant and registrar. 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.

3.2. Time Value

Some request and response messages in SPPF framework 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 SPPF messages. "2010-05-30T06:30:10+3:00" is a valid UTC time, but it is not approved for use in SPPF messages.

4. Transport Protocol Requirements

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

4.1. Connection Oriented

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

4.2. Request and Response Model

Provisioning operations in SPPF 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 SPPF 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 SPPF SHOULD be able to support both short-lived as well as long-lived connections.

4.4. Authentication

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

4.5. Authorization

After successful authentication of the SPPF 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 SPPF request. Refer to the Security Considerations section for further guidance.

4.6. Confidentiality and Integrity

In some deployments, the SPPF objects that an SPPF 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 SPPF registry. Therefore, the transport protocol SHOULD provide means for end-to-end encryption between the SPPF client and server.

For some SPPF 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 SPPF 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 SPPF 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 dataset as in adding millions records to a registry. As a result, a suitable transport protocol for SPPF SHOULD accommodate dataset of various sizes.

4.9. Request and Response Correlation

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

4.10. Request Acknowledgement

Data transported in the SPPF is likely crucial for the operation of the communication network that is being provisioned. A SPPF 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 SPPF 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 SPP Protocol over SOAP document. To encourage interoperability, the SPPF 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 Framework Data Structures and Response Codes

SPPF contains some common data structures for most of the supported object types. This section describes these common data structures.

5.1. 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 identifier of the registrar organization that created 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="sppfb:OrgIdType"/>
                <element name="rar" type="sppfb:OrgIdType"/>
                <element name="cDate" type="dateTime"
                   minOccurs="0"/>
                <element name="mDate" type="dateTime"
                   minOccurs="0"/>
                <element name="ext" type="sppfb:ExtAnyType"
                   minOccurs="0"/>
        </sequence>
</complexType>
```

The identifiers used for registrants (rant), registrars (rar), 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.

5.2. Various Object Key Types

The SPPF data model contains various object relationships. In some cases, these object relationships are established by embedding the unique identity of the related object inside the relating object. In addition, an object's unique identity is required to Delete or Get the details of an object. The following sub-sections normatively define the various object keys in SPPF and the attributes of those keys .

5.2.1. Generic Object Key Type

Most objects in SPPF are uniquely identified by an object key that has the object's name, object's type and its registrant's organization ID as its attributes. The abstract type called ObjKeyType is where this unique identity is housed. Any concrete representation of the ObjKeyType MUST contain the following:

Object Name: The name of the object.

Registrant Id: The unique organization ID that identifies the Registrant.

Type: The value that represents the type of SPPF object that. This is required as different types of objects in SPPF, that belong to the same registrant, can have the same name.

The structure of abstract ObjKeyType is as follows:

```
<complexType name="ObjKeyType" abstract="true">
        <annotation>
                <documentation>
                        ---- Generic type that represents the
                        key for various objects in SPPF. ----
                </documentation>
        </annotation>
</complexType>
```

5.2.2. Derived Object Key Types

The SPPF data model contains certain objects that are uniquely identified by attributes, different from or in addition to, the attributes in the generic object key described in previous section. These kind of object keys are derived from the abstract ObjKeyType and defined in there own abstract key types. Because these object key types are abstract, these MUST be specified in a concrete form in any conforming SPPF "protocol" specification. These are used in Delete and Get operations, and may also be used in Accept and Reject operations.

Following are the derived object keys in SPPF data model:

o RteGrpOfferKeyType: This uniquely identifies a Route Group object offer. This key type extends from ObjKeyType and MUST also have the organization ID of the Registrant to whom the object is being offered, as one of its attributes. In addition to the Delete and Get operations, these key types are used in

Accept and Reject operations on a Route Group Offer object. The structure of abstract RteGrpOfferKeyType is as follows:

