Network Working Group A. Shaikh
Internet-Draft Google
Intended status: Informational R. Shakir
Expires: September 10, 2015 BT
K. D'Souza
AT&T
L. Fang
Microsoft
March 9, 2015

Operational Structure and Organization of YANG Models
draft-openconfig-netmod-model-structure-00

Abstract

This document presents an approach for organizing YANG models in a comprehensive structure that defines how individual models may be composed to configure and operate network infrastructure and services. The structure is itself represented as a YANG model rooted at a device, with all of the related component models logically organized in a way that is operationally intuitive.

Status of This Memo

Copyright Notice

1. Introduction

The large number of configuration models recently published cover much of networking protocols and technology and, in theory, enable a programmatic, model-driven approach for configuring network devices. These models have been largely developed individually and in isolation, however, making it challenging to use them together to fully configure a device, or manage a set of devices comprising a service. For example, standard models for interface management [RFC7223] and system management [RFC7317] are available but there is no guidance for how they should be used together, or combined with other models for routing protocols, ACLs, etc. to form a complete model. Recently, some frameworks (e.g., [RTG-CFG] and [RTG-POLICY]) that tie models together have been developed, but they are incomplete, covering only a subset of related models.

1.1. Goals and approach

In this document, we describe a structure for organizing YANG [RFC6020] models that is broadly applicable to physical and virtual devices. Individual models are composed such that the data they define can be accessed in a predictable and operationally intuitive way that is common across implementations. This organization enables several important capabilities:

- a common schema to access data related to all aspects of a device

- an extensible structure that makes it clear where additional models or data should be fit (e.g., using YANG augmentation or imports)

- a place for including metadata that provides useful information about the corresponding individual models, such as which organization provides them, which vendors support them, or which version of the model is deployed

- a common infrastructure model layer on which higher layer service models can be built, for example by specifying which models are needed to provide the service

- o an ability to express an instance of the structure consisting of models that have been validated to work together (i.e., with information about sources of the models, their versions, etc.), so that operators can easily identify a set of models that is known to be mutually consistent

Our approach is to organize the models describing various aspects of network infrastructure, including devices and their subsystems, and relevant protocols operating at the link and network layers. The proposal does not consider a common model for higher level network services, nor does it specify details of how hardware-related data should be organized. Both of these are challenging to standardize -- services are subject to operational and business considerations that vary across network operators, and hardware models are necessarily dependent on specific platform features and architecture -- and are thus out of scope of this document. We instead consider the set of models that are commonly used by network operators, and suggest a corresponding organization.

As with other models developed from an operator perspective, the intent is not to be exhaustive by including all possible models in the overall structure, whether currently available or not. We focus on components that are deemed most useful for network operators across a variety of use cases. We recognize, however, that additional models will be needed in some cases, and this structure is useful for describing how new models can be fit into the overall structure.

2. Model overview

The model organization can itself be thought of as a "meta- model", in that it describes the relationships between individual models. We choose to represent it also as simple YANG model consisting of lists and containers to serve as anchor points for the corresponding individual models.

As shown below, our model is rooted at a "device", which represents a network router, switch, or similar device. The model is applicable to both physical, hardware-based devices, as well as software-based devices such as virtual network functions (VNFs). It does not follow the hierarchy of any particular implementation, and hence is vendor-neutral. Nevertheless, the structure should be familiar to network operators and also readily mapped to vendor implementations.

```
+--rw device
      +--rw info
      |  +--rw device-type?
      |     ...
      +--rw hardware
      +--rw system
      |     ...
      +--rw interfaces
      |     ...
      +--rw acl
      +--rw qos
      +--rw logical-routers
            ...
```

The key subsystems are represented at the top level of the device, including, system-wide configuration, interfaces, and routing instances. The info section can be used for basic device information such as its type (e.g., physical or virtual), vendor, and model. For physical devices, the hardware container is intended to be a placeholder for platform-specific configuration and operational state data. For example, a common structure for the hardware model might include chassis, linecards, and ports, but we leave this unspecified.

