Stream: Internet Engineering Task Force (IETF)
RFC: 9418
Category: Standards Track
Published: July 2023
ISSN: 2070-1721
Authors: B. Claise, *Huawei*; J. Quilbeuf, *Huawei*; P. Lucente, *NTT*; P. Fasano, *TIM S.p.A*; T. Arumugam, *Consultant*

RFC 9418
A YANG Data Model for Service Assurance

Abstract

This document specifies YANG modules for representing assurance graphs. These graphs represent the assurance of a given service by decomposing it into atomic assurance elements called subservices. The companion document, "Service Assurance for Intent-Based Networking Architecture" (RFC 9417), presents an architecture for implementing the assurance of such services.

The YANG data models in this document conform to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9418.

Copyright Notice

Table of Contents

1. Introduction
   1.1. Terminology
2. YANG Modules Overview
3. Base IETF Service Assurance YANG Module
   3.1. Concepts
   3.2. Tree View
   3.3. YANG Module
   3.4. Rejecting Circular Dependencies
4. Guidelines for Defining New Subservice Types
5. Subservice Augmentation: "ietf-service-assurance-device" YANG Module
   5.1. Tree View
   5.2. Concepts
   5.3. YANG Module
6. Subservice Augmentation: "ietf-service-assurance-interface" YANG Module
   6.1. Tree View
   6.2. Concepts
   6.3. YANG Module
7. Security Considerations
8. IANA Considerations
   8.1. The IETF XML Registry
   8.2. The YANG Module Names Registry
9. References
   9.1. Normative References
   9.2. Informative References

Appendix A. Vendor-Specific Subservice Augmentation: "example-service-assurance-device-acme" YANG Module

A.1. Tree View

A.2. Concepts

A.3. YANG Module

Appendix B. Further Augmentations: IP Connectivity and IS-IS Subservices

B.1. IP Connectivity Module Tree View

B.2. IS-IS Module Tree View

B.3. Global Tree View

B.4. IP Connectivity YANG Module

B.5. IS-IS YANG Module

Appendix C. Example of a YANG Instance

Appendix D. YANG Library for Service Assurance

Acknowledgements

Authors' Addresses

1. Introduction

[RFC9417] describes an architecture and a set of involved components for service assurance, called Service Assurance for Intent-based Networking (SAIN). This document complements the architecture by specifying a data model for the interfaces between components. More specifically, the document provides YANG modules for the purpose of service assurance in a format that is:

- machine readable,
- vendor independent, and
- augmentable such that SAIN agents from Figure 1 of [RFC9417] can support and expose new subservices to SAIN orchestrators and collectors.

1.1. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms used in this document are defined in [RFC9417].

The meanings of the symbols in the tree diagrams are defined in [RFC8340].

2. YANG Modules Overview

The main YANG module, "ietf-service-assurance" (Section 3), defines objects for assuring network services based on their decomposition into so-called subservices. The subservices are hierarchically organized by dependencies. The subservices, along with the dependencies, constitute an assurance graph. This module should be supported by an agent that is able to interact with the devices in order to produce the health statuses and symptoms for each subservice in an assurance graph. This module is intended for the following use cases:

- Assurance graph configuration:
  - Subservices: Configure a set of subservices to assure by specifying their types and parameters.
  - Dependencies: Configure the dependencies between the subservices, along with their types.
- Assurance telemetry: Export the assurance graph with health statuses and symptoms for each node.

The module is also intended to be exported by the SAIN collector that aggregates the output of several SAIN agents to provide the global assurance graph. In that case, only the telemetry export use case is considered.

The modules presented in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

The second YANG module, "ietf-service-assurance-device" (Section 5), augments the "ietf-service-assurance" module by adding support for the device subservice. Additional subservice types might be added following a similar approach.

The third YANG module, "ietf-service-assurance-interface" (Section 6), augments the "ietf-service-assurance" module as well by adding support for the interface subservice.

We provide additional examples in the appendix. The module "example-service-assurance-device-acme" (Appendix A) augments the "ietf-service-assurance-device" module to customize it for devices of the fictional Acme Corporation. Additional vendor-specific parameters might be added following a similar approach. We also provide the modules "example-service-assurance-ip-connectivity" and "example-service-assurance-is-is" (Appendix B) to model the example in Figure 2 from Section 3.1 of [RFC9417].

3. Base IETF Service Assurance YANG Module

3.1. Concepts

The "ietf-service-assurance" YANG module assumes a set of subservices to be assured independently. A subservice is a feature or a subpart of the network system that a given service instance depends on. Examples of subservice types include the following:

- device: Whether a device is healthy, and if not, what are the symptoms? Such a subservice might monitor the device resources, such as CPU, RAM, or Ternary Content-Addressable Memory (TCAM). Potential symptoms are "CPU overloaded", "Out of RAM", or "Out of TCAM".
- ip-connectivity: Given two IP addresses bound to two devices, what is the quality of the IP connectivity between them? Potential symptoms are "No route available" or "Equal-Cost Multipaths (ECMPs) imbalance".

An instance of the device subservice is representing a subpart of the network system, namely a specific device. An instance of the ip-connectivity subservice is representing a feature of the network, namely the connectivity between two specific IP addresses on two devices. In both cases, these subservices might depend on other subservices, for instance, the connectivity might depend on a subservice representing the routing system and on a subservice representing ECMPs.

The two example subservices presented above need different sets of parameters to fully characterize one of their instances. An instance of the device subservice is fully characterized by a single parameter allowing to identify the device to monitor. For the ip-connectivity subservice, at least the device and IP address for both ends of the link are needed to fully characterize an instance.

The base model presented in this section specifies a single type of subservice, which represents service instances. Such nodes play a particular role in the assurance graph because they represent the starting point, or root, for the assurance graph of the corresponding service instance. The parameters required to fully identify a service instance are the name of the service and the name of the service instance. To support other types of subservices, such as device or ip-connectivity, the "ietf-service-assurance" module is intended to be augmented.

The dependencies are modeled as a list, i.e., each subservice contains a list of references to its dependencies. That list can be empty if the subservice instance does not have any dependencies.

