

ALTO Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

X. Shi
Y. Yang
Yale University
M. Scharf
Alcatel-Lucent Bell Labs
October 27, 2014

A YANG Data Model for Base ALTO Data and Services
draft-shi-alto-yang-model-00

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol [RFC7285] defines a set of network information services, including the network-map service, the cost-map service, the filtered map services, the endpoint property service, and the endpoint cost service. A meta service, called the information resource directory (IRD) service, allows an ALTO server to provide ALTO clients with meta information (e.g., the access URI) about each resource and service it provides. [RFC7285] uses a RESTful design and encodes client request parameters and server responses using JSON. One may consider that most of these services are based on data maintained at an ALTO server. Hence, in this document, we explore how one may use the data modeling language YANG [RFC6020] to specify the services defined in [RFC7285]. We first define two YANG models for RPC specification and data instance description of of ALTO services. We then discuss the "standard operations" defined in NETCONF/RESTCONF to evaluate potential integration.

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 30, 2015.

Copyright Notice

Copyright (c) 2014 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	3
2. Design and Structure Overview of ALTO/YANG	4
2.1. Overview	4
2.2. ALTO/YANG using only RFC Specification	4
2.3. ALTO/YANG using Data Instance Description	7
2.4. ALTO/YANG with Integration with NETCONF/RESTCONF	12
3. Non-Trivial Specification Efforts	12
3.1. YANG Expressiveness Issues	12
3.1.1. Limitation on Modeling JSON key-value store	12
3.1.2. Limits of Leafrefs	15
3.2. Extensibility Issues	15
3.2.1. String Pattern Extensibility	15
3.2.2. Type Extensibility	15
4. Applications of ALTO/YANG	15
4.1. Other Applications	15
4.1.1. Verifier/Validator on the Client Side	15
4.1.2. Code Generator	15
5. Security Considerations	16
6. IANA Considerations	16
7. References	16
Appendix A. YANG Data Model for ALTO Protocol	17
A.1. Revision 1: RPC Only	17
A.2. Revision 2: Custom Data Instances	35
Appendix B. YANG-Validated JSON Messages for ALTO Examples	55
B.1. IRD Response Example	55
B.2. Network Map Service Response Example	59
B.3. Filtered Cost Map Response Example	61
B.4. Endpoint Property Service Response Example	63
Authors' Addresses	65

1. Introduction

This document explores how one may use the data modeling language YANG [RFC6020] to specify the network information services defined in the Application-Layer Traffic Optimization (ALTO) protocol [RFC7285]. In particular, this work is motivated by the recent substantial interest in the networking community to use YANG in networking protocol specification and development. This interest is highly justified, given that the potential benefits of using a formal specification can be multifold, including both precision and the potential of automation. However, at the same time, data modeling languages are generally domain specific languages (DSL) and hence are restricted by their specific domains. For example, YANG is developed in the specific domain of modeling configuration and state data stored at network devices. ALTO information does not fit in this domain but is highly related: ALTO information should be derived from network configuration and state data. Hence, modeling ALTO using YANG provides an interesting exercise on how one may apply YANG slightly outside its original domain.

The initial goal of this document was to produce an ALTO specification using YANG so that the YANG specification of ALTO, which we refer to as ALTO/YANG, and the original specification [RFC7285] can inter-operate. We refer to an interop specification as a syntactically equivalent specification. In other words, the ALTO/YANG specification produces the same behavior as that of the original specification. Unfortunately, as we have shown in [draft-shi-alto-yang-json-00], even without the need to consider other components such as NETCONF/RESTCONF to construct a complete YANG system, the basic encoding rule of YANG already makes syntactic equivalence unfeasible. In particular, [RFC7285] makes extensive use of the common key-value store abstraction in its JSON object encoding. But such encoding cannot be generated using the existing YANG/JSON encoding. As a result, the focus of this document is on using YANG to define semantically equivalent ALTO services. We still strive that the syntax generated by ALTO/YANG is close to that of [RFC7285].

The rest of this document is organized as follows. In Section 2, we provide an overview of our design approaches. We discuss three specification approaches, considering the three use cases of YANG. In Section 3, we discuss more detailed specification issues that we have encountered. The detailed specification of ALTO/YANG is in Appendix A.

2. Design and Structure Overview of ALTO/YANG

2.1. Overview

Our design tries to use YANG in three aspects: (1) RPC specification, (2) data instance description, and (3) standard operations. These three aspects are not mutually exclusive of each other, but assume different levels of matching and use of YANG.

The RPC specification aspect allows precise specification and automation of RPC input/output data serialization and deserialization. This aspect of using YANG can be considered the simplest. The protocol to be specified does not need to match the operational model of YANG. In Section Section 2.2, we specify a YANG model named `alto-service-rpc` that uses YANG only in this aspect.

The data instance description and standard operations aspects are analogous to databases. In relational database, one uses SQL to describe the data schema (e.g., in `CREATE TABLE`). We refer to this process of modeling the data as the data instance description aspect. Modeling the data in YANG does not guarantee full compatibility with the operational model of the existing protocol. In particular, the data instances specified in a YANG model may be virtual in a protocol design. In Section Section 2.3, we specify a YANG model named `alto-service-did` that uses YANG to specify both RPCs and potential ALTO base data instances to implement certain services.

SQL also defines statements such as `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. With a DBMS implementing SQL, when a user has a new set of data to be managed, the user describes the new data schema. The existing statements are already available to retrieve and manipulate the data. This substantially reduce the efforts to introduce and manage a new set of data. We refer to this as the standard operations aspect. In YANG related efforts, `NETCONF` and `RESTCONF` each tries to provide some of these standard operations (e.g., `RESTCONF` query). To utilize this aspect, the YANG specification need to be "embedded" in the chosen specification (e.g., `NETCONF`). In Section Section 2.4, we discuss specification by considering that the YANG model has the support of "standard operations" defined in `RESTCONF`. This document does not provide a specific specification, and will do so in the next version.

2.2. ALTO/YANG using only RFC Specification

Figure 1 gives the tree diagram of a YANG model named `alto-service-rpc` to specify ALTO services. As it is clear from the specification, it has only RPCs, which are supported by typedef and grouping not

shown in the tree diagram. Such a model is still valuable in that it clarifies the input and output data format of ALTO services.

```
module: alto-service-rpc
rpcs:
```

```

+---x IRD-service
|   +--ro output
|   |   +--ro IRD-service
|   |   |   +--ro meta
|   |   |   |   +--ro cost-types* [cost-type-name]
|   |   |   |   |   +--ro cost-type-name    cost-type-name
|   |   |   |   |   +--ro cost-mode        cost-mode
|   |   |   |   |   +--ro cost-metric      cost-metric
|   |   |   |   |   +--ro description?     string
|   |   |   |   +--ro default-alto-network-map resource-id
|   |   +--ro resources* [resource-id]
|   |   |   +--ro resource-id    resource-id
|   |   |   +--ro uri            inet:uri
|   |   |   +--ro media-type     media-type
|   |   |   +--ro accepts*       media-type
|   |   |   +--ro capabilities
|   |   |   |   +--ro cost-constraints?    boolean
|   |   |   |   +--ro cost-type-names*     cost-type-name
|   |   |   |   +--ro prop-types*          endpoint-property-type
|   |   +--ro uses*                    resource-id
+---x network-map-service
|   +--ro input
|   |   +--ro uri    inet:uri
|   +--ro output
|   |   +--ro network-map-service
|   |   |   +--ro meta
|   |   |   |   +--ro vtag
|   |   |   |   |   +--ro resource-id    resource-id
|   |   |   |   |   +--ro tag            string
|   |   +--ro network-map* [pid]
|   |   |   +--ro pid                    pid-name
|   |   |   +--ro endpoint-address-group* [address-type]
|   |   |   |   +--ro address-type        endpoint-address-type
|   |   |   |   +--ro endpoint-prefix*    endpoint-prefix
+---x cost-map-service
|   +--ro input
|   |   +--ro uri    inet:uri
|   +--ro output
|   |   +--ro cost-map-service
|   |   |   +--ro meta
|   |   |   |   +--ro dependent-vtags*
|   |   |   |   |   +--ro resource-id    resource-id
|   |   |   |   |   +--ro tag            string
```

```

|         |         +---ro cost-type
|         |         |         +---ro cost-mode         cost-mode
|         |         |         +---ro cost-metric        cost-metric
|         |         |         +---ro description?       string
+---ro cost-map* [src]
|         |         +---ro src         pid-name
|         |         +---ro dst-costs* [dst]
|         |         |         +---ro dst         pid-name
|         |         |         +---ro cost
+---x filtered-network-map-service
+---ro input
|         |         +---ro pids*         pid-name
|         |         +---ro address-types* endpoint-address-type
+---ro output
+---ro filtered-network-map-service
+---ro meta
|         |         +---ro vtag
|         |         |         +---ro resource-id     resource-id
|         |         |         +---ro tag             string
+---ro network-map* [pid]
|         |         +---ro pid         pid-name
|         |         +---ro endpoint-address-group* [address-type]
|         |         |         +---ro address-type     endpoint-address-type
|         |         |         +---ro endpoint-prefix*  endpoint-prefix
+---x filtered-cost-map-service
+---ro input
|         |         +---ro cost-type
|         |         |         +---ro cost-mode         cost-mode
|         |         |         +---ro cost-metric        cost-metric
|         |         |         +---ro description?       string
+---ro constraints* constraint
+---ro pids
|         |         +---ro srcs*        pid-name
|         |         +---ro dsts*        pid-name
+---ro output
+---ro filtered-cost-map-service
+---ro meta
|         |         +---ro dependent-vtags*
|         |         |         +---ro resource-id     resource-id
|         |         |         +---ro tag             string
+---ro cost-type
|         |         |         +---ro cost-mode         cost-mode
|         |         |         +---ro cost-metric        cost-metric
|         |         |         +---ro description?       string
+---ro cost-map* [src]
|         |         +---ro src         pid-name
|         |         +---ro dst-costs* [dst]
|         |         |         +---ro dst         pid-name

