Abstract

This document presents an approach for a YANG model catalog and registry that allows users to find models relevant to their use cases from the large and growing number of YANG modules being published. The model catalog may also be used to define bundles of YANG modules required to realize a particular service or function.

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 January 9, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must...
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.

Table of Contents

1. Introduction ............................................. 2
2. Model catalog and registry requirements .................. 3
3. Model catalog schema ...................................... 5
   3.1. Module information .................................. 5
4. Identifying interoperable models .......................... 7
   4.1. Schema for module feature bundles .................. 8
5. Module composition with feature bundles .................. 8
   5.1. Schema for module feature bundles .................. 9
6. Security Considerations .................................... 9
7. IANA Considerations ....................................... 10
8. YANG modules ............................................ 10
9. References ................................................ 32
   9.1. Normative references ................................ 32
   9.2. Informative references .............................. 32
Appendix A. Change summary .................................. 33
   A.1. Changes between revisions -00 and -01 ............... 33

Authors’ Addresses ........................................ 33

1. Introduction

As YANG [RFC6020] adoption and usage grows, the number of YANG models (and corresponding module and submodule files) published is increasing rapidly. This growing collection of modules potentially enables a large set of management use cases, but from a user perspective, it is a daunting task to navigate the largely ad-hoc landscape of models to determine their functionality, availability, and implementations. For example, the IETF Routing Area Coordination page [RTG-AD-YANG] currently tracks nearly 150 YANG models related to layer 2 and layer 3 technologies.

YANG models are also being developed and published beyond the IETF, for example by open source projects, other standards organizations, and industry forums. These efforts are generally independent from each other and often result in overlapping models. While we recognize that models may come from multiple sources, the current approach of having a flat online listing of models is not sufficient to help users find the models they need, along with the information to retrieve and utilize the models in actual operational systems. There is a need for a wider registry and catalog of available models that provides a central reference for model consumers and developers.
The idea of a model catalog is inspired by service catalogs in traditional IT environments. Service catalogs serve as software-based registries of available services with information needed to discover and invoke them.

In earlier proposals [I-D.openconfig-netmod-model-structure] we motivated the need for a common structure that allows a set of models to be used together coherently in order to manage, for example, a complete network device. Other efforts have subsequently proposed further options for modeling the complete device structure [I-D.rtgyangdt-rtgwg-device-model]. We also briefly described the notion of a model catalog to provide a structured view of all of the models available from different organizations. In this document, we further elaborate on some of the details and use cases for a model catalog and registry.

There are recent proposals that address related issues in terms of understanding the set of YANG models available on a device [I-D.ietf-netconf-yang-library], and how to classify models based on their role in describing a multi-layer service [I-D.ietf-netmod-yang-model-classification]. The latter, in particular, describes a taxonomy for classifying YANG models that could also be used in the model catalog, though it does not address the problem of classifying model functionality, which is a key requirement.

2. Model catalog and registry requirements

At a high level, the model catalog must provide enough information for users to determine which models are available to describe a specific service or technology, and attributes of those models that would help the user select the best model for their scenario. While this draft does not specifically address selection criteria -- they would be specific to each user -- some examples include:

- model maturity, including availability of server implementations (e.g., native device support)
- available of co-requisite models, and complexity of the model dependencies
- identity and reputation of the entity or organization publishing the model

The model catalog should, therefore, include key information about YANG modules, including:
o organization responsible for publishing and maintaining the module with contact information; organizations may include standards bodies (SDOs), industry forums, open source projects, individuals, etc.

o classification of the module or model, which could be along several axes, e.g., functional category, service vs. element models, commercial vs. free-to-use, etc.; currently, identities are available for the classifications defined in [I-D.ietf-netmod-yang-model-classification].

o for open models, the license under which the model is distributed; this is important if there are limitations on how the model may be modified or redistributed

o module dependencies, e.g., a list of all of the YANG modules that are required

o pointer to the YANG module, e.g., a URI that can be machine-processed

o implementation information, for example, a list of available server implementations that support the module

o authentication information to allow users to verify that the model they retrieve is authentic and unaltered

Establishing a globally applicable classification scheme for models is not straightforward -- each organization developing models likely has its own taxonomy or organization strategy for YANG modules. This is an area of the catalog that is likely to require extensibility and customization, e.g., by letting each organization augment the schema with its own categories. Similarly, users may want to define their own classifications for use by internal systems.