By specifying service instances and their dependencies in terms of subservices, one defines a global assurance graph. That assurance graph is the result of merging all the individual assurance graphs for the assured service instances. Each subservice instance is expected to appear only once in the global assurance graph even if several service instances depend on it. For example, an instance of the device subservice is a dependency of every service instance that

relies on the corresponding device. The assurance graph of a specific service instance is the subgraph obtained by traversing the global assurance graph through the dependencies, starting from the specific service instance.

An assurance agent configured with such a graph is expected to produce, for each configured subservice, a health status that indicates how healthy the subservice is. If the subservice is not healthy, the agent is expected to produce a list of symptoms explaining why the subservice is not healthy.

3.2. Tree View

The following tree diagram [RFC8340] provides an overview of the "ietf-service-assurance" module.

```
module: ietf-service-assurance
  +--ro assurance-graph-last-change    yang:date-and-time
  +--rw subservices
  |  +--rw subservice* [type id]
  |     +--rw type                                identityref
  |     +--rw id                                  string
  |     +--ro last-change?                        yang:date-and-time
  |     +--ro label?                              string
  |     +--rw under-maintenance!
  |     |  +--rw contact    string
  |     +--rw (parameter)
  |     |  +--:(service-instance-parameter)
  |     |     +--rw service-instance-parameter
  |     |        +--rw service          string
  |     |        +--rw instance-name    string
  |     +--ro health-score                        int8
  |     +--ro symptoms-history-start?             yang:date-and-time
  |     +--ro symptoms
  |     |  +--ro symptom* [start-date-time agent-id symptom-id]
  |     |     +--ro symptom-id             leafref
  |     |     +--ro agent-id               -> /agents/agent/id
  |     |     +--ro health-score-weight?   uint8
  |     |     +--ro start-date-time        yang:date-and-time
  |     |     +--ro stop-date-time?        yang:date-and-time
  |     +--rw dependencies
  |        +--rw dependency* [type id]
  |           +--rw type
  |           |       -> /subservices/subservice/type
  |           +--rw id                 leafref
  |           +--rw dependency-type?   identityref
  +--ro agents
  |  +--ro agent* [id]
  |     +--ro id          string
  |     +--ro symptoms* [id]
  |        +--ro id             string
  |        +--ro description    string
  +--ro assured-services
     +--ro assured-service* [service]
        +--ro service      leafref
        +--ro instances* [name]
           +--ro name           leafref
           +--ro subservices* [type id]
              +--ro type    -> /subservices/subservice/type
              +--ro id      leafref
```

The date of the last change in "assurance-graph-last-change" is read only. It must be updated each time the graph structure is changed by addition or deletion of subservices and dependencies or modifications of their configurable attributes, including their maintenance statuses. Such modifications correspond to a structural change in the graph. The date of the last change is useful for a client to quickly check if there is a need to update the graph structure. A change in the health score or symptoms associated to a service or subservice does not change the structure of the graph, and thus has no effect on the date of the last change.

The "subservices" list contains all the subservice instances currently known by the server (i.e., SAIN agent or SAIN collector). A subservice declaration **MUST** provide the following:

- a subservice type ("type"): a reference to an identity that inherits from "subservice-base", which is the base identity for any subservice type
- an id ("id"): a string uniquely identifying the subservice among those with the same type

The type and id uniquely identify a given subservice.

The "last-change" indicates when the dependencies or maintenance status of this particular subservice were last modified.

The "label" is a human-readable description of the subservice.

The presence of the "under-maintenance" container inhibits the emission of symptoms for the subservice and subservices that depend on them. In that case, a "contact" **MUST** be provided to indicate who or which software is responsible for the maintenance. See Section 3.6 of [RFC9417] for a more detailed discussion.

The "parameter" choice is intended to be augmented in order to describe parameters that are specific to the current subservice type. This base module defines only the subservice type representing service instances. Service instances **MUST** be modeled as a particular type of subservice with two parameters: "service" and "instance-name". The "service" parameter is the name of the service defined in the network orchestrator, for instance, "point-to-point-l2vpn". The "instance-name" parameter is the name assigned to the particular instance to be assured, for instance, the name of the customer using that instance.

The "health-score" contains a value normally between 0 and 100, indicating how healthy the subservice is. As mentioned in the health score definition, the special value -1 can be used to specify that no value could be computed for that health score, for instance, if some metric needed for that computation could not be collected.

The "symptoms-history-start" is the cutoff date for reporting symptoms. Symptoms that were terminated before that date are not reported anymore in the model.

The status of each subservice contains a list of symptoms. Each symptom is specified by:

- an identifier "symptom-id", which identifies the symptom locally to an agent,
- an agent identifier "agent-id", which identifies the agent raising the symptom,
- a "health-score-weight" specifying the impact to the health score incurred by this symptom,
- a "start-date-time" indicating when the symptom became active, and
- a "stop-date-time" indicating when the symptom stopped being active (this field is not present if the symptom is still active).

In order for the pair "agent-id" and "symptom-id" to uniquely identify a symptom, the following is necessary:

- "agent-id" **MUST** be unique among all agents of the system.

- "symptom-id" **MUST** be unique among all symptoms raised by the agent.

Note that "agent-id" and "symptom-id" are leafrefs pointing to the objects defined later in the document. While the combination of "symptom-id" and "agent-id" is sufficient as a unique key list, the "start-date-time" second key helps to sort and retrieve relevant symptoms.

The "dependency" list contains the dependencies for the current subservice. Each of them is specified by a leafref to both "type" and "id" of the target dependencies. A dependency has a type indicated in the "dependency-type" field. Two types are specified in the model:

- Impacting: Such a dependency indicates an impact on the health of the dependent.
- Informational: Such a dependency might explain why the dependent has issues but does not impact its health.

To illustrate the difference between "impacting" and "informational", consider the interface subservice representing a network interface. If the device to which the network interface belongs goes down, the network interface will transition to a "down" state as well. Therefore, the dependency of the interface subservice towards the device subservice is "impacting". On the other hand, a dependency towards the ecmp-load subservice, which checks that the load between ECMPs remains stable throughout time, is only "informational". Indeed, services might be perfectly healthy even if the load distribution between ECMPs changed. However, such an instability might be a relevant symptom for diagnosing the root cause of a problem.