```

```

|           +--ro cost
+---x endpoint-property-service
|   +--ro input
|   |   +--ro properties*   endpoint-property-type
|   |   +--ro endpoints*   typed-endpoint-address
|   +--ro output
|   |   +--ro endpoint-property-service
|   |   |   +--ro meta
|   |   |   |   +--ro dependent-vtags*
|   |   |   |   |   +--ro resource-id   resource-id
|   |   |   |   |   +--ro tag           string
|   |   |   +--ro endpoint-properties* [endpoint]
|   |   |   |   +--ro endpoint         typed-endpoint-address
|   |   |   |   +--ro properties* [property-type]
|   |   |   |   |   +--ro property-type   endpoint-property-type
|   |   |   |   |   +--ro property       endpoint-property-value
+---x endpoint-cost-service
|   +--ro input
|   |   +--ro cost-type
|   |   |   +--ro cost-mode       cost-mode
|   |   |   +--ro cost-metric     cost-metric
|   |   |   +--ro description?    string
|   |   +--ro constraints*       constraint
|   |   +--ro endpoints
|   |   |   +--ro srcs*           typed-endpoint-address
|   |   |   +--ro dsts*          typed-endpoint-address
+--ro output
|   +--ro endpoint-cost-service
|   |   +--ro meta
|   |   |   +--ro cost-type
|   |   |   |   +--ro cost-mode       cost-mode
|   |   |   |   +--ro cost-metric     cost-metric
|   |   |   |   +--ro description?    string
|   |   +--ro endpoint-cost-map* [src]
|   |   |   +--ro src               typed-endpoint-address
|   |   |   +--ro dst-costs* [dst]
|   |   |   |   +--ro dst           typed-endpoint-address
|   |   |   |   +--ro cost

```

2.3. ALTO/YANG using Data Instance Description

Figure 2 shows the tree diagram of a second YANG model for ALTO, named `alto-service-did`. Comparing Figure 1 with Figure 2, one can observe that a key difference is that the second model first introduces data instances, listed under resources, that an ALTO server may maintain. We use the word "may" because an ALTO server may not physically store such data. For example, a cost map could be generated on the fly, and hence can be in a sense a virtual data

table. One may observe from our specified instances that we do not specify a data instance to support the endpoint cost service (ECS), as this can be highly inefficient. For ECS, the value of YANG is mainly in RPC specification.

```

module: alto-service-did
  +--ro resources
    +--ro IRD
      +--ro meta
        +--ro cost-types* [cost-type-name]
          +--ro cost-type-name  cost-type-name
          +--ro cost-mode       cost-mode
          +--ro cost-metric     cost-metric
          +--ro description?    string
        +--ro default-alto-network-map  leafref
      +--ro resources* [resource-id]
        +--ro resource-id      resource-id
        +--ro uri              inet:uri
        +--ro media-type       media-type
        +--ro accepts*         media-type
        +--ro capabilities
          +--ro cost-constraints?  boolean
          +--ro cost-type-names*   cost-type-name
          +--ro prop-types*        endpoint-property-type
        +--ro uses*             leafref
    +--ro network-maps*
      +--ro meta
        +--ro vtag
          +--ro resource-id      resource-id
          +--ro tag              string
        +--ro network-map* [pid]
          +--ro pid              pid-name
          +--ro endpoint-address-group* [address-type]
            +--ro address-type   endpoint-address-type
            +--ro endpoint-prefix* endpoint-prefix
    +--ro cost-maps*
      +--ro meta
        +--ro dependent-vtags*
          +--ro resource-id      resource-id
          +--ro tag              string
        +--ro cost-type
          +--ro cost-mode       cost-mode
          +--ro cost-metric     cost-metric
          +--ro description?    string
        +--ro cost-map* [src]
          +--ro src              pid-name
          +--ro dst-costs* [dst]
            +--ro dst            pid-name

```



```

|         +--ro cost
+--ro endpoint-property-map
|   +--ro meta
|   |   +--ro dependent-vtags*
|   |   |   +--ro resource-id    resource-id
|   |   |   +--ro tag            string
|   +--ro endpoint-properties* [endpoint]
|   |   +--ro endpoint          typed-endpoint-address
|   |   +--ro properties* [property-type]
|   |   |   +--ro property-type    endpoint-property-type
|   |   |   +--ro property        endpoint-property-value
rpcs:
+---x IRD-service
|   +--ro output
|   |   +--ro IRD-service
|   |   |   +--ro meta
|   |   |   |   +--ro cost-types* [cost-type-name]
|   |   |   |   |   +--ro cost-type-name    cost-type-name
|   |   |   |   |   +--ro cost-mode        cost-mode
|   |   |   |   |   +--ro cost-metric      cost-metric
|   |   |   |   |   +--ro description?     string
|   |   |   |   +--ro default-alto-network-map    leafref
|   |   +--ro resources* [resource-id]
|   |   |   +--ro resource-id    resource-id
|   |   |   +--ro uri            inet:uri
|   |   |   +--ro media-type     media-type
|   |   |   +--ro accepts*       media-type
|   |   |   +--ro capabilities
|   |   |   |   +--ro cost-constraints?    boolean
|   |   |   |   +--ro cost-type-names*    cost-type-name
|   |   |   |   +--ro prop-types*         endpoint-property-type
|   |   |   +--ro uses*           leafref
+---x network-map-service
|   +--ro input
|   |   +--ro uri    leafref
|   +--ro output
|   |   +--ro network-map-service
|   |   |   +--ro meta
|   |   |   |   +--ro vtag
|   |   |   |   |   +--ro resource-id    resource-id
|   |   |   |   |   +--ro tag            string
|   |   +--ro network-map* [pid]
|   |   |   +--ro pid                pid-name
|   |   |   +--ro endpoint-address-group* [address-type]
|   |   |   |   +--ro address-type    endpoint-address-type
|   |   |   |   +--ro endpoint-prefix* endpoint-prefix
+---x cost-map-service
|   +--ro input

```

```

|   |--ro uri      leafref
+--ro output
  |--ro cost-map-service
    |--ro meta
      |--ro dependent-vtags*
      |   |--ro resource-id    resource-id
      |   |--ro tag            string
      |--ro cost-type
        |--ro cost-mode        cost-mode
        |--ro cost-metric      cost-metric
        |--ro description?     string
    +--ro cost-map* [src]
      |--ro src                pid-name
      +--ro dst-costs* [dst]
        |--ro dst              pid-name
        +--ro cost
+---x filtered-network-map-service
  +--ro input
    |--ro pids*                pid-name
    |--ro address-types*       endpoint-address-type
  +--ro output
    +--ro filtered-network-map-service
      |--ro meta
        |--ro vtag
          |--ro resource-id    resource-id
          |--ro tag            string
      +--ro network-map* [pid]
        |--ro pid              pid-name
        +--ro endpoint-address-group* [address-type]
          |--ro address-type    endpoint-address-type
          +--ro endpoint-prefix* endpoint-prefix
+---x filtered-cost-map-service
  +--ro input
    |--ro cost-type
      |--ro cost-mode          cost-mode
      |--ro cost-metric        cost-metric
      |--ro description?       string
    +--ro constraints*         constraint
  +--ro pids
    |--ro srcs*                pid-name
    |--ro dsts*                pid-name
  +--ro output
    +--ro filtered-cost-map-service
      |--ro meta
        |--ro dependent-vtags*
        |   |--ro resource-id    resource-id
        |   |--ro tag            string
        +--ro cost-type

```

```

|         |         +---ro cost-mode      cost-mode
|         |         +---ro cost-metric    cost-metric
|         |         +---ro description?   string
|         +---ro cost-map* [src]
|         |         +---ro src            pid-name
|         |         +---ro dst-costs* [dst]
|         |         +---ro dst            pid-name
|         |         +---ro cost
+---x endpoint-property-service
| +---ro input
| | +---ro properties*   endpoint-property-type
| | +---ro endpoints*   typed-endpoint-address
+---ro output
| +---ro endpoint-property-service
| | +---ro meta
| | | +---ro dependent-vtags*
| | | | +---ro resource-id    resource-id
| | | | +---ro tag            string
| | +---ro endpoint-properties* [endpoint]
| | | +---ro endpoint        typed-endpoint-address
| | | +---ro properties* [property-type]
| | | | +---ro property-type  endpoint-property-type
| | | | +---ro property      endpoint-property-value
+---x endpoint-cost-service
| +---ro input
| | +---ro cost-type
| | | +---ro cost-mode      cost-mode
| | | +---ro cost-metric    cost-metric
| | | +---ro description?   string
| | +---ro constraints*    constraint
| | +---ro endpoints
| | | +---ro srcs*         typed-endpoint-address
| | | +---ro dsts*         typed-endpoint-address
+---ro output
| +---ro endpoint-cost-service
| | +---ro meta
| | | +---ro cost-type
| | | | +---ro cost-mode      cost-mode
| | | | +---ro cost-metric    cost-metric
| | | | +---ro description?   string
| | +---ro endpoint-cost-map* [src]
| | | +---ro src            typed-endpoint-address
| | | +---ro dst-costs* [dst]
| | | | +---ro dst          typed-endpoint-address
| | | | +---ro cost

```

2.4. ALTO/YANG with Integration with NETCONF/RESTCONF

Our next model considers standard operations. In other words, if an RPC can be implemented using a standard operation, the model may use the standard operation and not specify the RPC.

Consider NETCONF. One may use the NETCONF <get> operation to retrieve the full ALTO network map, cost map, and the Information Resource Directory; with the <filter> parameter, NETCONF may retrieve filtered maps as well as endpoint properties. Hence, one may omit the corresponding RPCs defined in alto-service-did. The exact encoding of using "standard operations", however, can be less compact.

3. Non-Trivial Specification Efforts

3.1. YANG Expressiveness Issues

3.1.1. Limitation on Modeling JSON key-value store

ALTO is a JSON based protocol and makes extensive use of JSON key-value store, which is useful because of its efficiency, inherent uniqueness constraint on the keys, and its natural correspondence to and from structures such as indexed database tables or hashmaps.