```
<complexType name="RteGrpOfferKeyType"
abstract="true">
        <complexContent>
                <extension base="sppfb:ObjKeyType">
                        <annotation>
                                <documentation>
                                ---- Generic type that represents the
                                key for a object offer. ----
                                </documentation>
                        </annotation>
                </extension>
        </complexContent>
</complexType>
```

A Route Group Offer object MUST use RteGrpOfferKeyType. Refer the "Framework Data Model Objects" section of this document for description of Route Group Offer object.

o PubIdKeyType: This uniquely identifies a Public Identity object. This key type extends from abstract ObjKeyType. Any concrete defintion of PubIdKeyType MUST contain the elements that identify the value and type of Public Identity and also contain the organization ID of the Registrant that is the owner of the Public Identity object. A Public Identity object key in SPPF is uniquely identified by the the registrant's organization ID, the value of the public identity, and, optionally, the Destination Group name the public identiy belongs to. Consequently, any concrete representation of the PubIdKeyType MUST contain the following attributes:

  * Registrant Id: The unique organization ID that identifies the Registrant.

  * Destination Group name: The name of the Destination Group the Public Identity is associated with. This is an optional attribute.

  * Type: The type of Public Identity.

  * Value: The value of the Public Identity.

  The .PubIdKeyType is used in Delete and Get operations on a Public Identifier object.

o The structure of abstract PubIdKeyType is as follows:

```
<complexType name="PubIdKeyType" abstract="true">
        <complexContent>
                <extension base="sppfb:ObjKeyType">
                        <annotation>
                                <documentation>
---- Generic type that represents
the key for a Pub Id. ----
                                </documentation>
                        </annotation>
                </extension>
        </complexContent>
</complexType>
```

A Public Identity object MUST use attributes of PubIdKeyType for its unique identification . Refer the "Framework Data Model Objects" section of this document for a description of Public Identity object.

5.3. Response Message Types

This section contains the listing of response types that MUST be defined by the conforming "protocol" specification and implemented by a conforming SPPF server.

Response Type	Description
Request Succeeded	Any conforming specification MUST define a response to indicate that a given request succeeded.
Request syntax invalid	Any conforming specification MUST define a response to indicate that a syntax of a given request was found invalid.
Request too large	Any conforming specification MUST define a response to indicate that the count of entities in the request is larger than the server is willing or able to process.
Version not supported	Any conforming specification MUST define a response to indicate that the server does not support the version of the SPPF protocol specified in the request.
Command invalid	Any conforming specification MUST define a response to indicate that the operation and/or command being requested by the client is invalid and/or not supported by the server.
System temporarily unavailable	Any conforming specification MUST define a response to indicate that the SPPF server is temporarily not available to serve client request.
Unexpected internal system or server error.	Any conforming specification MUST define a response to indicate that the SPPF server encountered an unexpected error that prevented the server from fulfilling the request.
Attribute value invalid	Any conforming specification MUST define a response to indicate that the SPPF server encountered an attribute or property in the request that had an invalid/bad value. Optionally, the specification MAY provide a way to indicate the Attribute Name and the Attribute Value to identify the object that was found to be invalid.

Object does not exist	Any conforming specification MUST define a response to indicate that an object present in the request does not exist on the SPPF server. Optionally, the specification MAY provide a way to indicate the Attribute Name and the Attribute Value that identifies the non-existent object.
Object status or ownership does not allow for operation.	Any conforming specification MUST define a response to indicate that the operation requested on an object present in the request cannot be performed because the object is in a status that does not allow the said operation or the user requesting the operation is not authorized to perform the said operation on the object. Optionally, the specification MAY provide a way to indicate the Attribute Name and the Attribute Value that identifies the object.

Table 1: Response Types

When the response messages are "parameterized" with the Attribute Name and Attribute Value, then the use of these parameters MUST adhere to the following rules:

- Any value provided for the Attribute Name 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 Attribute Value MUST be the value of the data element to which the preceding Attribute Name refers.

- Response type "Attribute value invalid" SHOULD be used whenever an element value does not adhere to data validation rules.

- Response types "Attribute value invalid" and "Object does not exist" MUST NOT be used interchangeably. Response type "Object does not exist" SHOULD be returned by an Add/Del/Accept/Reject 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 type "Attribute value invalid" SHOULD be returned.

6. Framework Data Model Objects

This section provides a description of the specification of each supported data model object (the nouns) and identifies the commands (the verbs) that MUST be supported for each data model object. However, the specification of the data structures necessary to support each command is delegated to the "protocol" specification.

6.1. Destination Group

As described in the introductory sections, a Destination Group represents a set of Public Identifiers with common routing information. The transport protocol MUST support the ability to Create, Modify, Get, and Delete Destination Groups (refer the "Framework Operations" section of this document for a generic description of various operations).

A Destination Group object MUST be uniquely identified by attributes as defined in the description of "ObjKeyType" in the section "Generic Object Key Type" of this document.

The DestGrpType object structure is defined as follows:

```
<complexType name="DestGrpType">
  <complexContent>
   <extension base="sppfb:BasicObjType">
    <sequence>
       <element name="dgName" type="sppfb: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, registrar organization that provisioned this object on behalf of the registrant, 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.

