Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 29, 2015

A. Shaikh
Google
K. D'Souza
AT&T
D. Bansal
Microsoft
R. Shakir
BT

October 26, 2014

BGP Configuration Model for Service Provider Networks
draft-shaikh-idr-bgp-model-00

Abstract

This document defines a YANG data model for configuring and managing BGP, including protocol, policy, and operational aspects based on carrier and content provider operational requirements.

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 April 29, 2015.

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

1. Introduction

This document describes a YANG [RFC6020] data model for BGP [RFC4271] protocol and policy configuration, as well as defining key operational state data. The model is intended to be vendor-neutral, in order to allow operators to manage BGP configuration in heterogeneous environments with routers supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations, however, to facilitate support from as large a set of routing hardware and software vendors as possible.

1.1. Goals and approach

This model does not (in the current iteration) aim to be feature complete (i.e., cover all possible features of a BGP implementation). Rather its development is driven by examination of BGP configurations in use across a number of operator network deployments.

The focus area of the first version of the model is on base BGP protocol configuration and policy configuration with "hooks" to add support for additional address families, as well as operational data to enable a common model for reading BGP-related state from devices.

Configuration items that are deemed to be widely available in existing major BGP implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in companion modules that augment the current model. This allows clarity in identifying data that is part of the vendor-neutral model.

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation. Since implementations vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

2. Model overview

The BGP model is defined across several YANG modules but at a high level is organized into three main parts:

- protocol configuration -- configuration affecting BGP protocol-related operations, defined at various levels of hierarchy.
- policy configuration -- configuration defining the policies that act on routes sent (received) to (from) peers or other routing protocols.
- operational state -- variables used for monitoring, management, etc. of BGP operations.

These modules also make use of the standard Internet types, such as IP addresses, autonomous system numbers, etc., defined in RFC 6991 [RFC6991].

2.1. BGP protocol configuration

The BGP protocol configuration model is organized hierarchically, much like the majority of router implementations. That is, configuration items can be specified at multiple levels, as shown below.

```
+--rw bgp
   +--rw global
   +--rw afi* [afi-name]
   +--rw peer-group* [group-name]
   +--rw neighbor* [neighbor-address]
   +--rw policy
```

Users may specify configuration at a higher level and have it apply to all lower-level items, or provide overriding configuration at a lower level of the hierarchy. Overriding configuration items are optional.

The model makes the simplifying assumption that most of the configuration items are available at all levels of the hierarchy. That is, very little configuration is specific to a particular level in the hierarchy, other than obvious items such as "group- name" only being available for the peer group-level config. A notable exception is for sub-address family configuration where some items are only applicable for a given AFI-SAFI combination.

The initial version of the model includes placeholders (i.e., mount points) for configuring different address family or sub address

family (AFI/SAFI) combinations in support of BGP multiprotocol extensions [RFC4760]. The focus, however, is on base IPv4 and IPv6 configuration, with later versions of the model providing additional configuration for specific AFI/SAFI types. The current scope of AFI/SAFI combinations is indicated below. Each of the subtrees are containers for configuration specific to the corresponding AFI-SAFI combination.

```
+--rw bgp
   +--rw afi* [afi-name]
      +--rw safi* [safi-name]
         +--rw safi-name              identityref
         +--rw ipv4-ipv6-unicast
         +--rw ipv4-l3vpn-unicast
         +--rw ipv6-l3vpn-unicast
         +--rw ipv4-labeled-unicast
         +--rw l2vpn
         +--rw ipv4-multicast-vpn
         +--rw ipv6-multicast-vpn
         +--rw prefix-limit
         +--rw apply-policy
```

2.2. Policy configuration overview

The BGP policy configuration model is based on a condition-action model in which policies are expressed as a series of conditions, each with a corresponding action. Conditions include testing for membership in a predefined set (e.g., community attribute matches a predefined set of communities), or testing route attributes for specified values. Actions support setting specific attributes on a route or performing a control flow operation such as accepting or rejecting a route, or going to the next policy.

The general structure of policy definitions, policy statements, and condition-action policy rules is shown below.

```
+--rw bgp
   +--rw policy
      +--rw policy-definitions!
         +--rw policy-definition* [name]
            +--rw name          string
            +--rw statements* [name]
               +--rw name          string
               +--rw conditions
               +--rw actions
```

The policy model supports policy subroutines through the use of the condition statement call-policy which applies the conditions from

another named policy definition. It also supports policy chaining using the action statements goto-next and goto-policy, which allow policy evaluation to immediately move to the next policy statement in the current policy definition, or move to another named policy definition, respectively. Using multiple policy statements in a policy definition is also a form of policy chaining which will evaluate each policy statement in turn.

2.3. Operational data overview

The BGP operational data model contains an initial set of state variable definitions that would be required in most service provider deployments for management, monitoring, and fault detection.

3. Security Considerations

BGP configuration has a significant impact on network operations, and as such any related protocol or model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer BGP configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

4. IANA Considerations

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

5. YANG modules

The modules comprising the BGP configuration and operational model are described by the YANG modules in the sections below. The base module imports the other modules to create the overall model.