For example, the network map is defined as mapping from a PID to an endpoint address group. Here is an example network map in Section 11.2.1.7 of [RFC7285].

```

"network-map" : {
  "PID1" : {
    "ipv4" : [
      "192.0.2.0/24",
      "198.51.100.0/25"
    ]
  },
  "PID2" : {
    "ipv4" : [
      "198.51.100.128/25"
    ]
  },
  "PID3" : {
    "ipv4" : [
      "0.0.0.0/0"
    ],
    "ipv6" : [
      "::/0"
    ]
  }
}

```

As pointed out in [draft-shi-alto-yang-json-00], such JSON objects cannot be modeled in YANG. To achieve the semantical equivalence, we took the approach of modeling it as a "list" with a unique index ("key") in YANG.

```

list network-map {
  key "pid";
  leaf pid {
    type string;
  }
  list endpoint-address-group {
    key address-type;
    leaf address-type {
      type endpoint-address-type;
    }
    leaf-list endpoint-prefix {
      type endpoint-prefix;
    }
  }
}

```

According to [draft-ietf-netmod-yang-json-01], the above YANG model correspond to the following JSON text:

```

"network-map": [
  {
    "pid": "PID1",
    "endpoint-address-group": {
      "address-type": "ipv4",
      "endpoint-prefix": [
        "192.0.2.0/24",
        "198.51.100.0/25"
      ]
    }
  },
  {
    "pid": "PID2",
    "endpoint-address-group": {
      "address-type": "ipv4",
      "endpoint-prefix": ["198.51.100.128/25"]
    }
  },
  {
    "pid": "PID3",
    "endpoint-address-group": [
      {
        "address-type": "ipv4",
        "endpoint-prefix": ["0.0.0.0/0"]
      },
      {
        "address-type": "ipv6",
        "endpoint-prefix": [ "::/0" ]
      }
    ]
  }
]

```

We immediately notice a few disadvantages.

- (1) From the YANG validated JSON message alone, it is unclear that the "pid" field is the key to the list "network-map";
- (2) The YANG-validated JSON message is much more verbose, which increases the payload, especially when the model scales.
- (3) The consistency between address-type and endpoint-prefix is not enforced in the above YANG model.

3.1.2. Limits of Leafrefs

Leafrefs in YANG are heavily tied with XML and XPATH expressions. However, leafrefs cannot refer to an rpc node (or structures within a rpc node). For example, referring to the input parameters in rpc input from rpc output could be very useful.

3.2. Extensibility Issues

3.2.1. String Pattern Extensibility

For string types that have inherant relationships (e.g., extension, concatenation), it is useful to be able to extend the patterns of the strings for modularity and extensibility. For example, TypedEndpointAddress in [RFC7285] is defined as a string of the format AddressType, followed by the ':' separator, followed by a ip-address EndpointAddress. Definitions of the pattern of an ip-address is readily available in Common YANG Data Types [RFC6991]. It is not possible in YANG to inherit, extend, or refer to a pattern of a different string type. Another example is the resource-specific endpoint property, which is defined as a resource ID, followed by the '.' separator (U+002E), followed by a name obeying the same rules as for global endpoint property names.

3.2.2. Type Extensibility

YANG is unable to augment a typedef or enum type. Hence there is not a good way to add a cost metric or type of address if the model is fixed.

4. Applications of ALTO/YANG

4.1. Other Applications

4.1.1. Verifier/Validator on the Client Side

Using tools like pyang (<https://code.google.com/p/pyang/>), we can not only validate the YANG module, but also translate the module to DSDL schema and validate instance documents. The limitation is that pyang can only validate XML files and the JSON-XML translation is not well-defined. We provide a few examples of pyang-validated ALTO messages in the appendix.

4.1.2. Code Generator

OpenDayLight (ODL) provides automatic code generation for YANG models. ODL yangtools generates a Java OSGi bundle for a given YANG module, including typedefs and groupings, data instances, and RPCs.

One may integrate this bundle in the ODL controller which provides JSON parser and a RESTful API for the YANG RPCs. The datastore is automatically created and managed by ODL as well. The only manual code needed are the bundle activators and the server computation implementation.

5. Security Considerations

This document does not introduce security or privacy concerns.

6. IANA Considerations

This document does not have IANA considerations.

7. References

- [RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2010.
- [RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6020, October 2010.
- [draft-ietf-netmod-yang-json-01]
 Lhotka, L., "JSON Encoding of Data Modeled with YANG", October 2014.
- [draft-shi-alto-yang-json-00]
 Shi, X. and Y. Yang, "Modeling JSON Messages Using YANG", October 2014.
- [RESTCONF]
 Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", October 2014.

Appendix A. YANG Data Model for ALTO Protocol

A.1. Revision 1: RPC Only

```

module alto-service-rpc {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:alto-service";
  // TODO: replace with IANA namespace when assigned

  prefix "as";

  import ietf-inet-types {
    prefix inet;
  }

  organization "ALTO WG";
  contact "alto@ietf.org";

  description
    "This module defines a semantically equivalent data model
    for the ALTO services defined in RFC7285.";

  /* Potential issue for future EXTENSION: YANG 1.0 is unable to
   * augment a typedef or enum type. e.g., there is not a good way to
   * add a cost metric or type of address if this document is
   * fixed. We have marked such places with "EXTENSION." */

  revision 2014-10-24 {
    description "Initial revision: RPC only.";
  }

  /*****
   * TYPE DEFINITIONS *
   *****/

  /*****
   Definitions for addresses

   ALTO RFC7285 uses the following addresses, as shown in the
   examples below:

   - Endpoint property service (Sec. 11.4.1.7):
     "endpoints" : [ "ipv4:192.0.2.34",
                     "ipv4:203.0.113.129" ]
   - Endpoint cost service (Sec. 11.5.1.7):
     "endpoints" : {
       "srcs": [ "ipv4:192.0.2.2" ],

```

```

    "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv4:203.0.113.45"
    ]
- Network map (Sec. 11.2.1.7.):
    "ipv4": [
        "192.0.2.0/24",
        "198.51.100.0/25"
    ],
    "ipv6": [
        "2001:db8:0:1::/64",
        "2001:db8:0:2::/64"
    ]

```

To handle the proceeding, we need the following definitions:

- ipv4-address (e.g., 192.0.2.0, already defined in rfc6991),
- ipv6-address (already defined in rfc6991),
- ipv4-prefix (e.g., 192.0.2.0/24, already defined in rfc6991),
- ipv6-prefix (defined in rfc6991),
- typed-ipv4-address (e.g., ipv4:192.0.2.1, to be defined below)
- typed-ipv6-address
- typed-ipv4-prefix-list (e.g., "ipv4": [
 - "192.0.2.0/24",
 - "198.51.100.0/25"
],

*****/

/*

First define typed-ipv4-address and typed-ipv6-address, as used by endpoint services.

The ideal case is to define it as "ipv4:"+ipv4-address, but there is not such a type constructor (YANG EXTENSION). Hence, the current definition cuts-and-pastes (i.e., repeats verbatim) the definition of ipv4-address and prepend "ipv4:". The downside is that if someone redefines ipv4-address, there could be inconsistency.

*/

```

typedef typed-ipv4-address {
    type string {
        pattern
            'ipv4:(([0-9]|[1-9][0-9]|1[0-9][0-9]|'
            + '2[0-4][0-9]|25[0-5])\.){3}'
            + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
            + '(%[\p{N}\p{L}]+)?';
    }
}

```

```

}

typedef typed-ipv6-address {
  type string {
    pattern 'ipv6:((:|([0-9a-fA-F]{0,4})))([0-9a-fA-F]{0,4}:){0,5}'
    + '((((([0-9a-fA-F]{0,4})?:)?(:|([0-9a-fA-F]{0,4})))|'
    + '(((25[0-5]|2[0-4][0-9]|01?[0-9]?[0-9])\.){3}'
    + '(25[0-5]|2[0-4][0-9]|01?[0-9]?[0-9])))'
    + '(%[\p{N}\p{L}]+)?';
    pattern 'ipv6:(([^\:]+\:){6}([^\:]+\:|(\.\.\.)))|'
    + '(((([^\:]+\:)*[^\:]+)?\:([^\:]+\:)*[^\:]+)?'
    + '(%\.+)?';
  }
}

typedef typed-endpoint-address {
  type union {
    type typed-ipv4-address;
    type typed-ipv6-address;
    // EXTENSION: ADD NEW TYPE HERE.
  }
  description
    "Ref: RFC7285 Sec. 10.4.1 Typed Endpoint Addresses" +
    "= AddressType:EndpointAddr";
}

```

/* Next, we define endpoint address group, as used in the definition of ALTO network maps. Specifically, an endpoint address group in ALTO is defined as a key-value store, with address type as key, and an array of prefix as the value of each key:

```

EndpointAddrGroup. RFC7285 Sec. 10.4.5." +
  object-map {
    AddressType -> endpoint-prefix<0..*>;
  } EndpointAddrGroup;

```

There are two challenges:

1) To specify that AddressType is key, we must use the list type, which is the only type that one can specify key. However, the current JSON-YANG encoding generates an array, instead of a key-value map;

2) Ideally, we want to enforce address type and prefix consistency; for example, an ipv6 prefix in an ipv4 type should not be allowed. However, we encounter problems. We leave this as an OPEN ISSUE.

```

*/

typedef endpoint-address-type {
  type union {
    type enumeration {
      enum ipv4;
      enum ipv6;
      // EXTENSION: ADD NEW TYPE HERE
    }
  }
  description
    "Ref: RFC7285 Sec 2.2.";
}

typedef endpoint-prefix {
  type inet:ip-prefix;
  description
    "endpoint prefix, identical to ip-prefix defined in RFC6991.";
}

grouping endpoint-address-group {
  list endpoint-address-group {
    key address-type;
    leaf address-type {
      type endpoint-address-type;
      mandatory true;
    }
    leaf-list endpoint-prefix {
      type endpoint-prefix;
    }
  }
  description
    "EndpointAddrGroup. RFC7285 Sec. 10.4.5." +
    " object-map {
      AddressType -> endpoint-prefix<0..*>;
    } EndpointAddrGroup;";
}

/*****
* Definitions for IDs and names
*
* ALTO defines the following concepts that are names and IDs:
*
*   pid name (used in network map, cost map),
*   resource IDs (used to identify alto network/cost maps),
*   version tag (used to indicate uniqueness of resource),
*   cost-type-name (used in IRD),
*   cost-metric,