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

6.2. Public Identifier

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. SPPF 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.

SPPF identifies three types of Public Identifiers: telephone numbers (TN), routing numbers (RN), and URI type of Public Identifiers (like an email address). SPPF provides structures to manage a single TN, a contiguous range of TNs, and a TN prefix. The transport protocol MUST support the ability to Create, Modify, Get, and Delete Public Identifiers (refer the "Framework Operations" section of this document for a generic description of various operations).

A Public Identity object MUST be uniquely identified by attributes as defined in the description of "PubIdKeyType" in the section "Derived Object Key Types" of this document.

The abstract XML schema type definition PubIDType is a generalization for the concrete Public Identifier schema types. PubIDType element 'dgName' represents the name of the destination group that a given Public Identifier MAY be a member of. The PubIDType object structure is defined as follows:

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

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.

A telephone number is provisioned using the TNType, an extension of PubIDType. When a Public Identifier is provisioned as a member of a Destination Group, each TNType object is uniquely identified by the combination of its value contained within <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. A Public Identifier that is not provisioned as a member of a Destination Group is uniquely identified by the combination of its value, and its registrant ID. TNType is defined as follows:

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

<simpleType name="NumberValType">
        <restriction base="token">
                <maxLength value="20"/>
                <pattern value="\+?\d\d*"/>
        </restriction>
</simpleType>
```

TNType consists of the following attributes:

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

o rrRef: Optional reference to route records that are directly associated with the TN Public Identifier. Following the SPPF data model, the route record could be a protocol agnostic

URIRteRecType 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.

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 value inside the <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="sppfb:PubIdType">
    <sequence>
     <element name="rn"
     type="sppfb:NumberValType"/>
     <element name="corInfo"
     type="sppfb:CORInfoType" minOccurs="0"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

RNType has the following attributes:

- rn: Routing Number used as the search key.

o 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 value that in turn is a combination of the <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="sppfb:PubIdType">
    <sequence>
     <element name="range" type="sppfb:NumberRangeType"/>
     <element name="corInfo" type="sppfb:CORInfoType" minOccurs="0"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>

  <complexType name="NumberRangeType">
   <sequence>
    <element name="startTn" type="sppfb:NumberValType"/>
    <element name="endTn" type="sppfb:NumberValType"/>
   </sequence>
  </complexType>
```

TNRType has the following attributes:

o startTn: Starting TN in the TN range

o endTn: The last TN in the TN range

o 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 value in the <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="sppfb:PubIdType">
   <sequence>
    <element name="tnPrefix"
    type="sppfb:NumberValType"/>
    <element name="corInfo"
    type="sppfb:CORInfoType" minOccurs="0"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

TNPType consists of the following attributes:

o tnPrefix: The telephone number prefix

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

In some cases, a Public Identifier may be a URI, such as an email address. The URIPubIdType object is comprised of the data element necessary to house such Public Identifiers. Each URIPubIdType object is uniquely identified by the combination of its value in the <uri> element, and the unique key of its parent Destination Group (dgName and rantId). URIPubIdType is defined as follows:

```
 <complexType name="URIPubIdType">
  <complexContent>
   <extension base="sppfb:PubIdType">
    <sequence>
     <element name="uri"
     type="anyURI"/>
     <element name="ext"
     type="sppfb:ExtAnyType" minOccurs="0"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

URIPubIdType consists of the following attributes:

o uri: The value that acts a Public Identifier.

o ext: Point of extensibility.

6.3. Route Group

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 transport protocol MUST support the ability to Create, Modify, Get, and Delete Route Groups (refer the "Framework Operations" section of this document for a generic description of various operations).

A Route Group object MUST be uniquely identified by attributes as defined in the description of "ObjKeyType" in the section "Generic Object Key Type" of this document.

The RteGrpType object structure is defined as follows:

```
<complexType name="RteGrpType">
  <complexContent>
        <extension base="sppfb:BasicObjType">
          <sequence>
                <element name="rgName" type="sppfb:ObjNameType"/>
                <element name="rrRef" type="sppfb:RteRecRefType"
                  minOccurs="0" maxOccurs="unbounded"/>
                <element name="dgName" type="sppfb:ObjNameType"
                   minOccurs="0" maxOccurs="unbounded"/>
                <element name="peeringOrg" type="sppfb:OrgIdType"
                   minOccurs="0" maxOccurs="unbounded"/>
                <element name="sourceIdent"
                   type="sppfb:SourceIdentType"
                   minOccurs="0" maxOccurs="unbounded"/>
                <element name="isInSvc" type="boolean"/>
                <element name="priority" type="unsignedShort"/>
                <element name="ext" type="sppfb:ExtAnyType"
                   minOccurs="0"/>
                </sequence>
         </extension>
  </complexContent>
</complexType>