Within the container "agents", the list "agent" contains the list of symptoms per agent. The key of the list is the "id", which **MUST** be unique among agents of a given assurance system. For each agent, the list "symptoms-description" maps an "id" to its "description". The "id" **MUST** be unique among the symptoms raised by the agent.

Within the container "assured-services", the list "assured-service" contains the subservices indexed by assured service instances. For each service type identified by the "service" leaf, all instances of that service are listed in the "instances" list. For each instance identified by the "name" leaf, the "subservices" list contains all descendant subservices that are part of the assurance graph for that specific instance. These imbricated lists provide a query optimization to get the list of subservices in that assurance graph in a single query instead of recursively querying the dependencies of each subservice, starting from the node representing the service instance.

The relation between the health score ("health-score") and the "health-score-weight" of the currently active symptoms is not explicitly defined in this document. The only requirement is that a health score that is strictly smaller than 100 (the maximal value) must be explained by at least one symptom. A way to enforce that requirement is to first detect symptoms and then compute the health score based on the "health-score-weight" of the detected symptoms. As an example, such a computation could be to sum the "health-score-weight" of the active symptoms, subtract that value from 100, and change the value to 0 if the result is negative. The relation between health score and "health-score-weight" is left to the implementor (of an agent [RFC9417]).

Keeping the history of the graph structure is out of scope for this YANG module. Only the current version of the assurance graph can be fetched. In order to keep the history of the graph structure, some time-series database (TSDB) or similar storage must be used.