```

```

*    cost-mode
*
* We group their definitions together below.
*****/

typedef valid-id-string {
  type string {
    length "1..64";
    pattern "[0-9a-zA-Z_\\-:@\\.]+";
  }
  description
    "Type for valid ID strings.";
}

typedef pid-name {
  type valid-id-string;
  description
    "Name for the PID." +
    "RFC7285, Section 10.1. Note: the '.' separator MUST NOT be" +
    "used unless specifically indicated in RFC7285 or an" +
    " extension document.";
}

typedef resource-id {
  type valid-id-string;
  description
    "Resource-ID.";
}

grouping vtag {
  leaf resource-id {
    type resource-id;
    mandatory true;
  }
  leaf tag {
    type string {
      length "1..64";
      pattern "[!-~]+";
    }
    mandatory true;
    description
      "Tag. RFC7285 Sec. 10.3. U+0021-U+007E";
  }
  description
    "Version tag. Both resource-id and tag must be equal  

    byte-for-byte. RFC7285 Sec. 10.3." +
    " object {
      ResourceID resource-id;

```

```

        JSONString tag;
    } VersionTag;";
}

grouping dependent-vtags {
    list dependent-vtags {
        uses vtag;
        min-elements 1;
    }
}

/*****
Definitions for cost type and cost types

In ALTO, a cost type consists of two required components:

    cost-metric,
    cost-mode
    and an optional description component.

In the IRD, one can name each cost type. Such info is collected
in a hash map called cost types.
*****/

typedef cost-metric {
    type union {
        type enumeration {
            enum routingcost {
                description
                "Default metric. MUST support. RFC7285 Sec. 6.1.1.1.";
            }
            enum hopcount {
                description
                "Hopcount metric.";
            }
        }
        // EXTENSION: Additional cost-metric will be defined here.
    }
    type string {
        length 1..32;
        pattern "priv:[0-9a-zA-Z_\\-:\\.]+";
    }
}
description
    "Cost metric. for type string,
    'priv:' reserved for Private Use.";
}

typedef cost-mode {

```

```

    type enumeration {
        enum numerical {
            description
                "Numerical cost mode.";
        }
        enum ordinal {
            description
                "Ordinal cost mode.";
        }
        // EXTENSION: Additional cost-mode will be defined here.
    }
    description
        "Cost mode. MUST support at least one of numerical and ordinal";
}

grouping cost-type {
    leaf cost-mode {
        type cost-mode;
        mandatory true;
        description
            "Cost mode.";
    }
    leaf cost-metric {
        type cost-metric;
        mandatory true;
        description
            "Cost metric.";
    }
    leaf description {
        type string;
        description
            "Optional description field.";
    }
    description
        "Cost type. RFC7285 Sec. 10.7." +
        " object {
            CostMetric cost-metric;
            CostMode  cost-mode;
            [JSONString description;]
        } CostType;";
}

typedef cost-type-name {
    type valid-id-string;
    // NOTE: not fully specified in RFC7285, default as valid id
}

grouping cost-types {

```

```

    list cost-types {
        key cost-type-name;
        leaf cost-type-name {
            type cost-type-name;
        }
        uses cost-type;
    }
    description
        "RFC 7285 Sec. 9.2.2." +
        "object-map {
            JSONString -> CostType;
        } IRDMetaCostTypes;";
}

/*****
* Definitions for endpoint properties *
*****/
typedef global-endpoint-property {
    type union {
        type enumeration {
            enum pid {
                description "PID property.";
            }
            // EXTENSION: other options here
        }
        type string {
            pattern "priv:[\w\-\:@]+";
        }
    }
    description
        "Global endpoint property. RFC7285 Sec. 10.8.2." +
        "'priv:' for Private Use " +
        "length 1..32; '.' is not allowed";
}

/*
* Ideally we would want to extend the typedef of resource-id and
* global endpoint properties, however, YANG 1.0 does not allow
* that, hence we simply copied the regex for resource-id over
* verbatim.
*/

typedef resource-specific-endpoint-property {
    type string {
        length "3..97"; //len(resource-id) + 1 + len(global-property)
        pattern "(priv:)?[\w\-\:@\.\.]+\.[\w\-\:_]+"; // resource-id.property
    }
}

```



```

    description
        "Resource-specific endpoint property.";
}

typedef endpoint-property-type {
    type union {
        type resource-specific-endpoint-property;
        type global-endpoint-property;
    }
    description
        "Endpoint property type. RFC7285 Sec. 10.8.";
}

typedef endpoint-property-value {
    type string;
    description
        "Endpoint property (value).";
}

/*****
* Definitions for response header
*****/

typedef media-type {
    type union {
        type string {
            pattern "application/alto\-.*";
        }
        type enumeration {
            enum alto-directory+json;
            enum alto-networkmap+json;
            enum alto-networkmapfilter+json;
            enum alto-costmap+json;
            enum alto-costmapfilter+json;
            enum alto-endpointprop+json;
            enum alto-endpointpropparams+json;
            enum alto-endpointcost+json;
            enum alto-endpointcostparams+json;
            enum alto-error+json;
        }
    }
}

grouping alto-cost {
    anyxml cost {
        mandatory true;
        description
            "ALTO cost is a JSONValue, which could be

```

```

        an object, array, string, etc. (Ref: RFC 7159 Sec.3.)";
    }
}

typedef constraint {
    type string {
        pattern "(gt|ge|lt|le|eq) [0-9]+";
    }
    description
        "RFC7285 Sec. 11.3.2.3. The second part must be in the" +
        "same unit as cost-metric, IEEE 754 2008 floating point.";
}

/*****
Groupings for ALTO information resource
*****/

/* meta */
grouping IRD-meta {
    uses cost-types;
    leaf default-alto-network-map {
        type resource-id;
        mandatory true;
    }
}

grouping network-map-meta {
    container vtag {
        uses vtag;
    }
}

grouping cost-map-meta {
    uses dependent-vtags {
        refine dependent-vtags {
            max-elements 1;
        }
    }
    container cost-type {
        uses cost-type;
    }
}

grouping endpoint-property-meta {
    uses dependent-vtags;
}

/* accepts (optional) */

```

```

grouping accepts {
  leaf-list accepts {
    type media-type;
    min-elements 1;
  }
}

/* capabilities (capabilities) */
grouping IRD-capabilities {
  container capabilities {
    leaf cost-constraints {
      type boolean;
    }
    leaf-list cost-type-names {
      type cost-type-name;
    }
    leaf-list prop-types {
      type endpoint-property-type;
    }
  }
}

/* uses (optional) */
grouping uses {
  leaf-list uses {
    type resource-id;
    min-elements 1;
  }
}

/* Information Resource Directory Grouping */
grouping IRD {
  container meta {
    uses IRD-meta;
  }
  uses IRD-data;
}

grouping IRD-data {
  list resources {
    key resource-id;
    leaf resource-id {
      type resource-id;
      mandatory true;
    }
    leaf uri {
      type inet:uri;
      mandatory true;
    }
  }
}

```

```

    }
    leaf media-type {
        type media-type;
        mandatory true;
    }
    uses accepts {
        when "current()";
    }
    uses IRD-capabilities {
        when "current()";
    }
    uses uses {
        when "current()";
    }
    description
        "IRDRResourceEntry. RFC7285 9.2.2." +
        " object {
            JSONString      uri;
            JSONString      media-type;
            [JSONString      accepts;]
            [Capabilities    capabilities;]
            [ResourceID      uses<0..*>;]
        } IRDRResourceEntry;" +
        "IRDRResourceEntries. RFC7285 9.2.2." +
        " object-map {
            ResourceID -> IRDRResourceEntry;
        } IRDRResourceEntries;" +
        "InformationResourceDirectory. RFC7285 9.2.2." +
        " object {
            IRDRResourceEntries resources;
        } InfoResourceDirectory : ResponseEntityBase;";
    }
}

/* Network Map Grouping */
grouping network-map {
    container meta {
        uses network-map-meta;
    }
    uses network-map-data;
}

grouping network-map-data {
    list network-map {
        key "pid";
        leaf pid {
            type pid-name;
        }
    }
}

```

```

    uses endpoint-address-group;
    description
        "RFC7285 Sec. 11.2.1.6." +
        " object-map {
            PIDName -> EndpointAddrGroup;
        } NetworkMapData;";
    }
    description
        "Network map. RFC7285 Sec. 11.2.1.6." +
        "object {
            NetworkMapData network-map;
        } InfoResourceNetworkMap : ResponseEntityBase;";
    }

/* Cost Map Grouping */
grouping cost-map {
    container meta {
        uses cost-map-meta;
    }
    uses cost-map-data;
}

grouping cost-map-data {
    list cost-map {
        leaf src {
            type pid-name;
            description
                "Source PID.";
        }
        key "src";
        list dst-costs {
            leaf dst {
                type pid-name;
                description
                    "Destination PID.";
            }
            key "dst";
            uses alto-cost {
                description
                    "Cost from source to destination.";
            }
        }
        description
            "The list represents the inner part of the cost matrix." +
            "DstCosts. RFC7285 Sec. 11.2.3.6." +
            " object-map {
                PIDName -> JSONValue;
            } DstCosts;";
    }
}