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

The RteGrpType object is composed of the following elements:

o 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.

o 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="sourceIdentRegex" type="sppfb:RegexType"/>
        <element name="sourceIdentScheme"
          type="sppfb:SourceIdentSchemeType"/>
        <element name="ext" type="sppfb: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.

6.4. Route Record

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 transport protocol MUST support the ability to Create, Modify, Get, and Delete Route Records (refer the "Framework Operations" section of this document for a generic description of various operations).

A Route Record object MUST be uniquely identified by attributes as defined in the description of "ObjKeyType" in the section "Generic Object Key Type" of this document.

The RteRecType object structure is defined as follows:

```
 <complexType name="RteRecType" abstract="true">
  <complexContent>
   <extension base="sppfb:BasicObjType">
    <sequence>
     <element name="rrName"
     type="sppfb:ObjNameType"/>
     <element name="isInSvc" type="boolean"/>
     <element name="ttl" type="positiveInteger"
                  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).

- isInSvc: A boolean element that defines whether this Route Record is in service or not. The routing information contained in a Route Record which is in service is a candidate for inclusion in resolution responses for Telephone Numbers that are either directly associated to this Route Record, or for Public Identities residing in a Destination Group that is associated to a Route Group which in turn has an association to this Route Record.

- ttl: Number of seconds that an addressing server may cache a particular Route Record.

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 URIRteRecType 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. URIRteRecType fulfills the need for both SIP and ENUM protocols. When a given URIRteRecType 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 URIRteRecType in the NAPTR REGEX parameter.

```
<complexType name="NAPTRType">
  <complexContent>
        <extension base="sppfb:RteRecType">
          <sequence>
                <element name="order" type="unsignedShort"/>
                <element name="flags" type="sppfb:FlagsType"
                   minOccurs="0"/>
                <element name="svcs" type="sppfb:SvcType"/>
                <element name="regx" type="sppfb:RegexParamType"
                   minOccurs="0"/>
                <element name="repl" type="sppfb:ReplType"
                   minOccurs="0"/>
                <element name="ext" type="sppfb:ExtAnyType"
                   minOccurs="0"/>
          </sequence>
        </extension>
  </complexContent>
</complexType>

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

<complexType name="IPAddrType">
  <sequence>
        <element name="addr" type="sppfb:AddrStringType"/>
        <element name="ext" type="sppfb:ExtAnyType"
           minOccurs="0"/>
  </sequence>
  <attribute name="type" type="sppfb:IPType"
     default="v4"/>
</complexType>
```

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

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

<simpleType name="flagsType">
        <restriction base="token">
                <length value="1"/>
                <pattern value="[A-Z]|[a-z]|[0-9]"/>
        </restriction>
</simpleType>
```

The NAPTRType object is composed of the following elements:

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

- svcs: ENUM service(s) that are served by the SBE. This field's value must be of the form specified in [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

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

The URIRteRecType 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.

6.5. Route Group Offer

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 transport protocol MUST support the ability to Create, Modify, Get, Delete, Accept and Reject Route Group Offers (refer the "Framework Operations" section of this document for a generic description of various operations).

A Route Group Offer object MUST be uniquely identified by attributes as defined in the description of "RteGrpOfferKeyType" in the section "Derived Object Key Types" of this document.