3.3. YANG Module

This model contains references to [RFC6991].

```
<CODE BEGINS> file "ietf-service-assurance@2023-07-11.yang"

module ietf-service-assurance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance";
  prefix sain;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf   <mailto:jean.quilbeu@huawei.com>";
  description
    "This module defines objects for assuring services based on their
     decomposition into so-called subservices, according to the
     Service Assurance for Intent-based Networking (SAIN)
     architecture.

     The subservices hierarchically organized by dependencies
     constitute an assurance graph.  This module should be supported
     by an assurance agent that is able to interact with the devices
     in order to produce the health status and symptoms for each
     subservice in the assurance graph.

     This module is intended for the following use cases:
     * Assurance graph configuration:
       - Subservices: Configure a set of subservices to assure by
         specifying their types and parameters.
       - Dependencies: Configure the dependencies between the
         subservices, along with their type.
     * Assurance telemetry: Export the health statuses of the
       subservices, along with the observed symptoms.

     Copyright (c) 2023 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Revised BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
```

```
      Relating to IETF Documents
      (https://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC 9418; see the
      RFC itself for full legal notices.  ";

  revision 2023-07-11 {
    description
      "Initial version.";
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  identity subservice-base {
    description
      "Base identity for subservice types.";
  }

  identity service-instance-type {
    base subservice-base;
    description
      "Specific type of subservice that represents a service
       instance.  Instance of this type will depend on other
       subservices to build the top of the assurance graph.";
  }

  identity dependency-type {
    description
      "Base identity for representing dependency types.";
  }

  identity informational {
    base dependency-type;
    description
      "Indicates that symptoms of the dependency might be of interest
       for the dependent, but the status of the dependency should not
       have any impact on the dependent.";
  }

  identity impacting {
    base dependency-type;
    description
      "Indicates that the status of the dependency directly impacts
       the status of the dependent.";
  }

  grouping subservice-reference {
    description
      "Reference to a specific subservice identified by its type and
       identifier.  This grouping is only for internal use in this
       module.";
    leaf type {
      type leafref {
        path "/subservices/subservice/type";
      }
      description
        "The type of the subservice to refer to (e.g., device).";
    }
```

```
    leaf id {
      type leafref {
        path "/subservices/subservice[type=current()/../type]/id";
      }
      description
        "The identifier of the subservice to refer to.";
    }
  }

  grouping subservice-dependency {
    description
      "Represents a dependency to another subservice.  This grouping
       is only for internal use in this module";
    uses subservice-reference;
    leaf dependency-type {
      type identityref {
        base dependency-type;
      }
      description
        "Represents the type of dependency (e.g., informational or
         impacting).";
    }
  }

  leaf assurance-graph-last-change {
    type yang:date-and-time;
    config false;
    mandatory true;
    description
      "Time and date at which the assurance graph last changed after
       any structural changes (dependencies and/or maintenance
       windows parameters) are applied to the subservice(s).  The
       time and date must be the same or more recent than the most
       recent value of any changed subservices last-change time and
       date.";
  }
  container subservices {
    description
      "Root container for the subservices.";
    list subservice {
      key "type id";
      description
        "List of configured subservices.";
      leaf type {
        type identityref {
          base subservice-base;
        }
        description
          "Type of the subservice identifying the type of the part
           or functionality that is being assured by this list
           entry, for instance, interface, device, or
           ip-connectivity.";
      }
      leaf id {
        type string;
        description
          "Identifier of the subservice instance.  Must be unique
           among subservices of the same type.";
```

```
      }
      leaf last-change {
        type yang:date-and-time;
        config false;
        description
          "Date and time at which the structure for this
           subservice instance last changed, i.e., dependencies
           and/or maintenance windows parameters.";
      }
      leaf label {
        type string;
        config false;
        description
          "Label of the subservice, i.e., text describing what the
           subservice is to be displayed on a human interface.

           It is not intended for random end users but for
           network/system/software engineers that are able to
           interpret it.  Therefore, no mechanism for language
           tagging is needed.";
      }
      container under-maintenance {
        presence "true";
        description
          "The presence of this container indicates that the current
           subservice is under maintenance.";
        leaf contact {
          type string;
          mandatory true;
          description
            "A string used to model an administratively assigned name
             of the resource that is performing maintenance.

             It is suggested that this freeform field, which could be
             a URI, contains one or more of the following: IP
             address, management station name, network manager's
             name, location, and/or phone number.  It might even
             contain the expected maintenance time.

             In some cases, the agent itself will be the owner of an
             entry.  In these cases, this string shall be set to a
             string starting with 'monitor'.";
        }
      }
      choice parameter {
        mandatory true;
        description
          "Specify the required parameters per subservice type.  Each
           module augmenting this module with a new subservice type
           that is a new identity based on subservice-base should
           augment this choice as well by adding a container
           available only if the current subservice type is
           the newly added identity.";
        container service-instance-parameter {
          when "derived-from-or-self(../type,
                'sain:service-instance-type')";
          description
            "Specify the parameters of a service instance.";
```

```
          leaf service {
            type string;
            mandatory true;
            description
              "Name of the service.";
          }
          leaf instance-name {
            type string;
            mandatory true;
            description
              "Name of the instance for that service.";
          }
        }
        // Other modules can augment their own cases into here.
      }
      leaf health-score {
        type int8 {
          range "-1 .. 100";
        }
        config false;
        mandatory true;
        description
          "Score value of the subservice health.  A value of 100
           means that the subservice is healthy.  A value of 0 means
           that the subservice is broken.  A value between 0 and 100
           means that the subservice is degraded. The special value
           -1 means that the health score could not be computed.";
      }
      leaf symptoms-history-start {
        type yang:date-and-time;
        config false;
        description
          "Date and time at which the symptom's history starts for
           this subservice instance, either because the subservice
           instance started at that date and time or because the
           symptoms before that were removed due to a garbage
           collection process.";
      }
      container symptoms {
        config false;
        description
          "Symptoms for the subservice.";
        list symptom {
          key "start-date-time agent-id symptom-id";
          unique "agent-id symptom-id";
          description
            "List of symptoms of the subservice.  While the
             start-date-time key is not necessary per se, this would
             get the entries sorted by start-date-time for easy
             consumption.";
          leaf symptom-id {
            type leafref {
              path "/agents/agent[id=current()/../agent-id]"
                 + "/symptoms/id";
            }
            description
              "Identifier of the symptom to be interpreted according
               to the agent identified by the agent-id.";
```

```
          }
          leaf agent-id {
            type leafref {
              path "/agents/agent/id";
            }
            description
              "Identifier of the agent raising the current symptom.";
          }
          leaf health-score-weight {
            type uint8 {
              range "0 .. 100";
            }
            description
              "The weight to the health score incurred by this
               symptom.  The higher the value, the more of an impact
               this symptom has.  If a subservice health score is not
               100, there must be at least one symptom with a
               health-score-weight larger than 0.";
          }
          leaf start-date-time {
            type yang:date-and-time;
            description
              "Date and time at which the symptom was detected.";
          }
          leaf stop-date-time {
            type yang:date-and-time;
            description
              "Date and time at which the symptom stopped being
               detected.  Must be after the start-date-time.  If the
               symptom is ongoing, this field should not be
               populated.";
          }
        }
      }
      container dependencies {
        description
          "Indicates the set of dependencies of the current
           subservice, along with their types.";
        list dependency {
          key "type id";
          description
            "List of dependencies of the subservice.";
          uses subservice-dependency;
        }
      }
    }
  }
  container agents {
    config false;
    description
      "Container for the list of agents' symptoms.";
    list agent {
      key "id";
      description
        "Contains symptoms of each agent involved in computing the
         health status of the current graph.  This list acts as a
         glossary for understanding the symptom ids returned by each
         agent.";
```

```
      leaf id {
        type string;
        description
          "Id of the agent for which we are defining the symptoms.
           This identifier must be unique among all agents.";
      }
      list symptoms {
        key "id";
        description
          "List of symptoms raised by the current agent that is
           identified by the symptom-id.";
        leaf id {
          type string;
          description
            "Id of the symptom for the current agent.  The agent must
             guarantee the unicity of this identifier.";
        }
        leaf description {
          type string;
          mandatory true;
          description
            "Description of the symptom, i.e., text describing what
             the symptom is, is to be computer consumable and
             displayed on a human interface.

             It is not intended for random end users but for
             network/system/software engineers that are able to
             interpret it.  Therefore, no mechanism for language
             tagging is needed.";
        }
      }
    }
  }
  container assured-services {
    config false;
    description
      "Container for the index of assured services.";
    list assured-service {
      key "service";
      description
        "Service instances that are currently part of the assurance
         graph.  The list must contain an entry for every service
         that is currently present in the assurance graph.  This list
         presents an alternate access to the graph stored in
         subservices that optimizes querying the assurance graph of
         a specific service instance.";
      leaf service {
        type leafref {
          path "/subservices/subservice/service-instance-parameter/"
             + "service";
        }
        description
          "Name of the service.";
      }
      list instances {
        key "name";
        description
          "Instances of the service. The list must contain
```

```
           an entry for every instance of the parent service.";
        leaf name {
          type leafref {
            path "/subservices/subservice/service-instance-parameter"
               + "/instance-name";
          }
          description
            "Name of the service instance.  The leafref must point to
             a service-instance-parameter whose service leaf matches
             the parent service.";
        }
        list subservices {
          key "type id";
          description
            "Subservices that appear in the assurance graph of the
             current service instance.

             The list must contain the subservice corresponding to
             the service instance, i.e., the subservice that
             matches the service and instance-name keys.

             For every subservice in the list, all subservices listed
             as dependencies must also appear in the list.";
          uses subservice-reference;
        }
      }
    }
  }
}

<CODE ENDS>
```

3.4. Rejecting Circular Dependencies

The statuses of services and subservices depend on the statuses of their dependencies, and thus circular dependencies between them prevent the computation of statuses. Section 3.1.1 of the SAIN architecture document [RFC9417] discusses how such dependencies appear and how they could be removed. The responsibility of avoiding such dependencies falls to the SAIN orchestrator. However, we specify in this section the expected behavior when a server supporting the "ietf-service-assurance" module receives a data instance containing circular dependencies.

Enforcing the absence of circular dependencies as a YANG constraint falls back to implementing a graph traversal algorithm with XPath and checking that the current node is not reachable from its dependencies. Even with such a constraint, there is no guarantee that merging two graphs without dependency loops will result in a graph without dependency loops. Indeed, Section 3.1.1 of [RFC9417] presents an example where merging two graphs without dependency loops results in a graph with a dependency loop.