```

```

    description
      "The list represents the outer part of the cost matrix." +
      "CostMapData. RFC7285 Sec. 11.2.3.6." +
      " object-map {
        PIDName -> DstCosts;
      } CostMapData;";
  }
  description
    "Cost map. RFC7285 Sec. 11.2.3.6." +
    " object {
      CostMapData cost-map;
    } InfoResourceCostMap : ResponseEntityBase;";
}

/* Endpoint Property Map Grouping */
grouping endpoint-property-map {
  container meta {
    uses endpoint-property-meta;
  }
  uses endpoint-property-map-data;
}

grouping endpoint-property-map-data {
  list endpoint-properties {
    key endpoint;
    leaf endpoint {
      type typed-endpoint-address;
      mandatory true;
    }
    list properties {
      key property-type;
      leaf property-type {
        type endpoint-property-type;
        mandatory true;
      }
      leaf property {
        type endpoint-property-value;
        mandatory true;
      }
    }
    description
      "EndpointProps. RFC7285 Sec. 11.4.1.6." +
      " object {
        EndpointPropertyType -> JSONValue;
      } EndpointProps;";
  }
  description
    "EndpointPropertyMapData. Sec. 11.4.1.6." +
    " object-map {

```

```

        TypedEndpointAddr -> EndpointProps;
    } EndpointPropertyMapData;";
}
description
    "InfoResourceEndpointProperties. Sec. 11.4.1.6." +
    " object {
        EndpointPropertyMapData endpoint-properties;
    } InfoResourceEndpointProperties : ResponseEntityBase;";
}

/*****
* RPCs
*****/

rpc IRD-service {
    output {
        container IRD-service {
            uses IRD;
        }
    }
}

rpc network-map-service {
    input {
        leaf uri {
            type inet:uri;
            mandatory true;
            description
                "The uri of the request for the full network map.";
        }
    }
    output {
        container network-map-service {
            uses network-map;
        }
    }
}

rpc cost-map-service {
    input {
        leaf uri {
            type inet:uri;
            mandatory true;
            description
                "The uri of the request for the full network map.";
        }
    }
    output {

```

```

        container cost-map-service {
            uses cost-map;
        }
    }
}

rpc filtered-network-map-service {
    description
        "inquiries on filtered network map" +
        "ReqFilteredNetworkMap. RFC7285 Sec. 11.3.1.3." +
        " object {
            PIDName pids<0..*>;
            [AddressType address-types<0..*>;]
        } ReqFilteredNetworkMap;";
    input {
        leaf-list pids {
            must "current()";
            type pid-name;
        }
        leaf-list address-types {
            type endpoint-address-type;
        }
    }
    output {
        container filtered-network-map-service {
            uses network-map;
        }
    }
}

rpc filtered-cost-map-service {
    input {
        container cost-type {
            must "current()";
            uses cost-type;
        }
        leaf-list constraints {
            type constraint;
            description
                "RFC7285 Sec. 11.3.2.3.";
        }
        container pids {
            leaf-list srcs {
                type pid-name;
                description
                    "Source endpoint addresses.";
            }
            leaf-list dsts {

```



```

        type pid-name;
        description
            "Destination endpoint addresses.";
    }
    description
        "PIDFilter: Endpoint addresses. RFC7285 Sec. 11.3.2.3." +
        " object {
            PIDName srcs<0..*>;
            PIDName dsts<0..*>;
        } PIDFilter;";
    }
}
output {
    container filtered-cost-map-service {
        uses cost-map;
    }
}

rpc endpoint-property-service {
    description
        "inquiries on properties of an endpoint" +
        " object {
            EndpointPropertyType    properties<1..*>;
            TypedEndpointAddr        endpoints<1..*>;
        } ReqEndpointProp;";
    input {
        leaf-list properties {
            type endpoint-property-type;
            min-elements 1;
        }
        leaf-list endpoints {
            type typed-endpoint-address;
            min-elements 1;
        }
    }
    output {
        container endpoint-property-service {
            uses endpoint-property-map;
        }
    }
}

rpc endpoint-cost-service {
    description
        "ReqEndpointCostMap. RFC7285 Sec. 11.5.1.3." +
        " object {

```

```

        CostType          cost-type;
        [JSONString       constraints<0..*>;]
        EndpointFilter    endpoints;
    } ReqEndpointCostMap;";
input {
    container cost-type {
        must "current()";
        uses cost-type;
    }
    leaf-list constraints {
        type constraint;
        description
            "RFC7285 Sec. 11.5.1.3.";
    }
    container endpoints {
        must "current()";
        leaf-list srcs {
            type typed-endpoint-address;
            description
                "Source endpoint addresses.";
        }
        leaf-list dsts {
            type typed-endpoint-address;
            description
                "Destination endpoint addresses.";
        }
    }
    description
        " EndpointFilter: Endpoint addr. RFC7285 Sec. 11.5.1.3." +
        " object {
            [TypedEndpointAddr srcs<0..*>;]
            [TypedEndpointAddr dsts<0..*>;]
        } EndpointFilter;";
}
}
output {
    container endpoint-cost-service {
        container meta {
            container cost-type {
                uses cost-type;
            }
        }
        list endpoint-cost-map {
            leaf src {
                type typed-endpoint-address;
                description
                    "Source endpoint address.";
            }
            key "src";
        }
    }
}

```

```

    list dst-costs {
      leaf dst {
        type typed-endpoint-address;
        description
          "Destination endpoint address.";
      }
      key "dst";
      uses alto-cost {
        description
          "Cost from source to destination.";
      }
      description
        "The list represents the inner part of the cost matrix." +
        "EndpointDstCosts. RFC7285 Sec. 11.5.1.6." +
        " object-map {
          TypedEndpointAddr -> JSONValue;
        } EndpointDstCosts;";
    }
    description
      "The list represents the outer part of the cost matrix." +
      "EndpointCostMapData. RFC7285 Sec. 11.5.1.6." +
      " object {
        EndpointCostMapData endpoint-cost-map;
      } InfoResourceEndpointCostMap : ResponseEntityBase;
      object-map {
        TypedEndpointAddr -> EndpointDstCosts;
      } EndpointCostMapData;";
  }
}
}
}
}
}
}
}

```

A.2. Revision 2: Custom Data Instances

```

module alto-service-did {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:alto-service";
  // TODO: replace with IANA namespace when assigned

  prefix "as";

  import ietf-inet-types {
    prefix inet;
  }
}

```

```

organization "ALTO WG";
contact "alto@ietf.org";

description
  "This module defines a semantically equivalent data model
   for the ALTO services defined in RFC7285.";

/* Potential issue for future EXTENSION: YANG 1.0 is unable to
 * augment a typedef or enum type. e.g., there is not a good way to
 * add a cost metric or type of address if this document is
 * fixed. We have marked such places with "EXTENSION." */

revision 2014-10-27 {
  description "Revision 2: Custom Data Instances";
}

revision 2014-10-24 {
  description "Initial revision: RPC only.";
}

/*****
 * TYPE DEFINITIONS *
 *****/

/*****
Definitions for addresses

ALTO RFC7285 uses the following addresses, as shown in the
examples below:

- Endpoint property service (Sec. 11.4.1.7):
  "endpoints" : [ "ipv4:192.0.2.34",
                  "ipv4:203.0.113.129" ]
- Endpoint cost service (Sec. 11.5.1.7):
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45"
    ]
  }
- Network map (Sec. 11.2.1.7.):
  "ipv4": [
    "192.0.2.0/24",
    "198.51.100.0/25"
  ],
  "ipv6": [
    "2001:db8:0:1::/64",
    "2001:db8:0:2::/64"
  ]

```

```
]
```

To handle the proceeding, we need the following definitions:

```

  ipv4-address (e.g., 192.0.2.0, already defined in rfc6991),
  ipv6-address (already defined in rfc6991),
  ipv4-prefix (e.g., 192.0.2.0/24, already defined in rfc6991),
  ipv6-prefix (defined in rfc6991),
  typed-ipv4-address (e.g., ipv4:192.0.2.1, to be defined below)
  typed-ipv6-address
  typed-ipv4-prefix-list (e.g., "ipv4": [
    "192.0.2.0/24",
    "198.51.100.0/25"
  ],

```

```

*****/

```

```
/*
```

First define typed-ipv4-address and typed-ipv6-address, as used by endpoint services.

The ideal case is to define it as "ipv4:"+ipv4-address, but there is not such a type constructor (YANG EXTENSION). Hence, the current definition cuts-and-pastes (i.e., repeats verbatim) the definition of ipv4-address and prepend "ipv4:". The downside is that if someone redefines ipv4-address, there could be inconsistency.

```
*/
```

```

typedef typed-ipv4-address {
  type string {
    pattern
      'ipv4:(((0-9)|[1-9][0-9]|1[0-9][0-9]|'
      + '2[0-4][0-9]|25[0-5])\.){3}'
      + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
      + '(%[\p{N}\p{L}]+)?';
  }
}

```

```

typedef typed-ipv6-address {
  type string {
    pattern 'ipv6:(((0-9a-fA-F){0,4}):)([0-9a-fA-F]{0,4}):{0,5}'
      + '(((0-9a-fA-F){0,4}):)?(:|[0-9a-fA-F]{0,4}))|'
      + '(((25[0-5]|2[0-4][0-9]|01)?[0-9]?[0-9])\.){3}'
      + '(25[0-5]|2[0-4][0-9]|01)?[0-9]?[0-9]))'
      + '(%[\p{N}\p{L}]+)?';
    pattern 'ipv6:([[:xdigit:]]{6}([[:xdigit:]]+|(\.\.\.)))|'
      + '([[:xdigit:]]+)*[[:xdigit:]]?::([[:xdigit:]]+)*[[:xdigit:]]?'