The RteGrpOfferType object structure is defined as follows:

```
<complexType name="RteGrpOfferType">
  <complexContent>
        <extension base="sppfb:BasicObjType">
          <sequence>
                <element name="rteGrpOfferKey"
                   type="sppfb:RteGrpOfferKeyType"/>
                <element name="status"
                   type="sppfb:RteGrpOfferStatusType"/>
                <element name="offerDateTime" type="dateTime"/>
                <element name="acceptDateTime" type="dateTime"
                   minOccurs="0"/>
                <element name="ext" type="sppfb:ExtAnyType"
                   minOccurs="0"/>
                </sequence>
         </extension>
  </complexContent>
</complexType>

<complexType name="RteGrpOfferKeyType" abstract="true">
  <annotation>
        <documentation>
        -- Generic type that represents the key for a route
        route group offer. Must be defined in concrete form
        in the transport specificaiton. --
        </documentation>
  </annotation>
</complexType>

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

The RteGrpOfferType object is composed of the following elements:

o 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.

o rteGrpOfferKey: The object that identifies the route that is or has been offered and the organization that it is or has been offered to.

- 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.

6.6. Egress Route

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 EgrRteType object.

A Egress Route object MUST be uniquely identified by attributes as defined in the description of "ObjKeyType" in the section "Generic Object Key Type" of this document.

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 SPPF structure provides a way to include route precedence information to help manage traffic to more than one outbound egress SBE.

The transport protocol MUST support the ability to Add, Modify, Get, and Delete Egress Routes (refer the "Framework Operations" section of this document for a generic description of various operations). The EgrRteType object structure is defined as follows:

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

7. Framework Operations

7.1. Add Operation

Any conforming "protocol" specification MUST provide a definition for the operation that adds one or more SPPF objects into the registry. If the object, as identified by the request attributes that form part of the object's key, does not exist, then the registry MUST create the object. If the object does exist, then the registry MUST replace the current properties of the object with the properties passed in as part of the Add operation.

If the entity that issued the command is not authorized to perform this operation an appropriate error message MUST be returned from amongst the response messages defined in "Response Message Types" section of the document.

7.2. Delete Operation

Any conforming "protocol" specification MUST provide a definition for the operation that deletes one or more SPPF objects from the registry using the object's key.

If the entity that issued the command is not authorized to perform this operation an appropriate error message MUST be returned from amongst the response messages defined in "Response Message Types" section of the document.

When an object is deleted, any references to that object must of course also be removed as the SPPF 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 SPPF object types:

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

by the SPPF 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 SPPF 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 SPPF implementation as part of fulfilling the deletion request. And any route records contained directly within that Public Identifier must be deleted by the SPPF implementation as part of fulfilling the deletion request.

7.3. Get Operations

At times, on behalf of the registrant, the registrar may need to have access to SPPF 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). Any conforming "protocol" specification MUST provide a definition for the operation that queries the details of one or more SPPF objects from the registry using the object's key. If the entity that issued the command is not authorized to perform this operation an appropriate error message MUST be returned from amongst the response messages defined in "Response Message Types" section of the document.

7.4. Accept Operations

In SPPF, a Route Group Offer can be accepted or rejected by, or on behalf of, the registrant to whom the Route Group has been offered (refer "Framework Data Model Objects" section of this document for a description of the Route Group Offer object). The Accept operation is used to accept the Route Group Offers. Any conforming "protocol" specification MUST provide a definition for the operation to accept Route Group Offers by, or on behalf of the Registrant, using the Route Group Offer object key.

Not until access to a Route Group has been offered and accepted will the registrant's organization ID be included in the peeringOrg list in that Route Group object, and that Route Group's peering

information become a candidate for inclusion in the responses to the resolution requests submitted by that registrant. A Route Group Offer that is in the "offered" status is accepted by, or on behalf of, the registrant to which it has been offered. When the Route Group Offer is accepted the the Route Group Offer is moved to the "accepted" status and adds that data recipient's organization ID into the list of peerOrgIds for that Route Group.