2.1. System model components

The system container includes a number of subsystems that are typically configured globally for the device. Some of these, such as DHCP, Ethernet CFM, or sampling configuration also may have data that is associated with an interface. For simplicity, these relationships are not represented in this structural model. The currently defined subsystems are shown below:

```
+--rw device
   +--rw system
      +--rw dns
      +--rw ntp
      +--rw dhcp
      +--rw syslog
      +--rw ssh
      +--rw stat-coll
      +--rw oam
      |  +--rw snmp
      |  +--rw cfm
      |  +--rw twamp
      +--rw aaa
      |  +--rw tacacs
      |  +--rw radius
      +--rw users
```

2.2. Interface model components

Interfaces are a crucial part of any network device’s configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interfaces container includes a number of commonly used components as examples:

```
+--rw device
   +--rw interfaces
      +--rw ethernet
      |  +--rw aggregates
      |  +--rw vlans
      |  +--rw lfm
      +--rw sonet-sdh
      +--rw addressing
      |  +--rw ipv4
      |  |  +--rw vrrp
      |  +--rw ipv6
      |     +--rw vrrp
      +--rw tunnels
```

2.3. Logical routing instances

Logical routers represent the capability on some devices to partition resources into independent logical routers. In physical devices, some hardware features are shared across partitions, but routing protocol instances, routing tables, and configuration are managed separately. In virtual routers or VNFs, this may correspond to establishing multiple logical instances using a single software installation. The model supports configuration of multiple routing instances on a single device by creating a list of logical routers, each with their own configuration and operational state related to routing and switching protocols, as shown below:

```
+--rw device
   +--rw logical-routers
      +--rw logical-router* [router-id]
         +--rw router-id            uint8
         +--rw router-name?         string
         +--rw layer-2-protocols
         |     ...
         +--rw layer-3-protocols
               ...
```

2.4. VRFs and global routing configuration

Virtual routing and forwarding instances (VRFs) are commonly used to isolate routing domains, for example to create virtual private networks, each with their own active protocols and routing policies. Devices also have a global instance of each routing protocol that may also exchange routes with VRFs through routing policies. The model describes protocols and policies for both VRF routing instances and the global instance. The routing policy framework is expected to follow [RTG-POLICY], which enables import / export policies to be expressed with respect to a VRF, or the global routing instance.

```
+--rw device
      +--rw logical-routers
         +--rw logical-router* [router-id]
            +--rw router-id
            +--rw router-name?
            +--rw layer-3-protocols
               +--rw global
               |     ...
               +--rw vrf* [vrf-name]
               |     ...
               +--rw routing-policy
                     ...
```

3. Populating the structural model

The structural model in this document describes how individual YANG models may be used together to represent the configuration and operational state for all parts of a physical or virtual device. It does not, however, document the actual model in its entirety. In this section, we outline an option for creating the full model and also describe how it may be used.

3.1. Constructing the device model

One of the challenges in assembling existing YANG models is that they are generally written with the assumption that each model is at the root of the configuration or state tree. Combining models then results in a multi-rooted tree that does not follow any logical construction and makes it difficult to work with operationally. In some cases, models explicitly reference other models (e.g., via augmentation) to define a relationship, but this is the case for only a few existing models.

Some examples include the interfaces [RFC7223] and IP management [RFC7277] models, and proposed IS-IS [RTG-ISIS], OSPF [RTG-OSPF] and routing configuration [RTG-CFG] models.

3.2. Pull approach for model composition

To enable model composition, one possible approach is to avoid using root-level containers in individual component models. Instead, the top level container (and all other data definitions) can be enclosed in a YANG 'grouping' statement so that when the model is imported by another model, its location in the configuration tree can be controlled by the importing YANG module with the 'uses' statement. One advantage of this approach is that the importing module has the flexibility to readily use the data definitions where the author deems appropriate.

One obvious drawback is that individual models no longer contain any of their own data definitions and must be used by a higher-level model for their data nodes to become active. Some judgment as to which models are more suited for inclusion in higher level models is also necessary to decide when the corresponding YANG module should contain only groupings. Another potential drawback is that this approach does not define a common structure for models to fit together, limiting interoperability due to implementations using different structures. To address this, a top-level standard model structure could be defined and updated to import new models into the hierarchy as they are defined.