5.1. BGP base model

```
<CODE BEGINS> file bgp.yang
module bgp {

  yang-version "1";

  // namespace
  // TODO: change to an ietf or other more generic namespace
  namespace "http://google.com/yang/google-bgp-protocol-cfg";

  prefix "bgp";

  // import some basic inet types
  import ietf-inet-types { prefix inet; }
  import bgp-multiprotocol { prefix bgp-mp; }
  import bgp-policy { prefix bgp-pol; }
  import bgp-operational { prefix bgp-op; }


  // meta
  organization
    "Google, AT&T, BT, Microsoft";

  contact
    "Google, Inc.
    1600 Amphitheatre Way
    Mountain View, CA  94043

    AT&T Labs
    200 S. Laurel Avenue
    Middletown, NJ  07748

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

    Microsoft
    205 108th Ave. NE, Suite 400
    Bellevue, WA 98004";

  description
    "This module describes a YANG model for BGP protocol
    configuration.It is a limited subset of all of the configuration
    parameters available in the variety of vendor implementations,
    hence it is expected that it would be augmented with vendor-
```

```
       specific configuration data as needed.Additional modules or
       submodules to handle other aspects of BGP configuration,
       including policy, VRFs, VPNs, and additional address families
       are also expected.

       This model supports the following BGP configuration level
       hierarchy:

         BGP
           |
           +-> [ global BGP configuration ]
           +-> AFI / SAFI (address family)
             +-> [AFI-specific config ]
           +-> peer group
             +-> [ peer group config ]
             +-> AFI / SAFI [ per-AFI overrides ]
             +-> neighbor
               +-> [ per-neighbor overrides ]
               +-> AFI / SAFI [ per-AFI overrides ]
           +-> neighbor
             +-> [ neighbor config ]
             +-> AFI / SAFI [ per-AFI overrides ]";


    revision "2014-09-30" {
        description
          "Initial revision";
        reference "TBD";
    }


    typedef peer-type {
      type enumeration {
        enum INTERNAL {
          description "internal (iBGP) peer";
        }
        enum EXTERNAL {
          description "external (eBGP) peer";
        }
      }
      description
        "labels a peer or peer group as explicitly internal or
        external";
    }


    typedef remove-private-as-option {
      type enumeration {
```

```
        enum ALL {
          description "remove all private ASes in the path";
        }
        enum REPLACE {
          description "replace private ASes with local AS";
        }
      }
      description
        "set of options for configuring how private AS path numbers
        are removed from advertisements";
    }

    typedef percentage {
      type uint8 {
        range "0..100";
      }
      description
        "Integer indicating a percentage value";
    }

    typedef rr-cluster-id-type {
      type union {
        type uint32;
        type inet:ipv4-address;
      }
      description
        "union type for route reflector cluster ids:
        option 1: 4-byte number
        option 2: IP address";
    }

    grouping bgp-common-configuration {
      description "Common configuration available at all hierarchy
      levels, global, AFI, groups, neighbors, etc.";

      leaf description {
        type string;
        description
          "An optional textual description (intended primarily for use
          with a peer or group";
      }

      container route-selection-options {
          // TODO: consider moving this container to AFI/SAFI level
          // config
          description
            "Set of configuration options that govern best
            path selection.";
```

```
        leaf always-compare-med {
          type boolean;
          default "false";
          description
            "Compare multi-exit discriminator (MED) value from
            different ASes when selecting the best route.  The
            default behavior is to only compare MEDs for paths
            received from the same AS.";
        }

        leaf ignore-as-path-length {
          type boolean;
          default "false";
          description
            "Ignore the AS path length when selecting the best path.
            The default is to use the AS path length and prefer paths
            with shorter length.";
        }

        leaf external-compare-router-id {
          type boolean;
          default "true";
          description
            "When comparing similar routes received from external
            BGP peers, use the router-id as a criterion to select
            the active path.";
        }

        leaf advertise-inactive-routes {
          type boolean;
          default "false";
          description
            "Advertise inactive routes to external peers.  The
            default is to only advertise active routes.";
        }

        leaf enable-aigp {
          type empty;
          description
            "Flag to enable sending / receiving accumulated IGP
            attribute in routing updates";
        }
      }

      container use-multiple-paths {

        presence
          "Presence of this container indicates that multipath
```

```
            is enabled for both eBGP and iBGP, absence indicates
            that multi-path is not used";

          description
            "Configuration of BGP multi-path for iBGP and eBGP";

          container ebgp {
            description
              "Configuration of BGP multipath to enable load sharing
              across multiple paths to eBGP peers";

            leaf allow-multiple-as {
              type boolean;
              default "false";
              description
               "Allow multipath to use paths from different neighbouring
               ASes.  The default is to only consider multiple paths from
               the same neighbouring AS.";
            }

            leaf maximum-paths {
              type uint32;
              default 1;
              description
               "Maximum number of parallel paths to consider when using
               BGP multipath. The default is use a single path.";
            }
          }

          container ibgp {
            description
              "Configuration of BGP multipath to enable load-sharing
              across multiple paths to iBGP peers";

            leaf maximum-paths {
              type uint32;
              default 1;
              description
                "Maximum number of parallel paths to consider when using
                iBGP multipath. The default is to use a single path";
            }
          }

          container eibgp {
            description
              "Configuration of BGP multipath to enable load-sharing
              across multiple paths to external confederation sub-ASes";
```

```
        leaf maximum-paths {
          type uint32;
          default 1;
          description
            "Maximum number of parallel paths to consider when using
            eiBGP multipath. The default is to use a single path";
        }

      }
    }


    container graceful-restart {
      // TODO: most impls seem to require this at the global level
      // in order to specify at neighbor or other levels
      presence "Presence of this item indicates that BGP graceful
      restart is enabled.";

      description
        "Configures BGP graceful restart, which is a negotiated
        option that indicates that a BGP speaker is able to retain
        forwarding state when a BGP session restarts";

      reference "RFC 4724: Graceful Restart Mechanism for BGP";

      leaf restart-time {
        type uint16 {
          range 0..4096;
        }
        description
          "Estimated time in seconds for the BGP session to be
          re-established after a restart.  This is a 12-bit value
          advertised by the router to peers.  Per RFC 4724, the
          suggested default value is <= the hold-time value";
      }

      leaf stale-routes-time {
        type decimal64 {
          fraction-digits 2;
        }
        description
          "Sets an upper bound on the time in seconds that stale
          routes will be retained by the router after a session is
          restarted";
      }
    }

    uses bgp-pol:apply-policy-group;
```

```
    }

    grouping  bgp-global-configuration {
        description
          "Grouping for global level configuration items";

        leaf as {
          type inet:as-number;
          mandatory "true";
          description
            "Local autonomous system number of the router.  Uses
            the 32-bit as-number type from the model in RFC 6991";
        }
        leaf router-id {
          type inet:ipv4-address;
          description
            "Router id of the router, expressed as an
            32-bit value, IPv4 address.";
        }

        container default-route-distance {
          description