```

```

        + '(%.)?';
    }
}

typedef typed-endpoint-address {
    type union {
        type typed-ipv4-address;
        type typed-ipv6-address;
        // EXTENSION: ADD NEW TYPE HERE.
    }
    description
        "Ref: RFC7285 Sec. 10.4.1 Typed Endpoint Addresses" +
        "= AddressType:EndpointAddr";
}

```

/* Next, we define endpoint address group, as used in the definition of ALTO network maps. Specifically, an endpoint address group in ALTO is defined as a key-value store, with address type as key, and an array of prefix as the value of each key:

```

EndpointAddrGroup. RFC7285 Sec. 10.4.5." +
    object-map {
        AddressType -> endpoint-prefix<0..*>;
    } EndpointAddrGroup;

```

There are two challenges:

1) To specify that AddressType is key, we must use the list type, which is the only type that one can specify key. However, the current JSON-YANG encoding generates an array, instead of a key-value map;

2) Ideally, we want to enforce address type and prefix consistency; for example, an ipv6 prefix in an ipv4 type should not be allowed. However, we encounter problems. We leave this as an OPEN ISSUE.

```

*/

typedef endpoint-address-type {
    type union {
        type enumeration {
            enum ipv4;
            enum ipv6;
            // EXTENSION: ADD NEW TYPE HERE
        }
    }
    description
        "Ref: RFC7285 Sec 2.2.";
}

```

```

}

typedef endpoint-prefix {
  type inet:ip-prefix;
  description
    "endpoint prefix, identical to ip-prefix defined in RFC6991.";
}

grouping endpoint-address-group {
  list endpoint-address-group {
    key address-type;
    leaf address-type {
      type endpoint-address-type;
      mandatory true;
    }
    leaf-list endpoint-prefix {
      type endpoint-prefix;
    }
  }
  description
    "EndpointAddrGroup. RFC7285 Sec. 10.4.5." +
    " object-map {
      AddressType -> endpoint-prefix<0..*>;
    } EndpointAddrGroup;";
}

/*****
* Definitions for IDs and names
*
* ALTO defines the following concepts that are names and IDs:
*
*   pid name (used in network map, cost map),
*   resource IDs (used to identify alto network/cost maps),
*   version tag (used to indicate uniqueness of resource),
*   cost-type-name (used in IRD),
*   cost-metric,
*   cost-mode
*
* We group their definitions together below.
*****/

typedef valid-id-string {
  type string {
    length "1..64";
    pattern "[0-9a-zA-Z_\\-:@\\.]+";
  }
  description
    "Type for valid ID strings.";
}

```

```

    }

    typedef pid-name {
        type valid-id-string;
        description
            "Name for the PID." +
            "RFC7285, Section 10.1. Note: the '.' separator MUST NOT be" +
            "used unless specifically indicated in RFC7285 or an" +
            " extension document.";
    }

    typedef resource-id {
        type valid-id-string;
        description
            "Resource-ID.";
    }

    grouping vtag {
        leaf resource-id {
            type resource-id;
            mandatory true;
        }
        leaf tag {
            type string {
                length "1..64";
                pattern "[!-~]+";
            }
            mandatory true;
            description
                "Tag. RFC7285 Sec. 10.3. U+0021-U+007E";
        }
        description
            "Version tag. Both resource-id and tag must be equal
            byte-for-byte. RFC7285 Sec. 10.3." +
            " object {
                ResourceID resource-id;
                JSONString tag;
            } VersionTag;";
    }

    grouping dependent-vtags {
        list dependent-vtags {
            uses vtag;
            min-elements 1;
        }
    }

    /*****

```


Definitions for cost type and cost types

In ALTO, a cost type consists of two required components:

cost-metric,
cost-mode
and an optional description component.

In the IRD, one can name each cost type. Such info is collected in a hash map called cost types.

*****/

```
typedef cost-metric {
  type union {
    type enumeration {
      enum routingcost {
        description
          "Default metric. MUST support. RFC7285 Sec. 6.1.1.1.";
      }
      enum hopcount {
        description
          "Hopcount metric.";
      }
      // EXTENSION: Additional cost-metric will be defined here.
    }
    type string {
      length 1..32;
      pattern "priv:[0-9a-zA-Z_\-:\.]+";
    }
  }
  description
    "Cost metric. for type string,
    'priv:' reserved for Private Use.";
}

typedef cost-mode {
  type enumeration {
    enum numerical {
      description
        "Numerical cost mode.";
    }
    enum ordinal {
      description
        "Ordinal cost mode.";
    }
    // EXTENSION: Additional cost-mode will be defined here.
  }
  description
```

```

    "Cost mode. MUST support at least one of numerical and ordinal";
}

grouping cost-type {
  leaf cost-mode {
    type cost-mode;
    mandatory true;
    description
      "Cost mode.";
  }
  leaf cost-metric {
    type cost-metric;
    mandatory true;
    description
      "Cost metric.";
  }
  leaf description {
    type string;
    description
      "Optional description field.";
  }
  description
    "Cost type. RFC7285 Sec. 10.7." +
    " object {
      CostMetric cost-metric;
      CostMode cost-mode;
      [JSONString description;]
    } CostType;";
}

typedef cost-type-name {
  type valid-id-string;
  // NOTE: not fully specified in RFC7285, default as valid id
}

grouping cost-types {
  list cost-types {
    key cost-type-name;
    leaf cost-type-name {
      type cost-type-name;
    }
    uses cost-type;
  }
  description
    "RFC 7285 Sec. 9.2.2." +
    "object-map {
      JSONString -> CostType;
    } IRDMetaCostTypes;";
}

```

```

}

/*****
 * Definitions for endpoint properties *
 *****/
typedef global-endpoint-property {
    type union {
        type enumeration {
            enum pid {
                description "PID property.";
            }
            // EXTENSION: other options here
        }
        type string {
            pattern "priv:[\w\-\:@]+";
        }
    }
    description
        "Global endpoint property. RFC7285 Sec. 10.8.2." +
        "'priv:' for Private Use " +
        " length 1..32; '.' is not allowed";
}

/*
 * Ideally we would want to extend the typedef of resource-id and
 * global endpoint properties, however, YANG 1.0 does not allow
 * that, hence we simply copied the regex for resource-id over
 * verbatim.
 */

typedef resource-specific-endpoint-property {
    type string {
        length "3..97"; //len(resource-id) + 1 + len(global-property)
        pattern "(priv:)?[\w\-\:@\.\+]\. [\w\-\:@\.\+]+"; // resource-id.property
    }
    description
        "Resource-specific endpoint property.";
}

typedef endpoint-property-type {
    type union {
        type resource-specific-endpoint-property;
        type global-endpoint-property;
    }
    description
        "Endpoint property type. RFC7285 Sec. 10.8.";
}

```

```

typedef endpoint-property-value {
    type string;
    description
        "Endpoint property (value).";
}

/*****
* Definitions for response header
*****/

typedef media-type {
    type union {
        type string {
            pattern "application/alto\-.*";
        }
        type enumeration {
            enum alto-directory+json;
            enum alto-networkmap+json;
            enum alto-networkmapfilter+json;
            enum alto-costmap+json;
            enum alto-costmapfilter+json;
            enum alto-endpointprop+json;
            enum alto-endpointpropparams+json;
            enum alto-endpointcost+json;
            enum alto-endpointcostparams+json;
            enum alto-error+json;
        }
    }
}

grouping alto-cost {
    anyxml cost {
        mandatory true;
        description
            "ALTO cost is a JSONValue, which could be
            an object, array, string, etc. (Ref: RFC 7159 Sec.3.);";
    }
}

typedef constraint {
    type string {
        pattern "(gt|ge|lt|le|eq) [0-9]+";
    }
    description
        "RFC7285 Sec. 11.3.2.3. The second part must be in the " +
        "same unit as cost-metric, IEEE 754 2008 floating point.";
}

```

```

/*****
  Groupings for ALTO information resource
*****/

/* meta */
grouping IRD-meta {
  uses cost-types;
  leaf default-alto-network-map {
    type leafref {
      path "/resources/IRD/resources/resource-id";
    }
    mandatory true;
  }
}

grouping network-map-meta {
  container vtag {
    uses vtag;
  }
}

grouping cost-map-meta {
  uses dependent-vtags {
    refine dependent-vtags {
      max-elements 1;
    }
  }
  container cost-type {
    uses cost-type;
  }
}

grouping endpoint-property-meta {
  uses dependent-vtags;
}

/* accepts (optional) */
grouping accepts {
  leaf-list accepts {
    type media-type;
    min-elements 1;
  }
}

/* capabilities (capabilities) */
grouping IRD-capabilities {
  container capabilities {
    leaf cost-constraints {

```

```

        type boolean;
    }
    leaf-list cost-type-names {
        type cost-type-name;
    }
    leaf-list prop-types {
        type endpoint-property-type;
    }
}

/* uses (optional) */
grouping uses {
    leaf-list uses {
        type leafref {
            path "/resources/IRD/resources/resource-id";
        }
        min-elements 1;
    }
}

/* Information Resource Directory Grouping */
grouping IRD {
    container meta {
        uses IRD-meta;
    }
    uses IRD-data;
}

grouping IRD-data {
    list resources {
        key resource-id;
        leaf resource-id {
            type resource-id;
            mandatory true;
        }
        leaf uri {
            type inet:uri;
            mandatory true;
        }
        leaf media-type {
            type media-type;
            mandatory true;
        }
        uses accepts {
            when "current()";
        }
        uses IRD-capabilities {

```

```

        when "current()";
    }
    uses uses {
        when "current()";
    }
    description
        "IRDRResourceEntry. RFC7285 9.2.2." +
        " object {
            JSONString      uri;
            JSONString      media-type;
            [JSONString      accepts;]
            [Capabilities     capabilities;]
            [ResourceID      uses<0...*>;]
        } IRDRResourceEntry;" +
        "IRDRResourceEntries. RFC7285 9.2.2." +
        " object-map {
            ResourceID -> IRDRResourceEntry;
        } IRDRResourceEntries;" +
        "InformationResourceDirectory. RFC7285 9.2.2." +
        " object {
            IRDRResourceEntries resources;
        } InfoResourceDirectory : ResponseEntityBase;";
    }
}

/* Network Map Grouping */
grouping network-map {
    container meta {
        uses network-map-meta;
    }
    uses network-map-data;
}

grouping network-map-data {
    list network-map {
        key "pid";
        leaf pid {
            type pid-name;
        }
    }
    uses endpoint-address-group;
    description
        "RFC7285 Sec. 11.2.1.6." +
        " object-map {
            PIDName -> EndpointAddrGroup;
        } NetworkMapData;";
    }
    description
        "Network map. RFC7285 Sec. 11.2.1.6." +