The proposed catalog schema should be useful as a local database, deployed by a single user, and also as a global registry that can be used to discover available models. For example, the local catalog could be used to define the approved set of models for use within an organization, while the registry serves as a channel for all model developers to make information about their models available. The IETF XML Registry [RFC3688], maintained by IANA serves a similar purpose for XML documents used in IETF protocols, but it is limited to IETF-defined YANG models, is tied to XML encoded data, and has a very limited schema.

The registry implementation could be as simple as a metadata database that reflects the proposed catalog schema, along with means for
online access and viewing. A key requirement for the online registry would be a robust query capability that allows users to search for modules meeting a variety of selection criteria, along with an easy way to retrieve modules (where applicable).

3. Model catalog schema

We propose a schema for the model catalog defined using YANG (see the modules in Section 8). The YANG modules in the catalog are organized at the top level by the publishing organization and its associated contact information. The catalog structure is shown below.

```
+--rw organizations
  +--rw organization* [name]
    +--rw name       string
    +--rw type?      identityref
    +--rw contact?   string
  +--rw modules
    +--rw module* [name]
      +--rw name                string
      +--rw namespace?          string
      +--rw prefix?             string
      +--rw revision?           string
      +--rw summary?            string
      +--rw module-version?     string
      +--rw module-hierarchy    |
      |   ...                    |
      +--rw classification     |
      |   ...                    |
      +--rw dependencies        |
      |   ...                    |
      +--rw module-usage        |
      |   ...                    |
      +--rw implementations     ...
```

In this model, each organization publishes a list of available modules, each module having associated data describing its classification, dependencies, usage information, and implementation information. In addition, some of the basic module metadata is included in the catalog, e.g., namespace, prefix, and revision.

3.1. Module information

Each module has several types of information associated with it. These are described below.
The basic information includes module metadata as mentioned above and also the location of module in its own dependency chain. The module-hierarchy container indicates whether the module is a submodule of another module, and has a reference to its parent module.

The classification data is meant to capture some base information but leave the taxonomy largely to model publishers. The category and subcategory leaves are identities that are expected to be augmented with additional values. The classification also includes a status to indicate the development or deployment status of the module, e.g., whether it is purely experimental, or mature enough for production use. The classification data is shown below:

```
+--rw module* [name]
    +--rw classification
        +--rw status?    identityref
        +--rw category?  identityref
        +--rw subcategory? identityref
```

In this initial version of the catalog schema, the module dependencies are represented as a simple list of references to co-requisite modules. The model assumes that required modules are also represented in the catalog, and that only the first-level dependencies are included in the list. That is, each of the listed modules can be examined to determine its dependencies.

The usage data contains information required to retrieve and validate the module. Specifically, it includes authentication and validation data to ensure the origin and integrity of the module, respectively. The authentication information will be further developed in future revisions of the document; in the current version, these can be considered placeholders. This section also includes a URI for modules that can be downloaded directly. This part of the schema is shown below:

```
+--rw module* [name]
    +--rw module-usage
        +--rw authentication? string
        +--rw md5-hash?    string
        +--rw access-uri?  inet:uri
```

The implementation container provides information about known implementations of the module, for example by network devices or other servers. This data is structured as a list to account for multiple implementations of a module, e.g., by different vendors. It includes some basic information about the platform on which the module is supported, and the status of the implementation, but it is expected that details and limitations of the implementation will
require consulting the implementor. The implementation information in the catalog is shown below:

```yaml
+-rw module* [name]
  +--rw implementations
    +--rw implementation* [implementation-id]
      +--rw implementation-id string
      +--rw description? string
      +--rw reference? union
      +--rw implementor-name? string
      +--rw platform? string
      +--rw platform-version? string
      +--rw implementation-status? identityref
```

4. Identifying interoperable models

YANG models for configuration and operational state data are under active development and still maturing, especially with regard to their use in production networks. As models (and their corresponding YANG modules) evolve and are revised, there is a significant challenge for users to identify the set of models that are known, or designed, to work together. This is made more complicated by the fact that models are being sourced by different organizations which may use different modeling conventions. Since there are often cross-dependencies between modules (e.g., interface configuration and various routing protocols), it is critical that users understand which modules can be used together.