            "Administrative distance (or preference) assigned to
            routes received from different sources
            (external, internal, and local).";
          leaf external-route-distance {
            type uint8 {
              range "1..255";
            }
            description
              "Administrative distance for routes learned from external
              BGP (eBGP).";
          }
          leaf internal-route-distance {
            type uint8 {
              range "1..255";
            }
            description
              "Administrative distance for routes learned from internal
              BGP (iBGP).";
          }
        }

        container confederation {

          presence "Presence of this node indicates that the local AS
            is part of a confederation";
```

```
          description
            "Configuration for a BGP confederation consisting of a
            confed id and member sub-AS list";

          leaf identifier {
            type inet:as-number;
            description
              "Confederation identifier for the autonomous system";
          }

          leaf-list member-as {
            type inet:as-number;
            description
              "Remote autonomous systems that are to be treated
              as part of the local confederation.";
          }
        }

    }

    grouping bgp-group-common-configuration {
      description "Configuration items that are applied at the peer
      group level";

      // currently a placeholder in case we identify config that is
      // really only applicable at the group level
    }

    grouping bgp-group-neighbor-common-configuration {
      description "Configuration items that are applied at the peer
      or peer group levels";

      leaf auth-password {
        type string;
        description
          "Configures an MD5 authentication password for use with
          neighboring devices.";
      }

      leaf peer-type {
          type peer-type;
          description
            "Explicitly designate the peer or peer group as internal
            (iBGP) or external (eBGP).";
      }

      container timers {
        description "Configuration of various BGP timers";
```

```
        leaf connect-retry {
          type decimal64 {
            fraction-digits 2;
          }
          default 30;
          description
            "Time interval in seconds between attempts to establish a
            session with the peer.";
        }

        leaf hold-time {
          type decimal64 {
            fraction-digits 2;
          }
          default 90;
          description
            "Time interval in seconds that a BGP session will be
            considered active in the absence of keepalive or other
            messages from the peer.  The hold-time is typically
            set to 3x the keepalive-interval.";
          reference
            "RFC 4271 - A Border Gateway Protocol 4, Sec. 10";
        }

        leaf keepalive-interval {
          type decimal64 {
            fraction-digits 2;
          }
          default 30;
          description
            "Time interval in seconds between transmission of keepalive
            messages to the neighbor.  Typically set to 1/3 the
            hold-time.";
        }

        leaf minimum-advertisement-interval {
          type decimal64 {
            fraction-digits 2;
          }
          default 30;
          description
            "Mininum time interval in seconds between transmission
            of BGP updates to neighbors";
          reference
            "RFC 4271 - A Border Gateway Protocol 4, Sec 10";
        }

        leaf send-update-delay {
```

```
        type decimal64 {
          fraction-digits 2;
        }
        description
          "Time interval between routes changing in the routing
          table and corresponding updates sent to neighbors --
          serves to batch updates";
      }

    }

    container ebgp-multihop {
      description
        "Configure multihop BGP for peers that are not directly
        connected";

      leaf multihop-ttl {
        type uint8;
        default 1;
        description
          "Time-to-live for multihop BGP sessions.  The default
          value of 1 is for directly connected peers (i.e.,
          multihop disabled";

      }

    }

    container route-reflector {
      description
        "Configure the local router as a route-reflector
        server";

      leaf route-reflector-cluster-id {
        type rr-cluster-id-type;
        description
          "route-reflector cluster id to use when local router is
          configured as a route reflector.  Commonly set at the group
          level, but allows a different cluster
          id to be set for each neighbor.";
      }

      leaf route-reflector-client {
        type boolean;
        default "false";
        description
          "Configure the neighbor as a route reflector client.";
      }
```

```
      }

      leaf remove-private-as {
        // could also make this a container with a flag to enable
        // remove-private and separate option.  here, option implies
        // remove-private is enabled.
        type remove-private-as-option;
        description
          "Remove private AS numbers from updates sent to peers.";
      }

      container bgp-logging-options {
        description
          "Configure various tracing/logging options for BGP peers
          or groups.  Expected that additional vendor-specific log
          options would augment this container.";

        leaf log-neighbor-state-changes {
          type boolean;
          default "true";
          description
            "Configure logging of peer state changes.  Default is
            to enable logging of peer state changes.";
        }
      }

      container transport-options {
        description
          "Transport protocol options for BGP sessions";

          leaf tcp-mss {
            type uint16;
            description
              "Sets the max segment size for BGP TCP sessions.";
          }

          leaf mtu-discovery {
            type boolean;
            description
              "Turns path mtu discovery for BGP TCP sessions on (true)
              or off (false)";
          }

          leaf passive-mode {
            type boolean;
            description
              "Wait for peers to issue requests to open a BGP session,
              rather than initiating sessions from the local router.";
```

```
          }
      }

      leaf local-address {
        type inet:ip-address;
        description
          "Set the local IP (either IPv4 or IPv6) address to use for
          the session when sending BGP update messages.";
      }

      leaf route-flap-damping {
        type boolean;
        description
          "Enable route flap damping.";
      }
    }

    grouping bgp-neighbor-configuration {
      description
        "Neighbor-level configuration items";

      list neighbor {
        key "neighbor-address";
        description
          "List of BGP peers, uniquely identified by neighbor
          address.";
        leaf neighbor-address {
          type inet:ip-address;
          description
            "Address of the BGP peer, either IPv4 or IPv6.";
        }

        leaf peer-as {
          type inet:as-number;
          mandatory "true";
          description
            "AS number of the peer.";

        }
        uses bgp-common-configuration;
        uses bgp-mp:address-family-configuration;
        uses bgp-group-neighbor-common-configuration;
        uses bgp-op:bgp-op-neighbor-group;
      }
    }


    container bgp {
```

```
      description "Top-level configuration data for the BGP router";

      container global {
        description
          "Top-level bgp protocol options applied at the global level
          in the hierarchy -- these apply across peer-groups,
          neighbors, and address families";

        uses bgp-global-configuration;

        // attach global level operational data
        uses bgp-op:bgp-op-global-group;
      }

      // top level AF configuration
      uses bgp-mp:address-family-configuration;

      list peer-group {
        key "group-name";
        description
          "List of peer-groups, uniquely identified by the peer group
          name.";
        leaf group-name {
          type string;
          description "Name of the peer group.";
        }
        uses bgp-op:bgp-op-peergroup-group;
        uses bgp-common-configuration;
        uses bgp-mp:address-family-configuration;
        uses bgp-group-neighbor-common-configuration;

        // list of configurations for neighbors in this peer group
        uses bgp-neighbor-configuration;
      }

      // top level neighbor configuration
      uses bgp-neighbor-configuration;

      // hook for top-level policy definitions
      uses bgp-pol:policy-definition-group;
    }
  }
  <CODE ENDS>
```