3.3. "Push" approach for model composition

An alternative approach is to develop a top level model which defines the overall structure of the models, similar to the structure described in Section 2. Individual models may augment the top level model with their data nodes in the appropriate locations. The drawback is the need for a pre-defined top level model structure. On the other hand, when this top level model is standardized, it can become the basis for a vendor-neutral way to manage devices, assuming that the component models are supported by a given implementation.

One question in both approaches is what the root of the top- level model should be. In this document we selected to base the mode at a device because this layer should be common across many use cases and implementations. Starting at a higher layer (e.g., services) makes defining and agreeing on a common organization more challenging as discussed in Section 1.1.

Ideally, one could consider a hybrid construction mechanism that supports both styles of model composition. For example, a YANG compiler directive could be used to indicate whether an individual model should assume it is at the root, or whether it is meant for inclusion in other higher-level models.

4. Additional use cases

The goal of this document is to motivate the need for an overall structure for YANG data models that allows all of the data to be accessed in a common, logical way. With such a structure defined itself as a simple YANG model, it is possible to consider additional use cases.

4.1. Model catalog

YANG data models are being developed in a number of organizations, including standards bodies such as IETF, ONF, and IEEE, as well as open source projects and ad-hoc working groups. In addition to understanding how these models can work together, another challenge for users is the complexity of tracking which organization created a given model, and the capabilities and coverage each model provides. This becomes even more difficult when multiple overlapping models are available for a particular component.

Such a catalog could also be locally defined by an operator to describe the models needed to instantiate and manage different services.

The idea of a model catalog is similar to service catalogs in traditional IT environments. Service catalogs serve as a software-based registries of available services with information needed to discover and invoke available services.

The current model structure described in Section 2 focuses on describing relationships between the models, however there are several examples of additional metadata that could be captured for each component model in the overall structural model:

- origin and responsible party for maintenance of the model with contact information. In IETF standard models, the YANG 'organization' and 'contact' statement contents are a good example, but this is not necessarily the case for models from other sources.
- license under which the model is distributed, e.g., open source or as part of a commercial license
- classification of the model, including its category / subcategory, whether the model is intended to be used standalone, etc.
- model dependencies, e.g., a list of other modules that are required

o namespace information, including base namespace, prefixes, etc. to enable importing the model

o pointer to the YANG code, if it is freely downloadable

o implementation information, for example, a list of available implementations that support the model from vendors, open source projects, etc.

o authentication information to allow users to verify that the model they download does in fact originate from the stated organization

For such an approach to be useful, we also require a registration system where model developers can register information about their models, and update it as needed. The IANA XML Registry" [RFC3688] provides a basic registry for YANG models, but the information is somewhat limited and is currently targeted at IETF-standardized models only. Further details on the proposal for such a registry may be forthcoming in further revisions to this document.

4.2. Service-layer composition

The proposed structural model covers a wide variety of components and protocols, and clearly not all of them are needed for all services. Another envisioned use case for the structural model is the ability to reference the set of models that are needed for specific use cases or services. The intent is that the set would be based on best operational practices as defined by users or operators who run such services.

One approach for this would be to define a 'service overlay' model, for example for Layer 3 VPN services, that defines the set of required configuration and state models, such as VRFs, interfaces, BGP, policy, ACLs, and QoS. Similar overlay models can be defined for other services or use cases, for example, basic Internet operations such as adding new peers or customers, or setting up Layer 2 VPNs. Note these overlay models may be complementary to actual configuration models for such services, which may focus on providing an abstracted set of configuration or operational state variables, which would then be mapped onto device level variables. We leave discussion of such mapping mechanisms to future revisions.

5. Security Considerations

The model structure described in this document does not define actual configuration and state data, hence it is not directly responsible for security risks.

However, each of the component models that provide the corresponding configuration and state data should be considered sensitive from a security standpoint since they generally manipulate aspects of network configurations. Each component model should be carefully evaluated to determine its security risks, along with mitigations to reduce such risks.