The proposed model catalog defines the notion of "release" bundles which provide a grouping of YANG modules that are part of a cohesive release. For example, a release bundle can be defined at a granular level to collect all of the modules related to interface configuration that are known to work together. These bundles can be further grouped into larger releases of models that interoperate, e.g., a release containing interoperable routing, interface, and policy-related modules.

Release bundles are also useful for implementors who can indicate the set of supported modules in a software release by identifying the corresponding release bundle version. We expect that users and publishers of models would be the primary source of release bundle definitions, and vendors and implementors would be the primary consumers.
4.1. Schema for module feature bundles

We propose an initial YANG-defined schema for describing a "feature" bundle for building composite services and functions (shown below).

```
  +--rw release-bundles
    +--rw release-bundle* [name version]
      +--rw name       string
      +--rw version    string
      +--rw models
        +--rw model* [name]
          +--rw name                   string
          +--rw compatible-versions*   string
          +--rw type?                  identityref
          +--rw bundle?    -> ../name
          +--rw module?    -> .../modules/module/name
```

The release bundle has a name and version assigned to the bundle itself, and a list of models that are part of the bundle. The list may include a reference to a module or another bundle. The compatible-versions list indicate which semantic versions [OC-SEMVER] of the respective module or bundle are known to work together.

5. Module composition with feature bundles

From an operational perspective, the utility of a single module is quite limited. Most, if not all, use cases require multiple modules that work together coherently. Managing a network device typically requires configuration and operational state models for device-wide services, network protocols, virtual instances, etc. Network services, such as those delivered by many service providers, require not only infrastructure-level management models, such as devices and protocols, but also service-level models that describe service parameters.

The model catalog and registry provides a common way to define service bundles, or recipes, that describe the set of modules required for realizing the feature or service. For example, a Layer 3 VPN bundle would list its required configuration and state models, including VRFs, interfaces, BGP, policy, ACLs, and QoS. Similar bundles 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 bundle definitions complement actual configuration models for such services, which may focus on providing an abstracted set of configuration or operational state variables. These variables would then be mapped onto device level variables. We leave discussion of such mapping mechanisms to future revisions.
Bundle definitions are particularly useful for organizations that identify and validate a set of models that are used to build a service, and then define an approved bundle based on that set. Users within the organization can be assured that the corresponding bundles are known to work together to support the desired service. Another use case for bundle definitions is for third-party testing or certification organizations to provide services to validate a set of modules and maintain the bundle.

5.1. Schema for module feature bundles

We propose an initial YANG-defined schema for describing a "feature" bundle for building composite services and functions (shown below).

```
+--rw bundle
 +--rw name?          string
 +--rw version?       string
 +--rw description?   string
 +--rw category?      string
 +--rw subcategory?   string
 +--rw modules
    +--rw module* [module-type]
        +--rw module-type             string
        +--rw catalog-reference? -> /cat:organizations/.../module/name
        +--rw application-sequence?   uint8
```

Each feature bundle includes basic information such as the name of the feature or service, the bundle version, and its category and subcategory. The modules comprising the bundle are contained in the modules list with a reference to the module in the catalog. The application sequence number can be used to indicate an ordering of the modules in realizing the service, for example, device or element configuration modules followed by service configuration models. The application sequence is a high level indication; a complete realization of the service would require a detailed definition of the mapping between module variables at different levels as discussed in Section 5.

6. Security Considerations

The model catalog and registry described in this document do not define actual configuration and state data, hence are not directly responsible for security risks.

However, since the model catalog is intended to be an authoritative and authenticated database of published modules, there are security considerations in securing the catalog (both contents and access),
and also in authenticating organizations that deposit data into the catalog.

7. IANA Considerations

The YANG model catalog is intended to complement the IANA XML Registry. YANG modules defined in this document may be entered in the XML registry if they are placed or redirected for the standards track, with an appropriate namespace URI.