If the entity that issued the command is not authorized to perform this operation an appropriate error message MUST be returned from amongst the response messages defined in "Response Message Types" section of the document.

7.5. Reject Operations

In SPPF, a Route Group Offer object can be accepted or rejected by, or on behalf of, the registrant to whom the Route Group has been offered (refer "Framework Data Model Objects" section of this document for a description of the Route Group Offer object). Furthermore, that offer may be rejected, regardless of whether or not it has been previously accepted. The Reject operation is used to reject the Route Group Offers. When the Route Group Offer is rejected that Route Group Offer is deleted, and, if appropriate, the data recipient's organization ID is removed from the list of peeringOrg IDs for that Route Group. Any conforming "protocol" specification MUST provide a definition for the operation to reject Route Group Offers by, or on behalf of the Registrant, using the Route Group Offer object key.

If the entity that issued the command is not authorized to perform this operation an appropriate error message MUST be returned from amongst the response messages defined in "Response Message Types" section of the document.

7.6. Get Server Details Operation

In SPPF, Get Server Details operation can be used to request certain details about the SPPF server that include the SPPF server's current status, the major/minor version of the SPPF protocol supported by the SPPF server.

Any conforming "protocol" specification MUST provide a definition for the operation to request such details from the SPPF server. If the entity that issued the command is not authorized to perform this operation an appropriate error message MUST be returned from amongst the response messages defined in "Response Message Types" section of the document.

8. XML Considerations

XML serves as the encoding format for SPPF, 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 SPPF.

8.1. Namespaces

All SPPF elements are defined in the namespaces in the IANA Considerations section and in the Formal Framework 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 SPPF.

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. SPPF 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 SPPF implementations manage data that is considered confidential and critical. Furthermore, SPPF implementations can support provisioning activities for multiple registrars and registrants. As a result any SPPF 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 SPPF 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 SPPF object type for given registrant(s) and (2) which registrars will be authorized to view/get each SPPF object type for given registrant(s). These authorization rules are a matter of policy and are not specified within the context of SPPF. However, any SPPF 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 SPPF messages. This type of protection is useful to satisfy authenticity concerns related to SPPF messages beyond the end-to-end connection integrity, confidentiality, and authentication protection that the transport layer provides. This is an optional feature and some SPPF 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 SPPF 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, SPPF implementations MAY provide support for encryption at the SPPF object level.

Anti-replay protection ensures that a given SPPF object replayed at a later time doesn't affect the integrity of the system. SPPF provides at least one mechanism to fight against replay attacks. Use of the optional client transaction identifier allows the SPPF 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 SPPF 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 SPPF client and the SPPF 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 SPPF XML namespace:
urn:ietf:params:xml:ns:sppf: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:sppf:1
Registrant Contact: IESG
XML: See the "Formal Specification" section of this document (Section 11).

IANA is requested to create a new SPPF 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 SPPF Protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns:sppfb="urn:ietf:params:xml:ns:sppf:base:1"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:ietf:params:xml:ns:sppf:base:1"
elementFormDefault="qualified" xml:lang="EN">
 <annotation>
  <documentation>
               ---- Generic Object key
               types to be defined by specific
               Transport/Architecture.
               The types defined here can
               be extended by the
               specific architecture to
               define the Object Identifiers ----
  </documentation>
 </annotation>
 <complexType name="ObjKeyType"
  abstract="true">
  <annotation>
   <documentation>
   ---- Generic type that
   represents the key for various
   objects in SPPF. ----
   </documentation>
  </annotation>
 </complexType>

 <complexType name="RteGrpOfferKeyType" abstract="true">
  <complexContent>
   <extension base="sppfb:ObjKeyType">
    <annotation>
     <documentation>
     ---- Generic type
     that represents
     the key for a route
     group offer. ----
     </documentation>
    </annotation>
   </extension>
  </complexContent>
 </complexType>
```

```
    <complexType name="PubIdKeyType" abstract="true">
     <complexContent>
      <extension base="sppfb:ObjKeyType">
       <annotation>
        <documentation>
         ----Generic type that
         represents the key
         for a Pub Id. ----
        </documentation>
       </annotation>
      </extension>
