

YANG-Push status overview

INSA de Lyon

Alex Huang Feng (alex.huang-feng@insa-lyon.fr)
Pierre Francois (pierre.francois@insa-lyon.fr)

Last Updated: 11 september 2023

Table of contents

- [Purpose](#)
- [YANG-push specification](#)
- [YANG-push architecture overview](#)
 - [Configure a Subscription \(management-plane\)](#)
 - [YANG-push messages \(control-plane & data-plane\)](#)
 - [Tracking YANG-model changes](#)
- [YANG-push Open-source projects](#)
 - [Scapy mock generator](#)
 - [Kafka Schema Registry](#)
 - [UDP-notif C-collector & C-publisher](#)
 - [HTTPS-notif C-collector](#)
- [References](#)

Purpose

Purpose of this slide deck

- Track YANG-push development at IETF
 - Discover gaps to be filled from the current specification
- Overview of YANG-push deployment architecture
 - Track open-source projects development
 - List gaps in YANG-push tools
 - Track integration of YANG into the Kafka schema registry
- Track current issues

YANG-push specification

- RFCs, Ongoing drafts
- Dynamic vs. Configured subscriptions
- Stream vs. Datastore
- Issues on current specification

YANG-push related RFCs

- [**RFC6241**](#) – Network Configuration Protocol (NETCONF)
 - ◆ Mechanisms to install, manipulate, and delete the configuration of network devices
- [**RFC8040**](#) – RESTCONF Protocol
 - ◆ HTTP-based protocol using same concepts from NETCONF
- [**RFC5277**](#) – NETCONF Event Notifications
 - ◆ Message notification delivery service for NETCONF
- [**RFC8639**](#) – Subscribed Notifications
 - ◆ Subscription-based mechanism to stream YANG-based notifications
- [**RFC8641**](#) – YANG-push
 - ◆ Solution based on RFC8639 to stream YANG datastores
- [**RFC7223**](#) – A YANG Data Model for Interface Management
 - ◆ YANG data model for Interface configuration and monitoring

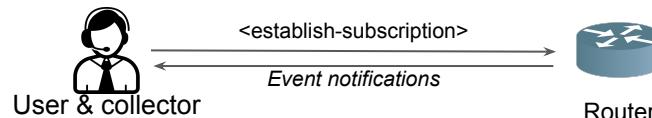
YANG-push related drafts

- [**draft-ahuang-netconf-notif-yang**](#) – YANG model for NETCONF Event Notifications
 - ◆ YANG model allowing encoding Netconf notifications in a YANG-based encoding such as JSON or CBOR
- [**draft-tgraf-netconf-notif-sequencing**](#) – Support of Hostname and Sequencing in YANG Notifications
 - ◆ Extension to YANG-push header to support Hostname, sequence numbers
- [**draft-tgraf-yang-push-observation-time**](#) – Support of Network Observation Timestamping in YANG Notifications
 - ◆ Extension to YANG-push header to support timestamps
- [**draft-tgraf-netconf-yang-notifications-versioning**](#) – Support of Versioning in YANG Notifications Subscription
 - ◆ Extension to YANG-push header to support versioning
- [**draft-ietf-netconf-distributed-notif**](#) – Subscription to Distributed Notifications
 - ◆ Extension to Subscribed Notifications to allow separating the management node from the Network telemetry collector
- [**draft-ietf-netconf-udp-notif**](#) – UDP-based Transport for Configured Subscriptions
 - ◆ UDP-based transport for YANG-push
- [**draft-ietf-netconf-https-notif**](#) – An HTTPS-based Transport for YANG Notifications
 - ◆ HTTPS-based transport for YANG-push