8. YANG modules

The main model catalog and associated types modules are listed below.

<CODE BEGINS> file "openconfig-catalog-types.yang"

module openconfig-catalog-types {
    yang-version "1";

    // namespace
    namespace "http://openconfig.net/yang/catalog-types";

    prefix "oc-cat-types";

    import openconfig-extensions { prefix oc-ext; }

    // meta
    organization "OpenConfig working group";

    contact
        "OpenConfig working group
        www.openconfig.net";

    description
        "This module defines types and identities used by the OpenConfig
        YANG module catalog model.";

    oc-ext:openconfig-version "0.1.0";

    revision "2016-02-15" {
        description
            "Initial OpenConfig public release";
        reference "0.1.0";
    }

    revision "2015-10-18" {


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

// extension statements

// feature statements

// identity statements

identity IMPLEMENTATION_STATUS_TYPE {
  description
    "Indications of the status of a module’s implementation on a
device or server";
}

identity IN_PROGRESS {
  base IMPLEMENTATION_STATUS_TYPE;
  description
    "Implementation is in progress";
}

identity PLANNED {
  base IMPLEMENTATION_STATUS_TYPE;
  description
    "Implementation is planned";
}

identity COMPLETE {
  base IMPLEMENTATION_STATUS_TYPE;
  description
    "Implementation is complete and fully supports the model";
}

identity PARTIAL {
  base IMPLEMENTATION_STATUS_TYPE;
  description
    "Implementation is complete, but only supports the model
partially";
}

identity MODULE_STATUS_TYPE {
  description
    "Indicates the deployment status of the module";
}

identity EXPERIMENTAL {

base MODULE_STATUS_TYPE;
description "Module should be considered experimental, not deployed in production settings";
}

identity PRODUCTION {
  base MODULE_STATUS_TYPE;
description "Module is suitable for use in production, or has been deployed in production";
}

identity MODULE_CATEGORY_BASE {
description "Base identity for the module category. It is expected that publishing organizations will define additional derived identities to describe their categorization scheme.";
}

identity MODULE_SUBCATEGORY_BASE {
description "Base identity for the module subcategory. It is expected that publishing organizations will define additional derived identities to describe their categorization scheme.";
}

identity ORGANIZATION_TYPE {
description "Publishing organization type for the set of modules";
}

identity STANDARDS {
  base ORGANIZATION_TYPE;
description "Standards development organization (SDO) publisher type";
}

identity INDUSTRY {
  base ORGANIZATION_TYPE;
description "Industry forum or other industry group";
}

identity COMMERCIAL {
  base ORGANIZATION_TYPE;
description "Commercial entity, company, etc.";
}
identity INDIVIDUAL {
  base ORGANIZATION_TYPE;
  description
    "For modules published by an individual";
}

identity IETF_MODEL_LAYER {
  base MODULE_CATEGORY_BASE;
  description
    "Describes layering of models based on their abstraction
    level as defined by IETF model classification proposals";
  reference
    "IETF draft-ietf-netmod-yang-model-classification";
}

identity IETF_MODEL_TYPE {
  base MODULE_SUBCATEGORY_BASE;
  description
    "IETF proposed classification dimension of YANG model types as
    standard YANG models, vendor-specific, or user-specific YANG
    models and extensions";
  reference
    "IETF draft-ietf-netmod-yang-model-classification";
}

identity IETF_NETWORK_SERVICE {
  base IETF_MODEL_LAYER;
  description
    "Service-layer model as defined by IETF classification
    proposal";
}

identity IETF_NETWORK_ELEMENT {
  base IETF_MODEL_LAYER;
  description
    "Network element-layer model as defined by IETF classification
    proposal";
}

identity IETF_TYPE_STANDARD {
  base IETF_MODEL_TYPE;
  description
    "Models published by standards-defining organizations (SDOs)";
}

identity IETF_TYPE_VENDOR {
base IETF_MODEL_TYPE;
description
  "Developed by organizations (e.g., vendors) with the intent
to support a specific set of implementations under control of
that organization";
}

identity IETF_TYPE_USER {
  base IETF_MODEL_TYPE;
description
  "Developed by organizations that operate YANG-based
  infrastructure including devices and orchestrators.
The intent of these models is to express the specific needs
for a certain implementation, above and beyond what is provided
by vendors";
}