     </complexContent>
    </complexType>

    <annotation>
     <documentation> ---- Object Type Definitions ---- </documentation>
    </annotation>

    <complexType name="RteGrpType">
     <complexContent>
      <extension base="sppfb:BasicObjType">
       <sequence>
        <element name="rgName"
        type="sppfb:ObjNameType"/>
        <element name="rrRef"
        type="sppfb:RteRecRefType"
        minOccurs="0" maxOccurs="unbounded"/>
        <element name="dgName"
        type="sppfb:ObjNameType"
        minOccurs="0" maxOccurs="unbounded"/>
        <element name="peeringOrg"
        type="sppfb:OrgIdType"
        minOccurs="0" maxOccurs="unbounded"/>
        <element name="sourceIdent"
         type="sppfb:SourceIdentType"
         minOccurs="0" maxOccurs="unbounded"/>
        <element name="isInSvc" type="boolean"/>
        <element name="priority" type="unsignedShort"/>
        <element name="ext"
        type="sppfb:ExtAnyType" minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
    <complexType name="DestGrpType">
     <complexContent>
      <extension base="sppfb:BasicObjType">
       <sequence>
```

```
      <element name="dgName"
      type="sppfb:ObjNameType"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="PubIdType" abstract="true">
   <complexContent>
    <extension base="sppfb:BasicObjType">
     <sequence>
      <element name="dgName"
      type="sppfb:ObjNameType" minOccurs="0"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="TNType">
   <complexContent>
    <extension base="sppfb:PubIdType">
     <sequence>
      <element name="tn"
      type="sppfb:NumberValType"/>
      <element name="corInfo"
      type="sppfb:CORInfoType" minOccurs="0"/>
      <element name="rrRef"
      type="sppfb:RteRecRefType"
      minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="TNRType">
   <complexContent>
    <extension base="sppfb:PubIdType">
     <sequence>
      <element name="range"
      type="sppfb:NumberRangeType"/>
      <element name="corInfo"
      type="sppfb:CORInfoType" minOccurs="0"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="TNPType">
   <complexContent>
    <extension base="sppfb:PubIdType">
     <sequence>
      <element name="tnPrefix"
```

```
        type="sppfb:NumberValType"/>
        <element name="corInfo"
        type="sppfb:CORInfoType" minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
    <complexType name="RNType">
     <complexContent>
      <extension base="sppfb:PubIdType">
       <sequence>
        <element name="rn"
        type="sppfb:NumberValType"/>
        <element name="corInfo"
        type="sppfb:CORInfoType" minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
     <complexType name="URIPubIdType">
     <complexContent>
      <extension base="sppfb:PubIdType">
       <sequence>
        <element name="uri"
        type="anyURI"/>
        <element name="ext"
       type="sppfb:ExtAnyType" minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
    <complexType name="RteRecType" abstract="true">
     <complexContent>
      <extension base="sppfb:BasicObjType">
       <sequence>
        <element name="rrName"
        type="sppfb:ObjNameType"/>
        <element name="isInSvc" type="boolean"/>
        <element name="ttl" type="positiveInteger"
                      minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
    <complexType name="NAPTRType">
     <complexContent>
      <extension base="sppfb:RteRecType">
       <sequence>
```

```
      <element name="order" type="unsignedShort"/>
      <element name="flags" type="sppfb:FlagsType" minOccurs="0"/>
      <element name="svcs" type="sppfb:SvcType"/>
      <element name="regx" type="sppfb:RegexParamType" minOccurs="0"/>
      <element name="repl" type="sppfb:ReplType" minOccurs="0"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="NSType">
   <complexContent>
    <extension base="sppfb:RteRecType">
     <sequence>
      <element name="hostName" type="token"/>
      <element name="ipAddr" type="sppfb:IPAddrType"
      minOccurs="0" maxOccurs="unbounded"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="URIRteRecType">
   <complexContent>
    <extension base="sppfb:RteRecType">
     <sequence>
      <element name="ere" type="token" default="^(.*)$"/>
      <element name="uri" type="anyURI"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="RteGrpOfferType">
   <complexContent>
    <extension base="sppfb:BasicObjType">
     <sequence>
      <element name="rteGrpOfferKey" type="sppfb:RteGrpOfferKeyType"/>
      <element name="status" type="sppfb:RteGrpOfferStatusType"/>
      <element name="offerDateTime" type="dateTime"/>
      <element name="acceptDateTime" type="dateTime" minOccurs="0"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </extension>
   </complexContent>
  </complexType>
  <complexType name="EgrRteType">
   <complexContent>
```

```
      <extension base="sppfb:BasicObjType">
       <sequence>
        <element name="egrRteName" type="sppfb:ObjNameType"/>
        <element name="pref" type="unsignedShort"/>
        <element name="regxRewriteRule" type="sppfb:RegexParamType"/>
        <element name="ingrRteRec" type="sppfb:ObjKeyType" minOccurs="0"
        maxOccurs="unbounded"/>
        <element name="ext" type="sppfb: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="sppfb:OrgIdType"/>
      <element name="rar" type="sppfb:OrgIdType"/>
      <element name="cDate" type="dateTime" minOccurs="0"/>
      <element name="mDate" type="dateTime" minOccurs="0"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </complexType>
    <complexType name="RegexParamType">
     <sequence>