Therefore, a server implementing the "ietf-service-assurance" module **MUST** check that there is no dependency loop whenever the graph is modified. A modification creating a dependency loop **MUST** be rejected.

4. Guidelines for Defining New Subservice Types

The base YANG module defined in Section 3.3 only defines a single type of subservice that represent service instances. As explained above, this model is meant to be augmented so that a variety of subservices can be used in the assurance graph. In this section, we propose some guidelines for specifying such extensions at IETF.

The mechanism to add a new subservice type is to define a new module for that subservice. The module name should start with "ietf-service-assurance-". The namespace of the module should start with "urn:ietf:params:xml:ns:yang:ietf-service-assurance-". The prefix of the module should start with "sain-". For instance, the subservice type representing the assurance of a device should have:

- the name "ietf-service-assurance-device",
- the namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device", and
- the prefix "sain-device".

The new module should define:

- a new identity to represent the new type and
- the parameters fully specifying an instance of the new subservice type.

The new identity should be based on the "subservice-base" identity. The name of the identity should end with "-type", for instance, "device-type".

The parameters should be defined in a container named "parameters" that augments the choice "/subservices/subservice/parameter" from the main module. The augmentation should be restricted to cases where the type of the subservice matches the identity representing the new service type.

We define two subservice types in the next sections: the "device" subservice type is defined in Section 5 and the "interface" subservice type is defined is Section 6. These subservices can be taken as examples of the rules defined in this section.

Vendors can specify their own subservices types by defining the corresponding modules in their own namespace. An example of such a vendor-specific module is specified in Appendix A. Vendors can also augment existing IETF-specified subservices to add their own vendor-specific information.

5. Subservice Augmentation: "ietf-service-assurance-device" YANG Module

5.1. Tree View

The following tree diagram [RFC8340] provides an overview of the "ietf-service-assurance-device" module.

```
module: ietf-service-assurance-device

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
       +--rw device    string
```

A complete tree view of the base module with all augmenting modules presented in this document is available in Appendix B.3.

5.2. Concepts

As the number of subservices will grow over time, the YANG module is designed to be extensible. A new subservice type requires the precise specifications of its type and expected parameters. Let us illustrate the example of the new device subservice type. As the name implies, it monitors and reports the device health, along with some symptoms in case of degradation.

For our device subservice definition, the new identity "device-type" is specified as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of a device.

The typical parameter for the configuration of the device subservice is the name of the device that we want to assure. By augmenting the parameter choice from the "ietf-service-assurance" YANG module for the case of the "device-type" subservice type, this new parameter is specified.

5.3. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance-device@2023-07-11.yang"

module ietf-service-assurance-device {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device";
  prefix sain-device;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }
```

```
  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf   <mailto:jean.quilbeuf@huawei.com>";
  description
    "This module augments the ietf-service-assurance module with
     support of the device subservice.

     Copyright (c) 2023 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Revised BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 9418; see the
     RFC itself for full legal notices.  ";

  revision 2023-07-11 {
    description
      "Initial revision.";
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  identity device-type {
    base sain:subservice-base;
    description
      "Identity of device subservice.";
  }

  augment "/sain:subservices/sain:subservice/sain:parameter" {
    when "derived-from-or-self(sain:type, 'device-type')";
    description
      "Augments the parameter choice from the ietf-service-assurance
       module with a case specific to the device subservice.";
    container parameters {
      description
        "Parameters for the device subservice type.";
      leaf device {
        type string;
        mandatory true;
        description
          "Identifier of the device to monitor. The
           identifier (e.g., device id, hostname, or management IP)
           depends on the context.";
      }
    }
  }
}
```

```
<CODE ENDS>
```

6. Subservice Augmentation: "ietf-service-assurance-interface" YANG Module

6.1. Tree View

The following tree diagram [RFC8340] provides an overview of the "ietf-service-assurance-interface" data model.

```
module: ietf-service-assurance-interface

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
       +--rw device       string
       +--rw interface    string
```

A complete tree view of the base module with all augmenting modules presented in this document is available in Appendix B.3.

6.2. Concepts

For the interface subservice definition, the new interface-type is specified as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of an interface.

The parameters for the configuration of the interface subservice are the name of the device and, on that specific device, a specific interface. These parameters are aligned with the "ietf-interfaces" model described in [RFC8343], where the name of the interface is the only key needed to identify an interface on a given device. By augmenting the parameter choice from the "ietf-service-assurance" YANG module for the case of the interface-type subservice type, those two new parameters are specified.

6.3. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance-interface@2023-07-11.yang"

module ietf-service-assurance-interface {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface";
  prefix sain-interface;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC 9418: YANG Modules for Service Assurance";
```

```
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf   <mailto:jean.quilbeuf@huawei.com>";
  description
    "This module extends the ietf-service-assurance module to add
     support for the interface subservice.

     It checks whether an interface is healthy.

     Copyright (c) 2023 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Revised BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 9418; see the
     RFC itself for full legal notices.  ";

  revision 2023-07-11 {
    description
      "Initial revision.";
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  identity interface-type {
    base sain:subservice-base;
    description
      "Checks whether an interface is healthy.";
  }

  augment "/sain:subservices/sain:subservice/sain:parameter" {
    when "derived-from-or-self(sain:type, 'interface-type')";
    description
      "Augments the parameter choice from ietf-service-assurance
       module with a case specific to the interface subservice.";
    container parameters {
      description
        "Parameters for the interface subservice type.";
      leaf device {
        type string;
        mandatory true;
        description
          "Device supporting the interface.";
      }
      leaf interface {
        type string;
        mandatory true;
```

```
        description
          "Name of the interface.";
      }
    }
  }
}

<CODE ENDS>
```

7. Security Considerations

The YANG modules specified in this document define schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in these YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /subservices/subservice : By modifying this subtree, one can modify the structure of the assurance graph, which could alter the status of the services reported by the assurance framework. On one hand, modifications can cause the assurance system to report a service as broken when it is actually healthy (false positive), resulting in engineers or automation software losing time and potentially causing real issues by doing unnecessary modifications on the network. On the other hand, modifications could prevent the assurance system from reporting actual issues (false negative), resulting in failures that could have been avoided. Depending on the service, the impact of these avoidable failures could be Service-Level Agreement (SLA) violations fees or disruption of emergency calls.