// grouping statements
// data definition statements
// augment statements
// rpc statements
// notification statements

<CODE ENDS>

<CODE BEGINS> file "openconfig-module-catalog.yang"

module openconfig-module-catalog {
  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/module-catalog";

  prefix "oc-cat";

  // import some basic types
  import ietf-inet-types { prefix inet; }
  import openconfig-catalog-types { prefix oc-cat-types; }
  import openconfig-extensions { prefix oc-ext; }

import openconfig-release-bundle { prefix oc-relbundle; }

// meta
organization "OpenConfig working group";

contact
  "OpenConfig working group
  www.openconfig.net";

description
  "This module provides a schema for cataloging and describing
  YANG models published across various organizations.";

oc-ext:openconfig-version "0.1.0";

revision "2016-02-15" {
  description
    "Initial OpenConfig public release";
  reference "0.1.0";
}

revision "2015-10-18" {
  description
    "Initial revision";
  reference "TBD";
}

// extension statements

// feature statements

// identity statements

// typedef statements

// grouping statements

grouping module-implementation-information {
  description
    "Data describing any available implementations";

container implementations {
  description
    "Container for module implementation information";

  list implementation {
    key "implementation-id";
  }
}
description
"List of available implementations, keyed by an identifier provided by either the implementor or the module maintainer. Such a key avoids needing a complex composite key to uniquely identify an implementation."

leaf implementation-id {
  type string;
  description
  "An identifier for the implementation, provided by the implementor or the module maintainer. This id should uniquely identify a specific implementation of the module, e.g., based on the vendor, platform, and platform version.";
}

leaf description {
  type string;
  description
  "A text summary of important information about the implementation";
}

leaf reference {
  type union {
    type string;
    type inet:uri;
  }
  description
  "A URI or text reference to more detailed information about the implementation.";
}

leaf implementor-name {
  type string;
  description
  "Name of the vendor or entity providing the module implementation";
}

leaf platform {
  type string;
  description
  "Name of the server platform on which the implementation is available -- this could be the model name of a network device, a server OS, etc.";
}
leaf platform-version {
  type string;
  description
  "Implementor-defined version name or number for the
  module implementation, corresponding to the platform.
  This could be the firmware version of a network device
  such as a router, OS version, or other server platform
  version.";
}

leaf implementation-status {
  type identityref {
    base oc-cat-types:IMPLEMENTATION_STATUS_TYPE;
  }
  description
  "Indicates the status of the implementation, e.g.,
  complete, partial, in-progress, etc. Implementors
  may define additional values for the base identity";
}

grouping module-dependency-information {
  description
  "Information about module dependencies";
  container dependencies {
    description
    "Container for information about module dependencies";
    leaf-list required-module {
      type leafref {
        path "../../name";
      }
      description
      "TODO: should this list be complete, or only the first-
      level dependencies?
      "A simple list of modules that are prerequisites for the
      current module. It is expected that each of the required
      modules would in turn list their dependencies. The list
      values should be references to other modules in the
      catalog.";
    }
  }
}

grouping module-classification-information {

}
description
"Data describing the module’s classification(s)";

classification {
  description
  "Container for data describing the module’s classification";

deployment-status {
  type identityref {
    base oc-cat-types:MODULE_STATUS_TYPE;
  }
  description
  "Deployment status of the module -- experimental, standards-track, production, etc.";}
}
category {
  type identityref {
    base oc-cat-types:MODULE_CATEGORY_BASE;
  }
  description
  "Categorization of the module based on identities defined or used by the publishing organizations.";
}
subcategory {
  type identityref {
    base oc-cat-types:MODULE_SUBCATEGORY_BASE;
  }
  description
  "Sub-categorization of the module based on identities defined or used by the publishing organizations.";
}
}

module-usage-information {
  description
  "Data pertaining to retrieval and usage of the module";

  module-usage {
    description
    "Container for data pertaining to retrieval and usage of the module";