5.2. BGP policy model

```
<CODE BEGINS> file bgp-policy.yang
module bgp-policy {

  yang-version "1";

  // namespace
  // TODO: change to an ietf or other generic namespace
  namespace "http://google.com/yang/google-bgp-policy-cfg";

  prefix "bgp-policy";

  // import some basic types
  import ietf-inet-types { prefix inet; }


  // meta
  // TODO: add collaborating organizations
  organization
    "Google, AT&T, BT, Microsoft";

  contact
    "Google, Inc.
    1600 Amphitheatre Way
    Mountain View, CA  94043

    AT&T Labs
    200 S. Laurel Avenue
    Middletown, NJ  07748

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

    Microsoft
    205 108th Ave. NE, Suite 400
    Bellevue, WA 98004";

  description
    "This module describes a YANG model for BGP policy
    configuration. It is a limited subset of all of the policy
    configuration parameters available in the variety of vendor
    implementations, but supports widely used constructs for managing
    how BGP routes are imported, exported, and modified.  This module
    works with the base BGP protocol configuration model defined in
```

```
      google-bgp.

      Route policy expression:

      Policies are expressed as a set of top-level policy definitions,
      each of which consists of a sequence of policy statements.
      Policy statements are simple condition-action tuples.  Conditions
      may include mutiple match or comparison operations, and similarly
      actions may be multitude of changes to route attributes and a
      final disposition of accepting or> rejecting the route.

      BGP
        |
        +->policy
          +-> policy definitions
            +-> policy statements
              +-> conditions
                +-> [ match conditions / comparison conditions ]
              +-> actions
                +-> [ set attribute actions / control-flow actions ]


      Route policy evaluation:

      Evaluation of a policy definition is expected to proceed by
      evaluating the individual policy statements in the specified
      order.  When a condition statement in a policy statement is
      satisfied, the corresponding action statement is executed.
      If the action statement has either accept-route or reject-route
      actions, policy evaluation stops.  If the condition is not
      satisfied, or if the action statement contains goto-next, then
      evaluation proceeds to the next policy statement.  If none of the
      policy statement conditions are satisfied, then the default
      action is applied.

      Policy 'subroutines' are supported by allowing condition
      statements to reference another policy definition which first
      applies conditions from the referenced policy before
      proceeding.";

    revision "2014-09-30" {
      description
        "Initial revision";
      reference "TBD";
    }

    // extension statements
```

```
    // feature statements

    // identity statements

    identity bgp-attribute-comparison {
      description
        "base type for supported comparison operators on route
        attributes";
    }

    identity attribute-eq {
      base bgp-attribute-comparison;
      description "== comparison";
    }

    identity attribute-ge {
      base bgp-attribute-comparison;
      description ">= comparison";
    }

    identity attribute-le {
      base bgp-attribute-comparison;
      description "<= comparison";
    }


    // typedef statements

    typedef match-set-options-type {
      type enumeration {
        enum ANY {
          description "match is true if given value matches any member
          of the defined set";
        }
        enum ALL {
          description "match is true if each given value matches a
          member of the defined set";
        }
        enum INVERT {
          description "match is true if given value does not match any
          member of the defined set";
        }
      }
      default ANY;
      description
        "Options that govern the behavior of a match statement.  The
        default behavior is ANY, i.e., the given value matches any
        of the members of the defined set";
```

```
    }

    typedef as-path-prepend-option-repeat {
      type uint32;
      description
        "Option for the as-prepend policy action.  Prepends the local
        AS number repeated n times";
    }

    typedef well-known-community-attr {
      type enumeration {
        enum INTERNET {
          description "entire Internet community (0x00000000)";
        }
        enum NO_EXPORT {
          // value 0xFFFFFF01;
          description "no export";
        }
        enum NO_ADVERTISE {
          description "no advertise (0xFFFFFF02)";
        }
        enum NO_EXPORT_SUBCONFED {
          description "no export subconfed, equivalent to
          local AS (0xFFFFFF03)";
        }
      }
      description
        "Type definition for well-known IETF community attribute
        values";
      reference "RFC 1997 - BGP Communities Attribute";
    }


    typedef std-community-attr-type {
      // TODO: further refine restrictions and allowed patterns
      // 4-octet value:
      //  <as number> 2 octets
      //  <community value> 2 octets
      type union {
        type uint32 {
        // per RFC 1997, 0x00000000 - 0x0000FFFF and 0xFFFF0000 -
        // 0xFFFFFFFF are reserved
          range "65536..4294901759"; // 0x00010000..0xFFFEFFFF
        }
        type string {
          pattern '([0-9]+:[0-9]+)';
        }
      }
```

```
      description
        "Type definition for standard commmunity attributes";
      reference "RFC 1997 - BGP Communities Attribute";
    }

    typedef ext-community-attr-type {
      // TODO: needs more work to make this more precise given the
      // variability of extended community attribute specifications
      // 8-octet value:
      //  <type> 2 octects
      //  <value> 6 octets
      type string {
        pattern '([0-9\.]+(:[0-9]+)?:[0-9]+)';
      }
      description
        "Type definition for extended community attributes";
      reference "RFC 4360 - BGP Extended Communities Attribute";
    }

    typedef community-regexp-type {
      // TODO: needs more work to decide what format these regexps can
      // take.
      type string;
      description
        "Type definition for communities specified as regular
        expression patterns";
    }

    typedef bgp-origin-attr-type {
      type enumeration {
        enum IGP {
          value 0;
          description "Origin of the NLRI is internal";
        }
        enum EGP {
          value 1;
          description "Origin of the NLRI is EGP";
        }
        enum INCOMPLETE {
          value 2;
          description "Origin of the NLRI is neither IGP or EGP";
        }
      }
      description
        "Type definition for standard BGP origin attribute";
      reference "RFC 4271 - A Border Gateway Protocol 4 (BGP-4),
        Sec 4.3";
    }
```

```
    typedef set-community-option-type {
      type enumeration {
        enum ADD {
          description "add the specified communities to the existing
          community attribute";
        }
        enum REMOVE {
          description "remove the specified communities from the
          existing community attribute";
        }
        enum REPLACE {
          description "replace the existing community attribute with
          the specified communities";
        }
        enum NULL {
          description "set the community attribute to empty / NULL";
        }
      }
      description
        "Type definition for options when setting the community
        attribute in a policy action";