```

```

    "object {
        NetworkMapData network-map;
    } InfoResourceNetworkMap : ResponseEntityBase;";
}

/* Cost Map Grouping */
grouping cost-map {
    container meta {
        uses cost-map-meta;
    }
    uses cost-map-data;
}

grouping cost-map-data {
    list cost-map {
        leaf src {
            type pid-name;
            description
                "Source PID.";
        }
        key "src";
        list dst-costs {
            leaf dst {
                type pid-name;
                description
                    "Destination PID.";
            }
            key "dst";
            uses alto-cost {
                description
                    "Cost from source to destination.";
            }
            description
                "The list represents the inner part of the cost matrix." +
                "DstCosts. RFC7285 Sec. 11.2.3.6." +
                " object-map {
                    PIDName -> JSONValue;
                } DstCosts;";
        }
        description
            "The list represents the outer part of the cost matrix." +
            "CostMapData. RFC7285 Sec. 11.2.3.6." +
            " object-map {
                PIDName -> DstCosts;
            } CostMapData;";
    }
    description
        "Cost map. RFC7285 Sec. 11.2.3.6." +

```



```

    " object {
        CostMapData cost-map;
    } InfoResourceCostMap : ResponseEntityBase;";
}

/* Endpoint Property Map Grouping */
grouping endpoint-property-map {
    container meta {
        uses endpoint-property-meta;
    }
    uses endpoint-property-map-data;
}

grouping endpoint-property-map-data {
    list endpoint-properties {
        key endpoint;
        leaf endpoint {
            type typed-endpoint-address;
            mandatory true;
        }
    }
    list properties {
        key property-type;
        leaf property-type {
            type endpoint-property-type;
            mandatory true;
        }
    }
    leaf property {
        type endpoint-property-value;
        mandatory true;
    }
    description
        "EndpointProps. RFC7285 Sec. 11.4.1.6." +
        " object {
            EndpointPropertyType -> JSONValue;
        } EndpointProps;";
    }
    description
        "EndpointPropertyMapData. Sec. 11.4.1.6." +
        " object-map {
            TypedEndpointAddr -> EndpointProps;
        } EndpointPropertyMapData;";
    }
    description
        "InfoResourceEndpointProperties. Sec. 11.4.1.6." +
        " object {
            EndpointPropertyMapData endpoint-properties;
        } InfoResourceEndpointProperties : ResponseEntityBase;";
}

```

```

/*****
  DATA INSTANCES of all ALTO information resources

  unfiltered network-maps, unfiltered cost-maps are all instances
  of resources. IRD is also modeled as data.

  The design uses augment as the basic approach to implement
  inheritance.
*****/

container resources {
  config false;

  /* Information Resource Directory */
  container IRD {
    uses IRD;
  }

  list network-maps {
    unique "meta/vtag/resource-id meta/vtag/tag";
    uses network-map;
  }

  list cost-maps {
    /*
      The dependent-vtags and cost-types in the meta uniquely
      identify a cost map. However, the unique statement in YANG
      does not allow reference to a list (dependent-vtags).
    */
    uses cost-map;
  }

  container endpoint-property-map {
    uses endpoint-property-map;
  }
}

/*****
 * RPCs
*****/

rpc IRD-service {
  output {
    container IRD-service {
      uses IRD;
    }
  }
}

```

```

    }

    rpc network-map-service {
        input {
            leaf uri {
                type leafref {
                    path "/resources/IRD/resources/uri";
                }
                mandatory true;
                description
                    "The uri of the request for the full network map.";
            }
        }
        output {
            container network-map-service {
                uses network-map;
            }
        }
    }

    rpc cost-map-service {
        input {
            leaf uri {
                type leafref {
                    path "/resources/IRD/resources/uri";
                }
                mandatory true;
                description
                    "The uri of the request for the full network map.";
            }
        }
        output {
            container cost-map-service {
                uses cost-map;
            }
        }
    }

    rpc filtered-network-map-service {
        description
            "inquiries on filtered network map" +
            "ReqFilteredNetworkMap. RFC7285 Sec. 11.3.1.3." +
            " object {
                PIDName pids<0..*>;
                [AddressType address-types<0..*>;]
            } ReqFilteredNetworkMap;";
        input {
            leaf-list pids {

```

```

        must "current()";
        type pid-name;
    }
    leaf-list address-types {
        type endpoint-address-type;
    }
}
output {
    container filtered-network-map-service {
        uses network-map;
    }
}
}

rpc filtered-cost-map-service {
    input {
        container cost-type {
            must "current()";
            uses cost-type;
        }
        leaf-list constraints {
            type constraint;
            description
                "RFC7285 Sec. 11.3.2.3.";
        }
        container pids {
            leaf-list srcs {
                type pid-name;
                description
                    "Source endpoint addresses.";
            }
            leaf-list dsts {
                type pid-name;
                description
                    "Destination endpoint addresses.";
            }
        }
        description
            "PIDFilter: Endpoint addresses. RFC7285 Sec. 11.3.2.3." +
            " object {
                PIDName srcs<0..*>;
                PIDName dsts<0..*>;
            } PIDFilter;";
    }
}
output {
    container filtered-cost-map-service {
        uses cost-map;
    }
}

```

```

    }
  }

  rpc endpoint-property-service {
    description
      "inquiries on properties of an endpoint" +
      " object {
          EndpointPropertyType  properties<1..*>;
          TypedEndpointAddr     endpoints<1..*>;
        } ReqEndpointProp;";
    input {
      leaf-list properties {
        type endpoint-property-type;
        min-elements 1;
      }
      leaf-list endpoints {
        type typed-endpoint-address;
        min-elements 1;
      }
    }
    output {
      container endpoint-property-service {
        uses endpoint-property-map;
      }
    }
  }
}

```

```

  rpc endpoint-cost-service {
    description
      "ReqEndpointCostMap. RFC7285 Sec. 11.5.1.3." +
      " object {
          CostType             cost-type;
          [JSONString          constraints<0..*>;]
          EndpointFilter        endpoints;
        } ReqEndpointCostMap;";
    input {
      container cost-type {
        must "current()";
        uses cost-type;
      }
      leaf-list constraints {
        type constraint;
        description
          "RFC7285 Sec. 11.5.1.3.";
      }
      container endpoints {
        must "current()";
      }
    }
  }
}

```

```

    leaf-list srcs {
        type typed-endpoint-address;
        description
            "Source endpoint addresses.";
    }
    leaf-list dsts {
        type typed-endpoint-address;
        description
            "Destination endpoint addresses.";
    }
    description
        " EndpointFilter: Endpoint addr. RFC7285 Sec. 11.5.1.3." +
        " object {
            [TypedEndpointAddr srcs<0..*>;]
            [TypedEndpointAddr dsts<0..*>;]
        } EndpointFilter;";
    }
}
output {
    container endpoint-cost-service {
        container meta {
            container cost-type {
                uses cost-type;
            }
        }
    }
    list endpoint-cost-map {
        leaf src {
            type typed-endpoint-address;
            description
                "Source endpoint address.";
        }
        key "src";
        list dst-costs {
            leaf dst {
                type typed-endpoint-address;
                description
                    "Destination endpoint address.";
            }
            key "dst";
            uses alto-cost {
                description
                    "Cost from source to destination.";
            }
        }
        description
            "The list represents the inner part of the cost matrix." +
            "EndpointDstCosts. RFC7285 Sec. 11.5.1.6." +
            " object-map {
                TypedEndpointAddr -> JSONValue;
            }
    }
}

```

```

    } EndpointDstCosts;";
}
description
"The list represents the outer part of the cost matrix." +
"EndpointCostMapData. RFC7285 Sec. 11.5.1.6." +
" object {
    EndpointCostMapData endpoint-cost-map;
} InfoResourceEndpointCostMap : ResponseEntityBase;
object-map {
    TypedEndpointAddr -> EndpointDstCosts;
} EndpointCostMapData;";
}
}
}
}
}

```

Appendix B. YANG-Validated JSON Messages for ALTO Examples

We established that the YANG validated messages cannot be syntactically equivalent to the ALTO protocol messages. For a selection of the examples in the ALTO protocol, we provide the YANG validated version of the message for comparison purposes.

B.1. IRD Response Example

The ALTO example of IRD response as specified in Section 9.2.3 of [RFC7285].

```
{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": "1",
    "IRD-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "meta": {
        "cost-types": [
          {
            "cost-type-name": "num-routing",
            "cost-mode": "numerical",
            "cost-metric": "routingcost",
            "description": "My default"
          },
          {
            "cost-type-name": "num-hop",
            "cost-mode": "numerical",

```

```

        "cost-metric": "hopcount"
      },
      {
        "cost-type-name": "ord-routing",
        "cost-mode": "ordinal",
        "cost-metric": "routingcost"
      },
      {
        "cost-type-name": "ord-hop",
        "cost-mode": "ordinal",
        "cost-metric": "hopcount"
      }
    ],
    "default-alto-network-map": "my-default-network-map"
  },
  "resources": [
    {
      "resource-id": "my-default-network-map",
      "uri": "http://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },
    {
      "resource-id": "numerical-routing-cost-map",
      "uri": "http://alto.example.com/costmap/num/routingcost",
      "media-type": "application/alto-costmap+json",
      "capabilities": { "cost-type-names": [ "num-routing" ] },
      "uses": [ "my-default-network-map" ]
    },
    {
      "resource-id": "numerical-hopcount-cost-map",
      "uri": "http://alto.example.com/costmap/num/hopcount",
      "media-type": "application/alto-costmap+json",
      "capabilities": { "cost-type-names": [ "num-hop" ] },
      "uses": [ "my-default-network-map" ]
    },
    {
      "resource-id": "custom-maps-resources",
      "uri": "http://custom.alto.example.com/maps",
      "media-type": "application/alto-directory+json"
    },
    {
      "resource-id": "endpoint-property",
      "uri": "http://alto.example.com/endpointprop/lookup",
      "media-type": "application/alto-endpointprop+json",
      "accepts": "application/alto-endpointpropparams+json",
      "capabilities": {
        "prop-types": [
          "my-default-network-map.pid",