    authentication {
      //TODO: requires more detailed model for different types of authentication / validation schemes
    }
  }
}

type string;
description
  "Authentication information to allow
  users to verify that the model originates from
  stated organization, e.g., X.509 certificate";
}

leaf md5-hash {
  type string;
description
  "MD5 hash of the module file, used by users to validate
  data integrity";
}

leaf access-uri {
  type inet:uri;
description
  "URI where module can be downloaded. Modules may be
  made available from the catalog maintainer, or directly
  from the publisher";
}
}
}
grouping module-base-information {
  description
    "Basic information describing the module, e.g., the
    YANG metadata in the module preface.";

  leaf name {
    type string;
description
    "The module name, as defined in the YANG module file."
  }

  leaf namespace {
    //type inet:uri;
    type string;
description
    "Published namespace of module";
  }

  leaf prefix {
    type string;
description "Published prefix of module";
  }

  leaf revision {

}
type string;

description
"Date in the revision statement of the module";
}

leaf summary {

type string;

description
"Brief summary of the module description";
}

leaf module-version {

type string;

description
"Optional version number for the module, in addition to the
YANG revision statement";
}

container module-hierarchy {

description
"YANG module hierarchy specification";

leaf module-hierarchy-level {

type uint8 {
    range 1..5;
}

default 1;

description
"Module hierarchy level. If this is a sub-module,
it is set to > 1, depending
on the hierarchy level of the sub-module";
}

leaf module-parent {

    when ".../module-hierarchy-level > '1'" {
        description "Only applicable to sub-modules";
    }

type leafref {
    path ".../../*name";
}

description
"Parent module, if this is a sub-module";
}
}

//module-base-information

grouping organization-information {

description
"Data describing the publisher of the module";

leaf name {
  type string;
  description
    "Name of Organization defining YANG Module:
    Standards Body examples:
      ietf, ieee,.opendaylight, etc.
    Commercial entity examples:
      AT&T, Facebook
    Name of industry forum examples:
      openconfig, other";
}

leaf type {
  type identityref {
    base oc-cat-types:ORGANIZATION_TYPE;
  }
  description
    "YANG modules publication authority";
}

leaf contact {
  type string;
  description
    "Contact information for the publishing organization";
}


grouping module-catalog-top {
  description
    "Top level structure of the module catalog";

  container organizations {
    description
      "List of organizations owning modules";

    list organization {
      key "name";
      description
        "List of organizations defining the YANG Modules";

      uses organization-information;
    }
  }
}
uses oc-relbundle:release-bundle-top;

container modules {
    description
    "Modules published by this organization";

    list module {
        key "name";
        description
        "List of published modules from the organization";

        uses module-base-information;
        uses module-classification-information;
        uses module-dependency-information;
        uses module-usage-information;

    }
}

uses module-implementation-information;

// data definition statements
uses module-catalog-top;

// augment statements

}
import openconfig-extensions { prefix oc-ext; }

// meta
organization "OpenConfig working group";

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

description
"This module can be used to build network features using
published YANG Models."

oc-ext:openconfig-version "0.1.0";

revision "2016-02-25" {
  description
  "Initial OpenConfig public release";
  reference "0.1.0";
}

identity MODEL_TYPE {
  description
  "A base identity used to reference the type of
  model that is specified in a feature bundle";
}

identity MODULE {
  base MODEL_TYPE;
  description
  "The model consists of a single entry within the
  YANG catalogue";
}

identity BUNDLE {
  base MODEL_TYPE;
  description
  "The model entry refers to another bundle within
  the YANG release bundle catalogue";
}

grouping release-bundle-common {

  description
  "Common characteristics of a release bundle";

  leaf name {
    type string;
description
"A canonical name for the overall bundle which is to be
released together. This name is consistent over multiple
releases";
}

leaf version {
  type string {
  }
  description
  "A semantic version number for the overall bundle. This
  version is to be defined as per the approach specified
  in the OpenConfig semantic version guidance - and hence
  is of the form x.y.z, where x is the major version, y is
  the minor version, and z is the patch level";
}
}

grouping release-bundle-models {

description
  "Parameters relating to models within release bundles";