    }

    typedef bgp-next-hop-type {
      type union {
        type inet:ip-address;
        type enumeration {
          enum SELF {
            description "special designation for local router's own
            address";
          }
        }
      }
      description "type definition for specifying next-hop in policy
      actions";
    }


    // grouping statements

    grouping defined-sets-definitions {
      description
        "Data definitions for pre-defined sets of attributes used in
        policy match conditions";

      list prefix-set {
        key prefix-set-name;
        description
```

```
          "Definitions for prefix sets";

        leaf prefix-set-name {
          type string;
          description
            "name / label of the prefix set -- this is used to
            reference the set in match conditions";
        }

        list prefix {
          key "address masklength masklength-range";
          description
            "list of prefix expressions that are part of the set";

          leaf address {
            type inet:ip-address;
            mandatory true;
            description
              "address portion of the prefix";
          }

          leaf masklength {
            type uint8 {
              // simple range covers both ipv4 and ipv6 --
              // could separate this into different types
              // for IPv4 and IPv6 prefixes
              range 1..128;
            }
            mandatory true;
            description
              "masklength for the prefix specification";
          }

          leaf masklength-range {
            type string {
              // pattern modeled after ietf-inet-types
              pattern '(([0-9])|([1-9][0-9])|(1[0-1][0-9])|'
                + '(12[0-8]))\.\.'
                + '(([0-9])|([1-9][0-9])|(1[0-1][0-9])|'
                + '(12[0-8]))';
            }
            description
              "Defines an optional range for the masklength.  Absence
              of the masklength-length implies that the prefix has an
              exact masklength given by the masklength parameter.
              Example: 10.3.192.0/21 through 10.3.192.0/24 would be
              expressed as address: 10.3.192.0, masklength: 21,
              masklength-range: 21..24";
```

```
      }
    }
  }

  list community-set {
    key community-set-name;
    description
      "Definitions for community sets";

    leaf community-set-name {
      type string;
      mandatory true;
      description
        "name / label of the community set -- this is used to
        reference the set in match conditions";
    }

    leaf-list community-members {
      type union {
        type std-community-attr-type;
        type community-regexp-type;
        type well-known-community-attr;
      }
      description
        "members of the community set";
    }

  }

  list ext-community-set {
    key ext-community-set-name;
    description
      "Definitions for extended community sets";

    leaf ext-community-set-name {
      type string;
      description
        "name / label of the extended community set -- this is used
        to reference the set in match conditions";
    }

    leaf-list ext-community-members {
      type union {
        type ext-community-attr-type;
        // TODO: is regexp support needed for extended communities?
        // TODO: is well-known needed for extended communities?
        type community-regexp-type;
      }
```

```
          description
            "members of the extended community set";
        }
      }

      list as-path-set {
        key as-path-set-name;
        description
          "Definitions for AS path sets";

        leaf as-path-set-name {
          type string;
          description
            "name of the AS path set -- this is used to reference the
            the set in match conditions";
        }

        leaf-list as-path-set-members {
          // TODO: need to refine typedef for AS path expressions
          type string;
          description
            "AS path expression -- list of ASes in the set";
        }

      }
    }

    grouping condition-set-matches {
      description
        "Condition statement definitions for checking membership in a
        defined set";

      leaf match-community-set {
        type leafref {
          path "/policy/defined-sets/community-set/community-set-name";
          require-instance true;
        }
        description
          "References a defined community set";
      }

      leaf match-ext-community-set {
        type leafref {
          path "/policy/defined-sets/ext-community-set"
            + "/ext-community-set-name";
        }
        description "References a defined extended community set";
      }
```

```
      leaf match-as-path-set {
        type leafref {
          path "/policy/defined-sets/as-path-set/as-path-set-name";
        }
        description "References a defined AS path set";
      }

      leaf match-prefix-set {
        type leafref {
          path "/policy/defined-sets/prefix-set/prefix-set-name";
        }
        description "References a defined prefix set";
      }

      leaf match-set-options {
        type match-set-options-type;
        description
          "Optional parameter that governs the behavior of the match
          operation";
      }
    }

    grouping condition-attribute-compare-operators {
      description "common definitions for comparison operations in
      condition statements";

      leaf operator {
          type identityref {
            base bgp-attribute-comparison;
          }
          description
            "type of comparison to be performed";
      }

      leaf value {
        type uint32;
        description
          "value to compare with the community count";
      }
    }

    grouping condition-attribute-comparisons {
      description
        "Condition statement definitions for comparing a route
        attribute to a specified value";

      leaf med-eq {
        type uint32;
```

```
        description
          "Condition to check if the received MED value is equal to
          the specified value";
      }

      leaf origin-eq {
        type bgp-origin-attr-type;
        description
          "Condition to check if the route origin is equal to the
          specified value";
      }

      leaf-list next-hop-in {
        type inet:ip-address;
        description
          "List of next hop addresses to check for in the route
          update";
      }

      leaf local-pref-eq {
        type uint32;
        // TODO: add support for other comparisons
        description
          "Condition to check if the local pref attribute is equal to
          the specified value";
      }

      container community-count {

        presence "node is present in the config data to indicate a
        community-count condition";

        description
          "Value and comparison operations for conditions based on the
          number of communities in the route update";

        uses condition-attribute-compare-operators;

      }

      container as-path-length {

        presence "node is present in the config data to indicate a
        as-path-length condition";

        description
          "Value and comparison operations for conditions based on the
          length of the AS path in the route update";
```

```
        uses condition-attribute-compare-operators;
      }

      leaf route-type {
        // TODO: verify extent of vendor support for this comparison
        type enumeration {
          enum INTERNAL {