Some readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /subservices/subservice
- /agents/agent
- /assured-services/assured-service

Each of these subtrees contains information about services, subservices, or possible symptoms raised by the agents. The information contained in this subtree might give information about the underlying network as well as services deployed for the customers. For instance, a customer might be given access to monitor their services status (e.g., via model-driven telemetry). In that example, the customer access should be restricted to nodes representing their services so as not to divulge information about the underlying network structure or others customers services.

8. IANA Considerations

8.1. The IETF XML Registry

IANA has registered the following three URIs in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance
Registrant Contact: The OPSAWG WG of the IETF.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device
Registrant Contact: The OPSAWG WG of the IETF.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface
Registrant Contact: The OPSAWG WG of the IETF.
XML: N/A; the requested URI is an XML namespace.

8.2. The YANG Module Names Registry

IANA has registered the following three YANG modules in the "YANG Module Names" registry [RFC7950]:

name: ietf-service-assurance
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance
prefix: sain
reference: RFC 9418

name: ietf-service-assurance-device
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device
prefix: sain-device
reference: RFC 9418

name: ietf-service-assurance-interface
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface
prefix: sain-interface

reference: RFC 9418

These modules are not maintained by IANA.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9417] Claise, B., Quilbeuf, J., Lopez, D., Voyer, D., and T. Arumugam, "Service Assurance for Intent-Based Networking Architecture", RFC 9417, DOI 10.17487/RFC9417, July 2023, <https://www.rfc-editor.org/info/rfc9417>.

9.2. Informative References

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

Appendix A. Vendor-Specific Subservice Augmentation: "example-service-assurance-device-acme" YANG Module

A.1. Tree View

The following tree diagram [RFC8340] provides an overview of the "example-service-assurance-device-acme" module.

```
module: example-service-assurance-device-acme

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
       +--rw device                     string
       +--rw acme-specific-parameter    string
```

A complete tree view of the base module with all augmenting modules presented in this document is available in Appendix B.3.

A.2. Concepts

Under some circumstances, vendor-specific subservice types might be required. As an example of this vendor-specific implementation, this section shows how to augment the "ietf-service-assurance-device" module to add custom support for the device subservice specific to the Acme Corporation. The specific version adds a new parameter named "acme-specific-parameter". It's an implementation choice to either derive a new specific identity from the "subservice-base" identity defined in the "ietf-service-assurance" module or to augment the parameters from the "ietf-service-assurance-device" module; here, we choose to create a new identity.

A.3. YANG Module

```
module example-service-assurance-device-acme {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-device-acme";
  prefix example-device-acme;
```

```
import ietf-service-assurance {
  prefix sain;
  reference
    "RFC 9418: YANG Modules for Service Assurance";
}
import ietf-service-assurance-device {
  prefix sain-device;
  reference
    "RFC 9418: YANG Modules for Service Assurance";
}

organization
  "IETF OPSAWG Working Group";
contact
  "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
   WG List:  <mailto:opsawg@ietf.org>
   Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
   Author:   Jean Quilbeuf   <mailto:jean.quilbeuf@huawei.com>";
description
  "This example module extends the ietf-service-assurance-device
   module to add specific support for devices of the Acme
   Corporation. ";

revision 2023-07-11 {
  description
    "Initial revision.";
  reference
    "RFC 9418: YANG Modules for Service Assurance";
}

identity device-acme-type {
  base sain-device:device-type;
  description
    "Network device is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'device-acme-type')";
  description
    "Augments the parameter choice from the ietf-service-assurance
     module with a case specific to the device-acme subservice.";
  container parameters {
    description
      "Parameters for the device-acme subservice type.";
    leaf device {
      type string;
      mandatory true;
      description
        "The device to monitor.";
    }
    leaf acme-specific-parameter {
      type string;
      mandatory true;
      description
        "The Acme-Corporation-specific parameter.";
    }
  }
```

```
    }
  }
```

Appendix B. Further Augmentations: IP Connectivity and IS-IS Subservices

In this section, we provide two additional YANG modules to completely cover the example in Figure 2 from Section 3.1 of [RFC9417]. The two missing subservice types are IP connectivity and the Intermediate System to Intermediate System (IS-IS) routing protocol. These modules are presented as examples; some future work is needed to propose a more complete version.

B.1. IP Connectivity Module Tree View

That subservice represents the unicast connectivity between two IP addresses located on two different devices. Such a subservice could report symptoms such as "No route found". The following tree diagram [RFC8340] provides an overview of the "example-service-assurance-ip-connectivity" module.

```
module: example-service-assurance-ip-connectivity

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
       +--rw device1     string
       +--rw address1    inet:ip-address
       +--rw device2     string
       +--rw address2    inet:ip-address
```

To specify the connectivity that we are interested in, we specify two IP addresses and two devices. The subservice assures that the connectivity between IP address 1 on device 1 and IP address 2 on device 2 is healthy.

B.2. IS-IS Module Tree View

The following tree diagram [RFC8340] provides an overview of the "example-service-assurance-is-is" module.

```
module: example-service-assurance-is-is

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
       +--rw instance-name    string
```

The parameter of this subservice is the name of the IS-IS instance to assure.