container models {
  description
  "List of models which make up this release bundle. A
  model is defined as an individual YANG module as
  specified in the YANG catalogue, or another release
  bundle - which can be used to group multiple YANG
  models together.";

  list model {
    key "name";
    description
    "A set of modules or bundles which are part of the bundle of
    models. For example, if 'ietf-yang-types' were to be specified
    within the bundle, then this would refer to the individual
    entry within the module catalogue. If the type of the entry is
    set to bundle, then for example, openconfig-bgp could be
    referenced - which itself consists of separate modules.";

    leaf name {
      type string;
      description
      "Name of the module set which is included in this bundle -
      for example, 'openconfig-bgp'";
    }
  }
}
leaf-list compatible-versions {
    type string {
        pattern "[0-9\*]+\.\[0-9\*]+\.\[0-9\*]+";
    }
    description
    "A list of semantic version specification of the versions of the specified module which can be considered to be compatible when building this version of the bundle. Version specifications may be added when changes are made to a module within a bundle, and this does not affect the interaction between it and other modules. In general, it is expected that backwards compatible changes to an individual module do not affect the compatibility of that module with other modules, and hence wildcard matches are allowed within the list.";
}

leaf type {
    type identityref {
        base MODEL_TYPE;
    }
    description
    "The type of model that is to be included within the feature bundle. When this value is set to MODULE then the entry can be directly looked up in the YANG catalog, whereas when it is set to BUNDLE the entry must be looked up for another bundle.";
}

leaf bundle {
    when ".../type = ‘BUNDLE’" {
        description
        "Specify the bundle name only when the type is equal to BUNDLE";
    }
    type leafref {
        path ".../name";
    }
    description
    "A reference to other bundles which are included within this bundle.";
}

leaf module {
    when ".../type = ‘MODULE’" {
        description

"Specify the module name only when the type is equal to MODULE";
}
type leafref {
   // we are at /organizations/organization/release-bundles
   // /bundle/models/model/modules
   path "../../../modules/module/name";
}
description
   "A reference to modules that are included within the bundle";
}
}
}

grouping release-bundle-top {

description
   "Top-level container for a release bundle";

container release-bundles {
   description
      "List of release bundles";

   list release-bundle {
      key "name version";

      description
         "List of release bundles - sets of modules which are commonly inter-operable";

      uses release-bundle-common;
      uses release-bundle-models;
   }
}
} //bundle

<CODE ENDS>

The feature bundle module is listed below.

<CODE BEGINS> file "openconfig-feature-bundle.yang"
module openconfig-feature-bundle {

  // namespace
  namespace "http://openconfig.net/yang/feature-bundle";

  prefix "oc-featbundle";

  import openconfig-module-catalog { prefix oc-cat; }
  import openconfig-extensions { prefix oc-ext; }

  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
    netopenconfig@googlegroups.com"

  description
    "This module can be used to build network features using
     published YANG Models.";

  oc-ext:openconfig-version "0.2.0";

  revision "2016-02-25" {
    description
      "OpenConfig revision to specify
       feature bundles";
    reference "0.2.0";
  }

  revision "2016-02-15" {
    description
      "Initial OpenConfig public release";
    reference "0.1.0";
  }

  revision "2015-10-18" {
    description
      "Initial revision";
    reference "TBD";
  }

  grouping feature-bundle-information {

    description
      "Template defining the bundle";

    leaf name {

leaf version {
    type string;
    description "bundle version number";
}

leaf description {
    type string;
    description "User defined information about bundle";
}

leaf category {
    type string;
    description "Categorization of bundle such as:
    network, service, oam, experimental, other";
}

leaf subcategory {
    type string;
    description "Sub-Categorization of bundle such as:
    protocol, operational, other";
}

} //bundle-template

grouping feature-bundle-ingredients {
    description "Module ingredients used in bundle";

    container modules {
        description "Modules that comprise the bundle";

        list module {
            key "module-type";

            description "List of modules from yang-module-catalog comprising
            the bundle";
        }
    }
}
leaf module-type {
  type string;
  description
    "A user-define type of the module";
}

leaf catalog-reference {
  type leafref {
    path "/oc-cat:organizations"
    + "/oc-cat:organization"
    + "/oc-cat:modules"
    + "/oc-cat:module"
    + "/oc-cat:name";
  }
  description
    "Link to the module metadata in the model catalog";
}

leaf application-sequence {
  type uint8;
  description
    "Sequence number indicating order of application of module";
}