6. IANA Considerations

This YANG model currently uses a temporary ad-hoc namespace. If it is placed or redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The YANG structure modules will be registered in the "YANG Module Names" registry [RFC6020].

7. YANG module

The model structure is described by the YANG module below.

7.1. Model structure

```
<CODE BEGINS> file model-structure.yang
module model-structure {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/structure";

  prefix "struct";

  // import some basic types


  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
    netopenconfig@googlegroups.com";

  description
    "This module describes a model structure for YANG
    configuration and operational state data models. Its intent is to
    describe how individual device protocol and feature models fit
    together and interact.";

  revision "2015-03-06" {
```

```
      description
        "Initial revision";
      reference "TBD";
    }

    // extension statements

    // feature statements

    // identity statements

    // typedef statements

    // grouping statements

    grouping info {
      description
        "base system information";

      container info {
        description
          "This container is for base system information, including
          device type (e.g., physcal or virtual), model, serial no.,
          location, etc.";

        leaf device-type {
          //TODO: consider changing to an identity if finer grained
          // device type classification is envisioned
          type enumeration {
            enum PHYSICAL {
              description "physical or hardware device";
            }
            enum VIRTUAL {
              description "virtual or software device";
            }
          }
          description
            "Type of the device, e.g., physical or virtual.  This node
            may be used to activate other containers in the model";
        }

      }
    }

    grouping hardware {
      description
        "hardware / vendor -specific data relevant to the platform";
```

```
       container hardware {
         description
           "This container is an anchor point for platform-specific
           configuration and operational state data.  It may be further
           organized into chassis, linecards, ports, etc.  It is
           expected that vendor or platform-specific augmentations
           would be used to populate this part of the device model";
       }
     }

     grouping l2-protocol-members {
       description "containers for each layer 2 protocol model";

       container vsi {
         description "virtual switch instance (or virtual forwarding
         instance) for use in PWE3 / VPLS services";

       }

       container ipv6-ndp {
         description "IPv6 neighbor discovery";
         reference "RFC 4861 - Neighbor Discovery for IP version 6
         (IPv6)";
       }

       container arp {
         description "Address resolution protocol";
         reference "STD 37 - An Ethernet Address Resolution Protocol";
       }

       container rstp {
         description "rapid spanning tree protocol";
         reference "IEEE 802.1D-2004";
       }

       container lldp {
         description "link layer discovery protocol";
         reference "IEEE 802.1AB";
       }

       container ptp {
         description
           "precision time protocol for time synchronization services.
           PTP also typically requires per-interface configuration";
         reference "IEEE 1588-2008";
       }
     }
```

```
     grouping l2-protocols {
       description "Layer 2 protocol models";

       container layer-2-protocols {
         description "layer 2 protocols and features";

         uses l2-protocol-members;

       }

     }

     grouping igp-protocol-members {
       description "containers for IGPs";

       container is-is {
         description "IS-IS IGP routing protocol";
         reference "RFC 1195 - Use of OSI IS-IS for Routing in TCP/IP
         and Dual Environments";
       }

       container ospf {
         description "OSPF IGP routing protocols";

         container ospf2 {
           description "OSPF v2";
           reference "RFC 2328 - OSPF Version 2";
         }

         container ospf3 {
           description "OSPF v3";
           reference "RFC 5340 - OSPF for IPv6";
         }
       }

       container igp-common {
         description "Common parameters for IGP protocols";
       }
     }

     grouping l3-protocol-members-vrf {
       description "containers for layer 3 protocol that are supported
       in a VRF instance";

       container bgp {
         description "BGP 4";
         reference "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
       }
```

```
       container igp {
         description "interior gateway protocols";

         uses igp-protocol-members;
       }

       container bfd {
         description "bidirectional forwarding detection";
         reference "RFC 5880 - Bidirectional Forwarding Detection
         (BFD)";
       }

       container pim {
         description "protocol independent multicast";
         reference "RFC 4601 - Protocol Independent Multicast -
         Sparse Mode (PIM-SM): Protocol Specification (Revised)";
       }

       container igmp {
         description "Internet group management protocol";
         reference "RFC 3376 - Internet Group Management Protocol,
         Version 3";
       }

       container static-routes {
         description "static route that are manually created";
       }

     }

     grouping l3-protocols-misc {
       description "containers for other features operating at the
       network layer";

     }

     grouping l3-protocols-mpls {
       description "models related to MPLS and TE";

       container mpls-te {
         description "MPLS and traffic engineering";

         container global {
           description "global MPLS configuration";
         }

         container signaling {
           description "MPLS signaling protocols";
```

```
          container rsvp {
            description "RSVP signaling";
            reference "RFC 3209 - RSVP-TE: Extensions to RSVP for LSP
            Tunnels";
          }

          container segment-routing {
            description "SR signaling";
            reference "Segment Routing Architecture -
            draft-filsfils-spring-segment-routing-04";
          }

          container ldp {
            description "label distribution protocol";
            reference "RFC 5036 - LDP Specification";
          }
        }

        container label-switched-paths {
          description "models for different types of LSPs";

          container constrained-path {
            description "traffic-engineered, or constrained path LSPs";
          }

          container igp-congruent {
            description "LSPs that follow the IGP-computed path";
          }

          container static {
            description "statically configured LSPs";
          }
        }
      }
    }

    grouping l3-protocol-members {
      description "containers for all layer 3 protocols";