      <element name="ere" type="sppfb:RegexType" default="^(.*)$"/>
      <element name="repl" type="sppfb:ReplType"/>
     </sequence>
    </complexType>
    <complexType name="IPAddrType">
     <sequence>
      <element name="addr" type="sppfb:AddrStringType"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
     <attribute name="type" type="sppfb:IPType" default="v4"/>
    </complexType>
    <complexType name="RteRecRefType">
     <sequence>
      <element name="rrKey" type="sppfb:ObjKeyType"/>
      <element name="priority" type="unsignedShort"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </complexType>
```

```
    <complexType name="SourceIdentType">
     <sequence>
      <element name="sourceIdentRegex" type="sppfb:RegexType"/>
      <element name="sourceIdentScheme"
      type="sppfb:SourceIdentSchemeType"/>
      <element name="ext" type="sppfb:ExtAnyType" minOccurs="0"/>
     </sequence>
    </complexType>
    <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="sppfb:ServerStatusType"/>
      <element name="majMinVersion" type="token" maxOccurs="unbounded"/>
      <element name="objURI" type="anyURI" maxOccurs="unbounded"/>
      <element name="extURI" type="anyURI" minOccurs="0"
      maxOccurs="unbounded"/>
     </sequence>
    </complexType>
    <complexType name="ExtAnyType">
     <sequence>
      <any namespace="##other" maxOccurs="unbounded"/>
     </sequence>
    </complexType>
    <simpleType name="FlagsType">
     <restriction base="token">
      <length value="1"/>
      <pattern value="[A-Z]|[a-z]|[0-9]"/>
     </restriction>
    </simpleType>
    <simpleType name="SvcType">
     <restriction base="token">
      <minLength value="1"/>
     </restriction>
    </simpleType>
    <simpleType name="RegexType">
     <restriction base="token">
      <minLength value="1"/>
     </restriction>
    </simpleType>
    <simpleType name="ReplType">
     <restriction base="token">
      <minLength value="1"/>
```

```
      <maxLength value="255"/>
     </restriction>
    </simpleType>
    <simpleType name="OrgIdType">
     <restriction base="token"/>
    </simpleType>
    <simpleType name="ObjNameType">
     <restriction base="token">
      <minLength value="3"/>
      <maxLength value="80"/>
     </restriction>
    </simpleType>
    <simpleType name="TransIdType">
     <restriction base="token">
      <minLength value="3"/>
      <maxLength value="120"/>
     </restriction>
    </simpleType>
    <simpleType name="MinorVerType">
     <restriction base="unsignedLong"/>
    </simpleType>
    <simpleType name="AddrStringType">
     <restriction base="token">
      <minLength value="3"/>
      <maxLength value="45"/>
     </restriction>
    </simpleType>
    <simpleType name="IPType">
     <restriction base="token">
      <enumeration value="v4"/>
      <enumeration value="v6"/>
     </restriction>
    </simpleType>
    <simpleType name="SourceIdentSchemeType">
     <restriction base="token">
      <enumeration value="uri"/>
      <enumeration value="ip"/>
      <enumeration value="rootDomain"/>
     </restriction>
    </simpleType>
    <simpleType name="ServerStatusType">
     <restriction base="token">
      <enumeration value="inService"/>
      <enumeration value="outOfService"/>
     </restriction>
    </simpleType>
    <simpleType name="RteGrpOfferStatusType">
     <restriction base="token">
```

```
      <enumeration value="offered"/>
      <enumeration value="accepted"/>
     </restriction>
    </simpleType>
    <simpleType name="NumberValType">
     <restriction base="token">
      <maxLength value="20"/>
      <pattern value="\+?\d\d*"/>
     </restriction>
    </simpleType>
    <simpleType name="NumberTypeEnum">
     <restriction base="token">
      <enumeration value="TN"/>
      <enumeration value="TNPrefix"/>
      <enumeration value="RN"/>
     </restriction>
    </simpleType>
    <complexType name="NumberType">
     <sequence>
      <element name="value" type="sppfb:NumberValType"/>
      <element name="type" type="sppfb:NumberTypeEnum"/>
     </sequence>
    </complexType>
    <complexType name="NumberRangeType">
     <sequence>
      <element name="startRange" type="sppfb:NumberValType"/>
      <element name="endRange" type="sppfb:NumberValType"/>
     </sequence>
    </complexType>
   </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, David Schwartz, Deborah A Guyton, 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., "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.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC5067] Lind, S. and P. Pfautz, "Infrastructure ENUM Requirements", RFC 5067, November 2007.

13.2. Informative References

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

[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.

[RFC4725] Mayrhofer, A. and B. Hoeneisen, "ENUM Validation Architecture", RFC 4725, November 2006.

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

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

[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.

[RFC6461] Channabasappa, S., "Data for Reachability of Inter-/ Intra-NetworK SIP (DRINKS) Use Cases and Protocol Requirements", RFC 6461, January 2012.

Authors' Addresses

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

Email: jfm@cablelabs.com

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

Email: kcartwright@tnsi.com

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

Email: syed.ali@neustar.biz

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

Email: alexander.mayrhofer@enum.at

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

Email: vbhatia@tnsi.com