} //module-info
} //bundle-modules
} //bundle-ingredients

grouping feature-bundle-top {
  description
    "Top-level grouping for OpenConfig feature bundles";
}

container feature-bundles {
  description
    "List of feature bundles";

  list feature-bundle {
    key "name";

    description
      "List of feature bundles - sets of modules that combine to create a set of functionality.";

    uses feature-bundle-information;
    uses feature-bundle-ingredients;
  }
}
Required extension modules included below.

<CODE BEGINS> file "openconfig-extensions.yang"

module openconfig-extensions {
  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/openconfig-ext";
  prefix "oc-ext";

  // meta
  organization "OpenConfig working group";
  contact
    "OpenConfig working group
     www.openconfig.net";
  description
    "This module provides extensions to the YANG language to allow
     OpenConfig specific functionality and meta-data to be defined.";

  revision "2016-07-08" {
    description
      "OpenConfig public release";
    reference "TBD";
  }

  revision "2015-10-09" {
    description
      "Initial OpenConfig public release";
    reference "TBD";
  }

  revision "2015-10-05" {
    description
      "Initial revision";
  }

<CODE ENDS>
reference "TBD";
}

// extension statements
extension openconfig-version {
  argument "semver" {
    yin-element false;
  }
  description "The OpenConfig version number for the module. This is expressed as a semantic version number of the form: x.y.z where:
  * x corresponds to the major version,
  * y corresponds to a minor version,
  * z corresponds to a patch version.
This version corresponds to the model file within which it is defined, and does not cover the whole set of OpenConfig models. Where several modules are used to build up a single block of functionality, the same module version is specified across each file that makes up the module.

A major version number of 0 indicates that this model is still in development (whether within OpenConfig or with industry partners), and is potentially subject to change.

Following a release of major version 1, all modules will increment major revision number where backwards incompatible changes to the model are made.

The minor version is changed when features are added to the model that do not impact current clients use of the model.

The patch-level version is incremented when non-feature changes (such as bugfixes or clarifications to human-readable descriptions that do not impact model functionality) are made that maintain backwards compatibility.

The version number is stored in the module meta-data."; 
} 
} 
<CODE ENDS>
9. References

9.1. Normative references

the Network Configuration Protocol (NETCONF)", RFC 6020,
DOI 10.17487/RFC6020, October 2010,

[RFC3688]  Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
DOI 10.17487/RFC3688, January 2004,

9.2. Informative references

[RTG-AD-YANG]  Wu, Q. and D. Sinicrope, "Routing Area Yang Coordinator’s
Summary Page", October 2015,
<http://trac.tools.ietf.org/area/rtg/trac/wiki/
RtgYangCoordSummary>.

[OC-SEMVER]  OpenConfig operator working group, "Semantic Versioning
for OpenConfig models", September 2015,
<http://www.openconfig.net/documentation/
semantic-versioning/>.

[I-D.openconfig-netmod-model-structure]  Shaikh, A., Shakir, R., D’Souza, K., and L. Fang,
"Operational Structure and Organization of YANG Models",
draft-openconfig-netmod-model-structure-00 (work in
progress), March 2015.

[I-D.rtgyangdt-rtgwg-device-model]  Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps,
"Network Device YANG Organizational Models",
draft-rtgyangdt-rtgwg-device-model-04 (work in
progress), May 2016.

Library", draft-ietf-netconf-yang-library-06 (work in
progress), April 2016.

[I-D.ietf-netmod-yang-model-classification]  Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module
Classification", draft-ietf-netmod-yang-model-
classification-02 (work in progress), June 2016.
Appendix A. Change summary

A.1. Changes between revisions -00 and -01


Authors' Addresses

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

Email: kd6913@att.com

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

Email: aashaikh@google.com

Rob Shakir
Jive Communications, Inc.
1275 West 1600 North, Suite 100
Orem, UT 84057

Email: rjs@rob.sh