```



```
    "priv:iETF-example-prop"
  ]
}
},
{
  "resource-id": "endpoint-cost",
  "uri": "http://alto.example.com/endpointcost/lookup",
  "media-type": "application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-constraints": "true",
    "cost-type-names": [
      "num-routing",
      "num-hop",
      "ord-routing",
      "ord-hop"
    ]
  }
}
]
}
}
}
```

The corresponding XML file which is validated by YANG.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="1">
    <IRD-service xmlns="urn:ietf:params:xml:ns:yang:alto-service">
      <meta>
        <cost-types>
          <cost-type-name>num-routing</cost-type-name>
          <cost-mode>numerical</cost-mode>
          <cost-metric>routingcost</cost-metric>
          <description>My default</description>
        </cost-types>
        <cost-types>
          <cost-type-name>num-hop</cost-type-name>
          <cost-mode>numerical</cost-mode>
          <cost-metric>hopcount</cost-metric>
        </cost-types>
        <cost-types>
          <cost-type-name>ord-routing</cost-type-name>
          <cost-mode>ordinal</cost-mode>
          <cost-metric>routingcost</cost-metric>
        </cost-types>
        <cost-types>
```

```

        <cost-type-name>ord-hop</cost-type-name>
        <cost-mode>ordinal</cost-mode>
        <cost-metric>hopcount</cost-metric>
    </cost-types>
    <default-alto-network-map>
        my-default-network-map
    </default-alto-network-map>
</meta>
<resources>
    <resource-id>my-default-network-map</resource-id>
    <uri>http://alto.example.com/networkmap</uri>
    <media-type>application/alto-networkmap+json</media-type>
</resources>
<resources>
    <resource-id>numerical-routing-cost-map</resource-id>
    <uri>http://alto.example.com/costmap/num/routingcost</uri>
    <media-type>application/alto-costmap+json</media-type>
    <capabilities>
        <cost-type-names>num-routing</cost-type-names>
    </capabilities>
    <uses>my-default-network-map</uses>
</resources>
<resources>
    <resource-id>numerical-hopcount-cost-map</resource-id>
    <uri>http://alto.example.com/costmap/num/hopcount</uri>
    <media-type>application/alto-costmap+json</media-type>
    <capabilities>
        <cost-type-names>num-hop</cost-type-names>
    </capabilities>
    <uses>my-default-network-map</uses>
</resources>
<resources>
    <resource-id>custom-maps-resources</resource-id>
    <uri>http://custom.alto.example.com/maps</uri>
    <media-type>application/alto-directory+json</media-type>
</resources>
<resources>
    <resource-id>endpoint-property</resource-id>
    <uri>http://alto.example.com/endpointprop/lookup</uri>
    <media-type>application/alto-endpointprop+json</media-type>
    <accepts>application/alto-endpointpropparams+json</accepts>
    <capabilities>
        <prop-types>my-default-network-map.pid</prop-types>
        <prop-types>priv:ietf-example-prop</prop-types>
    </capabilities>
</resources>
<resources>
    <resource-id>endpoint-cost</resource-id>

```

```
<uri>http://alto.example.com/endpointcost/lookup</uri>
<media-type>application/alto-endpointcost+json</media-type>
<accepts>application/alto-endpointcostparams+json</accepts>
<capabilities>
  <cost-constraints>true</cost-constraints>
  <cost-type-names>num-routing</cost-type-names>
  <cost-type-names>num-hop</cost-type-names>
  <cost-type-names>ord-routing</cost-type-names>
  <cost-type-names>ord-hop</cost-type-names>
</capabilities>
</resources>
</IRD-service>
</rpc-reply>
```

B.2. Network Map Service Response Example

The ALTO example of a network map response as specified in Section 11.2.1.7 of [RFC7285].

```

{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": 2,
    "network-map-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "media-type": "application/alto-networkmap+json",
      "meta": {
        "vtag": {
          "resource-id": "my-default-network-map",
          "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
        }
      },
      "network-map": [
        {
          "pid": "PID1",
          "endpoint-address-group": {
            "address-type": "ipv4",
            "endpoint-prefix": [
              "192.0.2.0/24",
              "198.51.100.0/25"
            ]
          }
        },
        {
          "pid": "PID2",
          "endpoint-address-group": {
            "address-type": "ipv4",
            "endpoint-prefix": ["198.51.100.128/25"]
          }
        },
        {
          "pid": "PID3",
          "endpoint-address-group": [
            {
              "address-type": "ipv4",
              "endpoint-prefix": ["0.0.0.0/0"]
            },
            {
              "address-type": "ipv6",
              "endpoint-prefix": [ "::/0" ]
            }
          ]
        }
      ]
    }
  }
}

```

The corresponding YANG-validated XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <network-map-service
    xmlns="urn:ietf:params:xml:ns:yang:alto-service">
    <meta>
      <vtag>
        <resource-id>my-default-network-map</resource-id>
        <tag>da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785</tag>
      </vtag>
    </meta>
    <network-map>
      <pid>PID1</pid>
      <endpoint-address-group>
        <address-type>ipv4</address-type>
        <endpoint-prefix>192.0.2.0/24</endpoint-prefix>
        <endpoint-prefix>198.51.100.0/25</endpoint-prefix>
      </endpoint-address-group>
    </network-map>
    <network-map>
      <pid>PID2</pid>
      <endpoint-address-group>
        <address-type>ipv4</address-type>
        <endpoint-prefix>198.51.100.128/25</endpoint-prefix>
      </endpoint-address-group>
    </network-map>
    <network-map>
      <pid>PID3</pid>
      <endpoint-address-group>
        <address-type>ipv4</address-type>
        <endpoint-prefix>0.0.0.0/0</endpoint-prefix>
      </endpoint-address-group>
      <endpoint-address-group>
        <address-type>ipv6</address-type>
        <endpoint-prefix>::/0</endpoint-prefix>
      </endpoint-address-group>
    </network-map>
  </network-map-service>
</rpc-reply>
```

B.3. Filtered Cost Map Response Example

The ALTO example of a filtered cost map response as specified in Section 11.3.2.7 of [RFC7285].

```

{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": 2,
    "filtered-cost-map-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "meta": {
        "dependent-vtags": {
          "resource-id": "my-default-network-map",
          "tag": "75ed013b3cb58f896e839582504f622838ce670f"
        },
        "cost-type": {
          "cost-mode": "numerical",
          "cost-metric": "routingcost"
        }
      },
      "cost-map": [
        {
          "src": "PID1",
          "dst-costs": [
            {
              "dst": "PID1",
              "cost": "0"
            },
            {
              "dst": "PID2",
              "cost": "1"
            },
            {
              "dst": "PID3",
              "cost": "2"
            }
          ]
        }
      ]
    }
  }
}

```

The corresponding YANG-validated XML file:

```

<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <filtered-cost-map-service
    xmlns="urn:ietf:params:xml:ns:yang:alto-service">
    <meta>
      <dependent-vtags>
        <resource-id>my-default-network-map</resource-id>
        <tag>75ed013b3cb58f896e839582504f622838ce670f</tag>
      </dependent-vtags>
      <cost-type>
        <cost-mode>numerical</cost-mode>
        <cost-metric>routingcost</cost-metric>
      </cost-type>
    </meta>
    <cost-map>
      <src>PID1</src>
      <dst-costs>
        <dst>PID1</dst>
        <cost>0</cost>
      </dst-costs>
      <dst-costs>
        <dst>PID2</dst>
        <cost>1</cost>
      </dst-costs>
      <dst-costs>
        <dst>PID3</dst>
        <cost>2</cost>
      </dst-costs>
    </cost-map>
  </filtered-cost-map-service>
</rpc-reply>

```

B.4. Endpoint Property Service Response Example

The ALTO example of an endpoint property service response as specified in Section 11.4.1.7 of [RFC7285].

```

{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": "2",
    "endpoint-property-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "meta" : {
        "dependent-vtags" : [
          { "resource-id": "my-default-network-map",
            "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"
          }
        ]
      },
      "endpoint-properties": [
        {
          "endpoint": "ipv4:192.0.2.34",
          "properties" : [
            {
              "property-type": "my-default-network-map.pid",
              "property": "PID1"
            },
            {
              "property-type": "priv:ietf-example-prop",
              "property": "1"
            }
          ]
        },
        {
          "endpoint": "ipv4:203.0.113.129",
          "properties": [
            {
              "property-type": "my-default-network-map.pid",
              "property": "PID3"
            }
          ]
        }
      ]
    }
  }
}

```

The corresponding YANG-validated XML file:


```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <endpoint-property-service
    xmlns="urn:ietf:params:xml:ns:yang:alto-service">
    <meta>
      <dependent-vtags>
        <resource-id>my-default-network-map</resource-id>
        <tag>7915dc0290c2705481c491a2b4ffbec482b3cf62</tag>
      </dependent-vtags>
    </meta>
    <endpoint-properties>
      <endpoint>ipv4:192.0.2.34</endpoint>
      <properties>
        <property-type>my-default-network-map.pid</property-type>
        <property>PID1</property>
      </properties>
      <properties>
        <property-type>priv:ietf-example-prop</property-type>
        <property>1</property>
      </properties>
    </endpoint-properties>
    <endpoint-properties>
      <endpoint>ipv4:203.0.113.129</endpoint>
      <properties>
        <property-type>my-default-network-map.pid</property-type>
        <property>PID3</property>
      </properties>
    </endpoint-properties>
  </endpoint-property-service>
</rpc-reply>
```

Authors' Addresses

Xiao Shi
Yale University
51 Prospect Street
New Haven, CT 06511
USA

Email: xiao.shi@yale.edu

Y. Richard Yang
Yale University
51 Prospect St
New Haven CT
USA

Email: yang.r.yang@gmail.com

Michael Scharf
Alcatel-Lucent Bell Labs
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: michael.scharf@alcatel-lucent.com