      uses l3-protocol-members-vrf;
      uses l3-protocols-misc;
      uses l3-protocols-mpls;

    }

    grouping l3-routing-policy {
      description "containers for routing policy models";
```

```
       container common {
         description "generic routing policy framework and
         configuration parameters";
       }

       container bgp-policy {
         description "BGP-specific routing policy parameters";
       }

       container igp-policy {
         description "IGP routing policy knobs -- may include
         policy parameters for specific IGPs";
       }

       container vrf-policy {
         description "import/export policies for VRFs";
       }
     }

     grouping l3-protocols {
       description "Layer 3 protocol models";

       container layer-3-protocols {
         description "layer 3 protocols and features";

         container global {
           description "router-wide instance of each routing protocol";

           uses l3-protocol-members;

         }

         list vrf {
           key vrf-name;
           description "list of VRF instances";

           leaf vrf-name {
             type string;
             description "name or id of the routing instance / VRF";
           }

           uses l3-protocol-members-vrf;
         }

         container routing-policy {
           description "models related to routing policy across
           protocols and VRFs";
```

```
           uses l3-routing-policy;

         }
       }
     }

     grouping interface-ip-common {
       description
         "interface-specific configuration for IP interfaces, IPv4 and
         IPv6";

       container vrrp {
         description "virtual router redundancy protocol";
         reference "RFC 5798 - Virtual Router Redundancy Protocol
         (VRRP) Version 3 for IPv4 and IPv6";
       }
     }

     grouping interface-addr-families {
       description
         "containers for addr family-specific data attached
         to interfaces";

       container ipv4 {
         description "IPv4 interfaces";

         uses interface-ip-common;
       }

       container ipv6 {
         description "IPv6 interfaces";

         uses interface-ip-common;
       }
     }

     grouping interfaces {
       description "interface-related models";

       container interfaces {
         description "various interface models";

         container ethernet {
           description "Ethernet interface config, e.g., 10, 40,
           100GBE";

           container aggregates {
             description "LAGs, LACP, etc. for Ethernet interfaces";
```

```
            reference "IEEE 802.1ad, 802.1AX";
          }

          container vlans {
            description "VLANs, 802.1q, q-in-q, etc.";
            reference "IEEE 802.1Q";
          }

          container lfm {
            description
              "Link-layer fault management for Ethernet interfaces";
            reference "IEEE 802.3ah";
          }
        }

        container sonet-sdh {
          description "SONET/SDH interfaces";
          reference
            "SDH: ITU standards G.707, G.783, G.784, and G.803
            SONET: ANSI standard T1.105";

        }

        container addressing {
          description "addressing and other interface-specific data,
          e.g., data plane protocols";

          uses interface-addr-families;
        }

        container tunnels {
          description
            "logical tunnel interfaces incl. GRE, VxLAN, L2TP etc.";
        }

      }
    }

    grouping oam {
      description "containers for features related to operations,
      administration, and management";

      container oam {
        description "commonly use OAM functions on devices";

        container snmp {
          description "SNMP server information, e.g., allowed clients";
        }
```

```
        container cfm {
          description
            "Ethernet connectivity fault management.  Also includes
            options that are associated with specific interfaces, such
            as maintenance endpoint domains.";
          reference "IEEE 802.1ag";
        }

        container twamp {
          description
            "Two-way active measurement protocol for measuring
            round-trip IP layer performance.";
          reference "RFC 5357 A Two-Way Active Measurement Protocol
          (TWAMP)";
        }
      }
    }

    grouping system-services {
      description "containers for system service models";

      container dns {
        description "domain name service and resolver configurration";
      }

      container ntp {
        description "network time protocol configuration";
      }

      container dhcp {
        description "dhcp and relay services";
      }

      container syslog {
        description "syslog configuration";
      }

      container ssh {
        description "ssh server configuration";
      }

      container stat-coll {
        description
          "mechanisms for data collection from devices, including
          packet and flow-level sampling";
      }

      uses oam;
```

```
     }

     grouping system-aaa {
       description "AAA-related services";

       container aaa {
         description "authentication, authorization, and accounting";

         container tacacs {
           description "TACACS+ configuration";
         }

         container radius {
           description "RADIUS";
           reference "RFC 2865 - Remote Authentication Dial In User
           Service (RADIUS)";
         }
       }
     }

     grouping system {
       description "system-wide services";

       container system {
         description "system services";

         uses system-services;
         uses system-aaa;

         container users {
           description "local user configuration";
         }
       }
     }

     grouping acl {
       description "forwarding rules";

       container acl {
         description "ACLs and packet forwarding rules";
       }
     }

     grouping qos {
       description "QoS features";

       container qos {
         description "QoS, including policing, shaping, etc.";
```

```
    }
  }

  // data definition statements

  container device {
    description "top-level anchor point for models.  Device is a
    generic L2/L3 network element";

    uses info;
    uses hardware;
    uses system;
    uses interfaces;
    uses acl;
    uses qos;

    container logical-routers {
      description "devices may support multiple logical router
      instances";

      list logical-router {

        key router-id;
        description "list of logical router instances";

        leaf router-id {
          type uint8; // expect a small number of logical routers
          description "identifier of the logical router instance";
        }

        leaf router-name {
          type string; // expect a small number of logical routers
          description "identifier of the logical router instance";
        }

        uses l2-protocols;
        uses l3-protocols;

      }
    }

  }

  // augment statements

  // rpc statements

  // notification statements
```

```
   }
   <CODE ENDS>
```