B.3. Global Tree View

The following tree diagram [RFC8340] provides an overview of the "ietf-service-assurance", "ietf-service-assurance-device", "example-service-assurance-device-acme", "example-service-assurance-ip-connectivity", and "example-service-assurance-is-is" modules.

```
module: ietf-service-assurance
  +--ro assurance-graph-last-change    yang:date-and-time
  +--rw subservices
  |  +--rw subservice* [type id]
  |     +--rw type                                        identityref
  |     +--rw id                                          string
  |     +--ro last-change?
  |     |       yang:date-and-time
  |     +--ro label?                                      string
  |     +--rw under-maintenance!
  |     |  +--rw contact    string
  |     +--rw (parameter)
  |     |  +--:(service-instance-parameter)
  |     |  |  +--rw service-instance-parameter
  |     |  |     +--rw service          string
  |     |  |     +--rw instance-name    string
  |     |  +--:(example-ip-connectivity:parameters)
  |     |  |  +--rw example-ip-connectivity:parameters
  |     |  |     +--rw example-ip-connectivity:device1     string
  |     |  |     +--rw example-ip-connectivity:address1
  |     |  |     |       inet:ip-address
  |     |  |     +--rw example-ip-connectivity:device2     string
  |     |  |     +--rw example-ip-connectivity:address2
  |     |  |             inet:ip-address
  |     |  +--:(example-is-is:parameters)
  |     |  |  +--rw example-is-is:parameters
  |     |  |     +--rw example-is-is:instance-name    string
  |     |  +--:(sain-device:parameters)
  |     |  |  +--rw sain-device:parameters
  |     |  |     +--rw sain-device:device    string
  |     |  +--:(example-device-acme:parameters)
  |     |  |  +--rw example-device-acme:parameters
  |     |  |     +--rw example-device-acme:device
  |     |  |     |       string
  |     |  |     +--rw example-device-acme:acme-specific-parameter
  |     |  |             string
  |     |  +--:(sain-interface:parameters)
  |     |     +--rw sain-interface:parameters
  |     |        +--rw sain-interface:device       string
  |     |        +--rw sain-interface:interface    string
  |     +--ro health-score                                int8
  |     +--ro symptoms-history-start?
  |     |       yang:date-and-time
  |     +--ro symptoms
  |     |  +--ro symptom* [start-date-time agent-id symptom-id]
  |     |     +--ro symptom-id             leafref
  |     |     +--ro agent-id               -> /agents/agent/id
  |     |     +--ro health-score-weight?   uint8
  |     |     +--ro start-date-time        yang:date-and-time
```

```
   |     |     +--ro stop-date-time?        yang:date-and-time
   |     +--rw dependencies
   |        +--rw dependency* [type id]
   |           +--rw type
   |           |       -> /subservices/subservice/type
   |           +--rw id                 leafref
   |           +--rw dependency-type?   identityref
   +--ro agents
   |  +--ro agent* [id]
   |     +--ro id          string
   |     +--ro symptoms* [id]
   |        +--ro id             string
   |        +--ro description    string
   +--ro assured-services
      +--ro assured-service* [service]
         +--ro service      leafref
         +--ro instances* [name]
            +--ro name           leafref
            +--ro subservices* [type id]
               +--ro type    -> /subservices/subservice/type
               +--ro id      leafref
```

B.4. IP Connectivity YANG Module

```
module example-service-assurance-ip-connectivity {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-ip-connectivity";
  prefix example-ip-connectivity;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf   <mailto:jean.quilbeuf@huawei.com>";
  description
    "This example module augments the ietf-service-assurance module
     to add support for the subservice ip-connectivity.

     It checks whether the IP connectivity between two IP addresses
     belonging to two network devices is healthy.";

  revision 2023-07-11 {
    description
```

```
      "Initial version.";
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  identity ip-connectivity-type {
    base sain:subservice-base;
    description
      "Checks connectivity between two IP addresses.";
  }

  augment "/sain:subservices/sain:subservice/sain:parameter" {
    when "derived-from-or-self(sain:type, 'ip-connectivity-type')";
    description
      "Augments the parameter choice from the ietf-service-assurance
       module with a case specific to the ip-connectivity
       subservice.";
    container parameters {
      description
        "Parameters for the ip-connectivity subservice type.";
      leaf device1 {
        type string;
        mandatory true;
        description
          "Device at the first end of the connection.";
      }
      leaf address1 {
        type inet:ip-address;
        mandatory true;
        description
          "Address at the first end of the connection.";
      }
      leaf device2 {
        type string;
        mandatory true;
        description
          "Device at the second end of the connection.";
      }
      leaf address2 {
        type inet:ip-address;
        mandatory true;
        description
          "Address at the second end of the connection.";
      }
    }
  }
}
```

B.5. IS-IS YANG Module

```
module example-service-assurance-is-is {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-is-is";
  prefix example-is-is;

  import ietf-service-assurance {
```

```
    prefix sain;
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf  <mailto:jean.quilbeuf@huawei.com>";
  description
    "This example module augments the ietf-service-assurance module
     to add support for the subservice is-is.

     It checks whether an IS-IS instance is healthy.";

  revision 2023-07-11 {
    description
      "Initial version.";
    reference
      "RFC 9418: YANG Modules for Service Assurance";
  }

  identity is-is-type {
    base sain:subservice-base;
    description
      "Health of IS-IS routing protocol.";
  }

  augment "/sain:subservices/sain:subservice/sain:parameter" {
    when "derived-from-or-self(sain:type, 'is-is-type')";
    description
      "Augments the parameter choice from the ietf-service-assurance
       module with a case specific to the is-is subservice.";
    container parameters {
      description
        "Parameters for the is-is subservice type.";
      leaf instance-name {
        type string;
        mandatory true;
        description
          "The instance to monitor.";
      }
    }
  }
}
```

Appendix C. Example of a YANG Instance

This section contains an example of a YANG instance that conforms to the YANG modules. The validity of this data instance has been checked using yangson. Yangson requires a YANG library [RFC8525] to define the complete model against which the data instance must be validated. In Appendix D, we provide the JSON library file named "ietf-service-assurance-library.json", which we used for validation.