            description "route type is internal";
          }
          enum EXTERNAL {
            description "route type is external";
          }
        }
        description
          "Condition to check the route type in the route update";
      }
    }

    grouping set-attribute-actions {
      description
        "Definitions for base set of policy action statements that
        change various attributes of the route";

      container set-as-path-prepend {

        presence "node is present in the config data to use the AS
        prepend action";
        description
          "action to prepend local AS number to the AS-path a
          specified number of times";

        leaf repeat-n {
          type uint8;
          description "number of times to prepend the local AS number";
        }
     }

      container set-community {

       presence "node is present in the config data when set-community
       action is used";
       description
         "action to set the community attributes of the route, along
         with options to modify how the community is modified";

       leaf-list communities {
         type union {
           type std-community-attr-type;
```

```
            type well-known-community-attr;
          }
          description
            "community values for the update";
        }

        leaf options {
          type set-community-option-type;
          description
            "options for modifying the community attribute with the
            specified values";
        }
      }

      container set-ext-community {

        presence "node is present in the config data when set-community
        action is used";
        description
          "action to set the extended community attributes of the
          route, along with options to modify how the community is
          modified";

        leaf-list communities {
          type union {
            type ext-community-attr-type;
            type well-known-community-attr;
          }
          description
            "community values for the update";
        }

        leaf options {
          type set-community-option-type;
          description
            "options for modifying the community attribute with the
            specified values";
        }
      }

      leaf set-route-origin {
        type bgp-origin-attr-type;
        description "set the origin attribute to the specified value";
      }

      leaf set-local-pref {
        type uint32;
        description "set the local pref attribute on the route update";
```

```
      }

      leaf set-next-hop {
        type bgp-next-hop-type;
        description "set the next-hop attribute in the route update";
      }

      leaf set-med {
        type uint32;
        description "set the med metric attribute in the route update";
      }
    }

    grouping control-flow-actions {
      description
        "Definitions for base set of policy action statements that
        manage the disposition or control flow of the policy";

      leaf accept-route {
        type empty;
        description "accepts the route into the routing table";
      }

      leaf reject-route {
        type empty;
        description "rejects the route";
      }

      leaf goto-next {
        type empty;
        description
          "proceed to evaluate the next policy statement in the
          policy definition";
      }

      leaf goto-policy {
        type string;
        description
          "proceed to the named policy definition and continue
          evaluating the policy";
      }

    }

    grouping conditions {
      description
        "Condition statement definitions for policy statements";
```

```
      leaf call-policy {
        type string;
        description
         "Applies the conditions from the specified policy definition
         in the current policy statement.";
      }

      uses condition-set-matches;
      uses condition-attribute-comparisons;

    }

    grouping actions {
      description
        "Action statement definitions for policy statements";

      uses set-attribute-actions;
      uses control-flow-actions;

    }

    grouping apply-policy-group {
      description
        "top level configuration for applying policies at various
         points in the configuration hierarchy";

      container apply-policy {
        description
          "Anchor point for policies in the BGP configuration.  Import
          and export policies are with respect to the local routing
          table, i.e., export (send) and import (receive).";

        leaf-list import-policies {
          type leafref {
            path "/bgp/policy/policy-definitions/policy-definition"
              + "/name";
            require-instance true;
          }
          description
            "list of policy names in sequence to be applied on
            receiving a routing update in the current context, e.g.,
            for the current peer group, neighbor, address family,
            etc.";
        }

        leaf-list export-policies {
          type leafref {
            path "/bgp/policy/policy-definitions/policy-definition"
```

```
            + "/name";
          require-instance true;
        }
        description
          "list of policy names in sequence to be applied on
          sending a routing update in the current context, e.g.,
          for the current peer group, neighbor, address family,
          etc.";
      }
    }
  }


  grouping policy-definition-group {
    description
      "top level set of policy defined sets and policy definitions";

    container policy {
      description
        "Top level container for BGP policy-related configuration
        items";

      container defined-sets {
        presence "Container for sets defined for matching in policy
        statements";
        description
          "Predefined sets of attributes used in policy match
          statements";

        uses defined-sets-definitions;
      }

      container policy-definitions {
        presence "Container for the set of policy definitions";
        description
          "Top level container for policy definitions";

        list policy-definition {
          key name;
          ordered-by user;
          description
            "List of top-level policy definitions, keyed by a unique
            name";

          leaf name {
            type string;
            description
              "Name of the top-level policy definition -- this name
```

```
                is used in references to the current policy";
            }

            list statements {
              key name;
              // TODO: names of policy statements withing a policy defn
              // should be optional, however, YANG requires a unique id
              // for lists; not sure that a compound key works either;
              // need to investigate further.
              ordered-by user;
              description
                "Name of this policy statement";

              leaf name {
                type string;
                description "name of the policy statement";
              }

              container conditions {
                description "Condition statements for this
                policy statement";

                uses conditions;
              }

              container actions {
                description "Action statements for this policy
                statement";

                uses actions;
             }
            }
          }
        }
      }
    }

    // augment statements

    // rpc statements

    // notification statements


  }
  <CODE ENDS>
```