8. References

8.1. Normative references

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2014.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, June 2014.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, August 2014.

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

8.2. Informative references

[RTG-CFG] Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-16 (work in progress), October 2014.

[RTG-POLICY] Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-shaikh-rtgwg-policy-model-00 (work in progress), January 2015.

[RTG-OSPF] Yeung, D., Qu, Y., Zhang, J., and D. Bogdanovic, "Yang Data Model for OSPF Protocol", draft-yeung-netmod-ospf-02 (work in progress), October 2014.

[RTG-ISIS] Litkowski, S., Yeung, D., Lindem, A., Zhang, J., and L. Lhotka, "YANG Data Model for ISIS protocol", draft-ietf-isis-yang-isis-cfg-01 (work in progress), October 2014.

Appendix A. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Deepak Bansal, Paul Borman, Chris Chase, Josh George, Marcus Hines, and Jim Uttaro.

Authors' Addresses

Anees Shaikh
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: aashaikh@google.com

Rob Shakir
BT
pp. C3L, BT Centre
81, Newgate Street
London EC1A 7AJ
UK

Email: rob.shakir@bt.com
URI: http://www.bt.com/

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
US

Email: kd6913@att.com

Luyuan Fang
Microsoft
205 108th Ave. NE, Suite 400
Bellevue, WA
US

Email: lufang@microsoft.com