Below, we provide the contents of the file "example_configuration_instance.json", which contains the configuration data that models Figure 2 from Section 3.1 of [RFC9417]. The instance can be validated with yangson by using the invocation "yangson -v example_configuration_instance.json ietf-service-assurance-library.json", assuming all the files (YANG and JSON) defined in this document reside in the current folder.

```
{
  "ietf-service-assurance:subservices": {
    "subservice": [
      {
        "type": "service-instance-type",
        "id": "simple-tunnel/example",
        "service-instance-parameter": {
          "service": "simple-tunnel",
          "instance-name": "example"
        },
        "dependencies": {
          "dependency": [
            {
              "type":
                "ietf-service-assurance-interface:interface-type",
              "id": "interface/peer1/tunnel0",
              "dependency-type": "impacting"
            },
            {
              "type":
                "ietf-service-assurance-interface:interface-type",
              "id": "interface/peer2/tunnel9",
              "dependency-type": "impacting"
            },
            {
              "type":
    "example-service-assurance-ip-connectivity:ip-connectivity-type",
              "id":
                "connectivity/peer1/2001:db8::1/peer2/2001:db8::2",
              "dependency-type": "impacting"
            }
          ]
        }
      },
      {
        "type":
    "example-service-assurance-ip-connectivity:ip-connectivity-type",
        "id": "connectivity/peer1/2001:db8::1/peer2/2001:db8::2",
```

```
        "example-service-assurance-ip-connectivity:parameters": {
          "device1": "Peer1",
          "address1": "2001:db8::1",
          "device2": "Peer2",
          "address2": "2001:db8::2"
        },
        "dependencies": {
          "dependency": [
            {
              "type":
                "ietf-service-assurance-interface:interface-type",
              "id": "interface/peer1/physical0",
              "dependency-type": "impacting"
            },
            {
              "type":
                "ietf-service-assurance-interface:interface-type",
              "id": "interface/peer2/physical5",
              "dependency-type": "impacting"
            },
            {
              "type": "example-service-assurance-is-is:is-is-type",
              "id": "is-is/instance1",
              "dependency-type": "impacting"
            }
          ]
        }
      },
      {
        "type": "example-service-assurance-is-is:is-is-type",
        "id": "is-is/instance1",
        "example-service-assurance-is-is:parameters": {
          "instance-name": "instance1"
        }
      },
      {
        "type": "ietf-service-assurance-interface:interface-type",
        "id": "interface/peer1/tunnel0",
        "ietf-service-assurance-interface:parameters": {
          "device": "Peer1",
          "interface": "tunnel0"
        },
        "dependencies": {
          "dependency": [
            {
              "type":
                "ietf-service-assurance-interface:interface-type",
              "id": "interface/peer1/physical0",
              "dependency-type": "impacting"
            }
          ]
        }
      },
      {
        "type": "ietf-service-assurance-interface:interface-type",
        "id": "interface/peer1/physical0",
        "ietf-service-assurance-interface:parameters": {
          "device": "Peer1",
```

```
          "interface": "physical0"
        },
        "dependencies": {
          "dependency": [
            {
              "type": "ietf-service-assurance-device:device-type",
              "id": "interface/peer1",
              "dependency-type": "impacting"
            }
          ]
        }
      },
      {
        "type": "ietf-service-assurance-device:device-type",
        "id": "interface/peer1",
        "ietf-service-assurance-device:parameters": {
          "device": "Peer1"
        }
      },
      {
        "type": "ietf-service-assurance-interface:interface-type",
        "id": "interface/peer2/tunnel9",
        "ietf-service-assurance-interface:parameters": {
          "device": "Peer2",
          "interface": "tunnel9"
        },
        "dependencies": {
          "dependency": [
            {
              "type":
                "ietf-service-assurance-interface:interface-type",
              "id": "interface/peer2/physical5",
              "dependency-type": "impacting"
            }
          ]
        }
      },
      {
        "type": "ietf-service-assurance-interface:interface-type",
        "id": "interface/peer2/physical5",
        "ietf-service-assurance-interface:parameters": {
          "device": "Peer2",
          "interface": "physical5"
        },
        "dependencies": {
          "dependency": [
            {
              "type": "ietf-service-assurance-device:device-type",
              "id": "interface/peer2",
              "dependency-type": "impacting"
            }
          ]
        }
      },
      {
        "type": "ietf-service-assurance-device:device-type",
        "id": "interface/peer2",
        "ietf-service-assurance-device:parameters": {
```

```
          "device": "Peer2"
        }
      }
    ]
  }
}
```

Appendix D. YANG Library for Service Assurance

This section provides the JSON encoding of the YANG library [RFC8525] that lists all modules defined in this document and their dependencies. This library can be used to validate data instances using yangson, as explained in the previous section.

```
{
  "ietf-yang-library:modules-state": {
    "module-set-id": "ietf-service-assurance@2023-07-11",
    "module": [
      {
        "name": "ietf-service-assurance",
        "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-service-assurance",
        "revision": "2023-07-11",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-service-assurance-device",
        "namespace":
         "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device",
        "revision": "2023-07-11",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-service-assurance-interface",
        "namespace":
      "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface",
        "revision": "2023-07-11",
        "conformance-type": "implement"
      },
      {
        "name": "example-service-assurance-device-acme",
        "namespace":
          "urn:example:example-service-assurance-device-acme",
        "revision": "2023-07-11",
        "conformance-type": "implement"
      },
      {
        "name": "example-service-assurance-is-is",
        "namespace": "urn:example:example-service-assurance-is-is",
        "revision": "2023-07-11",
        "conformance-type": "implement"
      },
      {
        "name": "example-service-assurance-ip-connectivity",
        "namespace":
```

```
            "urn:example:example-service-assurance-ip-connectivity",
          "revision": "2023-07-11",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "revision": "2013-07-05",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "revision": "2013-07-05",
          "conformance-type": "import"
        }
      ]
    }
  }
```

Acknowledgements

The authors would like to thank Jan Lindblad for his help during the design of these YANG modules. The authors would like to thank Stephane Litkowski, Charles Eckel, Mohamed Boucadair, Tom Petch, Dhruv Dhody, and Rob Wilton for their reviews.

Authors' Addresses

Benoit Claise
Huawei
Email: benoit.claise@huawei.com

Jean Quilbeuf
Huawei
Email: jean.quilbeuf@huawei.com

Paolo Lucente
NTT
Siriusdreef 70-72
2132 Hoofddorp
Netherlands
Email: paolo@ntt.net

Paolo Fasano
TIM S.p.A
via G. Reiss Romoli, 274
10148 Torino
Italy
Email: paolo2.fasano@telecomitalia.it

Thangavelu Arumugam
Consultant
Milpitas, California
United States of America
Email: thangavelu@yahoo.com