5.3. BGP multiprotocol model

```
<CODE BEGINS> file bgp-multiprotocol.yang
module bgp-multiprotocol {

  yang-version "1";

  // namespace
  // TODO: change to an ietf or other more generic namespace
  namespace "http://google.com/yang/google-bgp-multiprotocol-cfg";

  prefix "bgp-mp";

  // import some basic inet types
  import ietf-inet-types { prefix inet; }
  import bgp-policy { prefix bgp-pol; }
  import bgp-operational { prefix bgp-op; }


  // meta
  organization
    "Google, AT&T, BT, Microsoft";

  contact
    "Google, Inc.
    1600 Amphitheatre Way
    Mountain View, CA  94043

    AT&T Labs
    200 S. Laurel Avenue
    Middletown, NJ  07748

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

    Microsoft
    205 108th Ave. NE, Suite 400
    Bellevue, WA 98004";

  description
    "This module is part of a YANG model for BGP protocol
    configuration, focusing on configuration of multiprotocol
    BGP, in particular various relevant address families (AFI) and
    sub-address families (SAFI).
```

```
      Identities (rather than enumerated types) are used to identify
      each AFI / SAFI type to make it easier for users to extend to
      pre-standard or custom AFI/SAFI types.  This module is only
      intended to capture the most";


    revision "2014-10-13" {
      description
        "Initial revision";
      reference "TBD";
    }

    // extension statements

    // feature statements

    // identity statements

    identity afi-type {
      description
        "base identity type for BGP address family identifiers (AFI)";
      reference "RFC 4760 - Multiprotocol Extensions for BGP-4";
    }

    identity safi-type {
      description
        "base identity type for BGP subsequent address family
        identifiers (SAFI)";
      reference "RFC 4760 - Multiprotocol Extensions for BGP-4";
    }

    identity ipv4-afi {
      base bgp-mp:afi-type;
      description
        "IPv4 AF identifier (AFI = 1)";
    }

    identity ipv6-afi {
      base bgp-mp:afi-type;
      description
        "IPv6 AF identifier (AFI = 2)";
    }

    identity unicast-safi {
      base bgp-mp:safi-type;
      description
        "unicast SAFI identifier (SAFI = 1)";
    }
```

```
identity l3vpn-unicast-safi {
  base safi-type;
  description
    "L3 / MPLS virtual private networks SAFI (SAFI = 128/129)";
  reference "RFC 4364 - BGP/MPLS IP Virtual Private Networks
  (VPNs)";
}

identity labeled-unicast-safi {
  base safi-type;
  description
    "labeled unicast SAFI identifier (SAFI = 4)";
  reference "RFC 3107 - Carrying Label Information in BGP-4";
}

identity l2vpn-vpls-afi {
  base afi-type;
  description
    "AFI for BGP L2 VPN / VPLS (AFI = 25)";
  reference "RFC 4761 - Virtual Private LAN Service (VPLS)
  Using BGP for Auto-Discovery and Signaling";
}

identity l2vpn-vpls-safi {
  base safi-type;
  description
    "BGP L2 VPN / VPLS service SAFI (SAFI = 65)";
}

identity multicast-safi {
  base safi-type;
  description
    "multicast SAFI (SAFI = 2)";
  reference "RFC 4760 - Multiprotocol Extensions for BGP-4";
}

identity multicast-vpn-safi {
  base safi-type;
  description
    "Multicast VPN SAFI (SAFI = 5)";
  reference "RFC 6514 - BGP Encodings and Procedures for Multicast
  in MPLS/BGP IP VPNs";
}

// typedef statements

// TODO: move this and other commonly types to a common bgp-types
// module
```

```
    typedef percentage {
      type uint8 {
        range "0..100";
      }
      description
        "Integer indicating a percentage value";
    }

    // grouping statements

    grouping address-family-common {
      description
        "Configuration that is for the address family level,
        but applies across AFI/SAFI";

      container prefix-limit {
        description
        "Configure the maximum number of prefixes that will be
        accepted from a peer.";

        leaf max-prefixes {
          type uint32;
          description
            "Maximum number of prefixes that will be accepted from
            the neighbor.";
        }

        leaf shutdown-threshold-pct {
          type percentage;
          description
            "Threshold on number of prefixes that can be received
            from a neighbor before generation of warning messages
            or log entries.  Expressed as a percentage of
            max-prefixes.";
        }

        leaf restart-timer {
          type decimal64 {
            fraction-digits 2;
          }
          units "seconds";
          description
            "Time interval in seconds after which the BGP session
            is reestablished after being torn down due to exceeding
            the max-prefixes limit.";
        }
      }
```

```
      // policies can be applied at a specific AF level
      uses bgp-pol:apply-policy-group;


    }

    grouping ipv4-ipv6-unicast-common {
      description
        "common configuration for base ipv4 and ipv6 unicast; may
        need to be split into separate containers for each of ipv4
        and ipv6";

      container ipv4-ipv6-unicast {
        // YANG uses XPath 1.0 expression syntax
        when "(../../afi-name = 'ipv4-afi' or " +
          "../../afi-name = 'ipv6-afi') " +
          "and ../safi-name = 'unicast-safi'" {
            description
              "Include this container for unicast ipv4 or ipv6
              AFI-specific configuration";
        }
        description "ipv4 unicast config items";

        leaf send-default-route {
          // TODO: consider moving this to policy
          type boolean;
          default "false";
          description "if set to true, send the default route, i.e.,
          0.0.0.0/0 to the neighbor(s)";
        }
      }
    }


    grouping ipv4-l3vpn-unicast-group {
      description
        "configuration group for L3 VPN VRFs for IPv4";

      container ipv4-l3vpn-unicast {
        when "../../afi-name = 'bgp-mp:ipv4-afi' and " +
        "../safi-name = 'l3vpn-unicast-safi'" {
          description
            "Include this container when AFI = ipv4 and
            SAFI = l3vpn-unicast";
        }
        description "ipv4 l3vpn config items";
```

```
        list vrfs {
          key name;
          description "list of configured VRFs";

          leaf name {
            type string;
            description "name / identifier of the VRF";
          }

          leaf route-distinguisher {
            // TODO: consider expanding to a union type to make it more
            // convenient to express as AS:addr or other common formats
            type uint64;
            description
              "route distinguisher value assigned to this VRF";
          }

          uses bgp-pol:apply-policy-group;

          /* additional leafs to consider --- should these be in BGP?
          interface-name
          retain-local-label-size
          advertise-best-external
          no-synchronization
          */

        }
      }
    }

    grouping ipv6-l3vpn-unicast-group {
      description
        "configuration group for L3 VPN VRFs for IPv6";

      container ipv6-l3vpn-unicast {
        when "../../afi-name = 'bgp-mp:ipv6-afi' and " +
        "../safi-name = 'l3vpn-unicast-safi'" {
          description
            "Include this container only when AFI = ipv6 and
            SAFI = l3vpn-unicast";
        }
        description "ipv6 l3vpn config items";
      }
    }

    grouping ipv4-labeled-unicast-group {
      description
        "configuration group for IPv4 labeled unicast";
```

```
      container ipv4-labeled-unicast {
        when "../../afi-name = 'ipv4-afi' and " +
        "../safi-name = 'labeled-unicast-safi'" {
          description
            "Include this container when AFI = ipv4 and
            SAFI = labeled-unicast";
        }
        description "ipv4 labeled unicast config items";
      }
    }

    grouping l2vpn-group {
      description
        "configuration group for L2 VPN";

      container l2vpn {
        // TODO: confirm that both AFI/SAFI values are set
        // for L2 VPNs
        when "../../afi-name = 'l2vpn-vpls-afi' and " +
        "../safi-name = 'l2vpn-vpls-safi'" {
          description
            "Include this container when AFI = l2vpn-vpls and
            SAFI = l2vpn-vpls";
        }
        description "l2vpn config items";
      }
    }

    grouping ipv4-multicast-vpn-group {
      description
        "configuration group for IPv4 multicast VPNs";

      container ipv4-multicast-vpn {
        when "../../afi-name = 'ipv4-afi' and " +
        "../safi-name = 'multicast-vpn-safi'" {
          description
            "Include this container when AFI = ipv4 and
            SAFI = multicast-vpn";
        }
        description "ipv4 multicast vpn config items";
      }
    }

    grouping ipv6-multicast-vpn-group {
      description
        "configuration group for IPv6 multicast VPNs";

      container ipv6-multicast-vpn {
```

```
        when "../../afi-name = 'ipv6-afi' and " +
        "../safi-name = 'multicast-vpn-safi'" {
          description
            "Include this container when AFI = ipv6 and
            SAFI = multicast-vpn";
        }
        description "ipv6 multicast vpn config items";
      }
    }


    grouping address-family-configuration {
      description "Configuration options that are applied at the
      address family level.";

      list afi {

        key "afi-name";
        description
          "Per address-family configuration, uniquely identified by AF
          name.";
        leaf afi-name {
          type identityref {
            base "afi-type";
          }
          description
            "Address family names are drawn from the afi-type base
            identity, which has specific address family types as
            derived identities.";
        }

        list safi {
          key "safi-name";
          description
            "Per subsequent address family configuration, under a
            specific address family.";

          leaf safi-name {
            type identityref {
              base "safi-type";
            }
            description
              "Within each address family, subsequent address family
              names are drawn from the subsequent-address-family base
              identity.";
          }

          // these grouping references conditionally add config nodes
```

```
          // that are specific to each AFI / SAFI combination
          uses ipv4-ipv6-unicast-common;
          uses ipv4-l3vpn-unicast-group;
          uses ipv6-l3vpn-unicast-group;
          uses ipv4-labeled-unicast-group;
          uses l2vpn-group;
          uses ipv4-multicast-vpn-group;
          uses ipv6-multicast-vpn-group;

          // this grouping pulls in the config items common across
          // AF/SAFI combinations
          uses address-family-common;


        }
       // operational state common acrossr address families
       uses bgp-op:bgp-op-af-group;
     }
   }


   // data definition statements

   // augment statements

   // rpc statements

   // notification statements

}
<CODE ENDS>
```

5.4. BGP operational data model

```
<CODE BEGINS> file bgp-operational.yang
module bgp-operational {

  yang-version "1";

  // namespace
  // TODO: change to an ietf or other more generic namespace
  namespace "http://google.com/yang/google-bgp-operational";

  prefix "bgp-op";

  // import some basic inet types
  import ietf-inet-types { prefix inet; }
```

```
     // meta

     organization
       "Google, AT&T, BT, Microsoft";

     contact
       "Google, Inc.
       1600 Amphitheatre Way
       Mountain View, CA  94043

       AT&T Labs
       200 S. Laurel Avenue
       Middletown, NJ  07748

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

       Microsoft
       205 108th Ave. NE, Suite 400
       Bellevue, WA 98004";

     description
       "This module is part of a YANG model for BGP protocol
       configuration, focusing on operational data (i.e., state
       variables) related to BGP operations";


     revision "2014-10-13" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

     grouping bgp-op-global-group {
       description
```

```
        "top level container for operational state data";

      container bgp-global-state {
        config false;
        description
          "data definitions for operational state variables related
          to the global BGP instance";
      }
    }

    grouping bgp-op-af-group {
      description
        "top level container for operational state data";

      container bgp-af-common-state {
        config false;
        description
          "data definitions for operational state variables related
          to all BGP address families instance";
      }
    }

    grouping bgp-op-peergroup-group {
      description
        "top level container for operational state data";

      container bgp-group-common-state {
        config false;
        description
          "data definitions for operational state variables related
          to BGP peer groups";
      }
    }

    grouping bgp-op-neighbor-group {
      description
        "top level container for operational state data";

      container bgp-neighbor-common-state {
        config false;
        description
          "data definitions for operational state variables related
          to BGP neighbor sesions";
      }
    }

    // data definition statements
```

```
    // augment statements

    // rpc statements

    // notification statements
  }
  <CODE ENDS>
```

6. References

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

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

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

Appendix A. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Chris Chase, Ed Crabbe, Josh George, Vijay Gill, Ina Minei, Ashok Narayanan, Steve Padgett, Puneet Sood, and Jim Uttaro.

Authors' Addresses

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

Email: aashaikh@google.com

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

Email: kd6913@att.com

Deepak Bansal
Microsoft
205 108th Ave. NE, Suite 400
Bellevue, WA
US

Email: dbansal@microsoft.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/