YANG-push variants

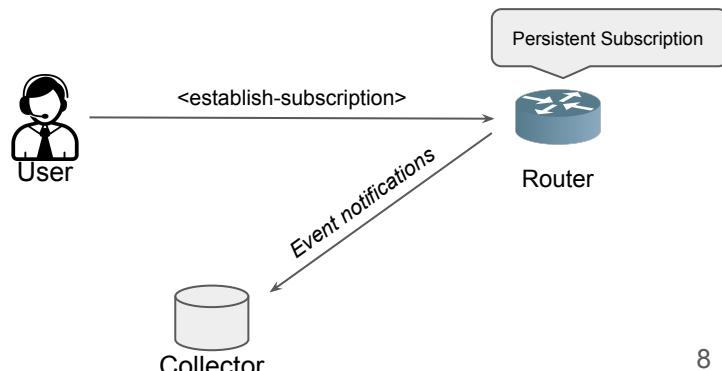
Dynamic Subscriptions ([RFC8639 Section 2.4](#))

- Event Notifications are sent through the same session establishing the subscription
- The subscription and streaming is triggered after the <establish-subscription> RPC



Configured Subscriptions ([RFC8639 Section 2.5](#))

- Event Notifications can be sent through a different session from the session configuring the subscription
- The subscription is persistent across reboots and Notifications can be streamed after start-up once configured through the <establish-subscription> RPC



Subscription configuration: Stream vs. Datastore

Stream (RFC8639)

- [RFC8639] A configuration performed to obtain a flow of events from the publisher
- NIC temperatures
- Netconf events [RFC8639]

```
module: ietf-subscribed-notifications
++-ro streams
| +-+ro stream* [name]
| | +-+ro name
| | | string
| | +-+ro description?
| | | string
| | +-+ro replay-support?
| | | empty {replay}?
| | +-+ro replay-log-creation-time    ietf-yang-types:date-and-time {replay}?
| | +-+ro replay-log-aged-time?      ietf-yang-types:date-and-time {replay}?
| |
| +-+ro stream-filter-ref
| | +-+ro (stream)
| | | +-+rw (stream-filter)?
| | | | +-+:(by-reference)
| | | | | +--+rw stream-filter-name          stream-filter-ref
| | | | +-+:(within-subscription)
| | | | | +-+rw (filter-spec)?
| | | | | | +-+:(stream-subtree-filter)
| | | | | | | +--+rw stream-subtree-filter?    <anydata> {subtree}?
| | | | | | +-+:(stream-xpath-filter)
| | | | | | | +--+rw stream-xpath-filter?      ietf-yang-types>xpath1.0 {xpath}?
| | | | +-+rw stream
| | | +-+ro replay-start-time?
| | | +-+rw configured-replay?
```

Datastore (RFC8641)

- [RFC8641] A configuration performed to obtain a flow of updates happening on a datastore [RFC8342] of the publisher
- An update to a config
<startup>; <candidate>; <running>;
<operational>; <intended>

```
+-+:(ietf-yang-push:datastore)
| +-+rw ietf-yang-push:datastore           identityref
| +-+rw (ietf-yang-push:selection-filter)?
| | +-+:(ietf-yang-push:by-reference)
| | | +-+rw ietf-yang-push:selection-filter-ref  selection-filter-ref
| | +-+:(ietf-yang-push:within-subscription)
| | | +-+rw (ietf-yang-push:filter-spec)?
| | | | +-+:(ietf-yang-push:datatype-filter)
| | | | | +--+rw ietf-yang-push:datatype-filter?    <anydata> {sn:datatype}?
| | | | +-+:(ietf-yang-push:subtree-filter)
| | | | | +--+rw ietf-yang-push:subtree-filter?    <anydata> {sn:subtree}?
| | | | +-+:(ietf-yang-push:datatype-xpath-filter)
| | | | | +--+rw ietf-yang-push:datatype-xpath-filter?  ietf-yang-types:xpath1.0 {sn:xpath}?
```

Specification issues & discussions

1. [Issue #1](#) - To which YANG module is referring an Xpath? (Solved)
2. [Discuss #2](#) - Are YANG-push control-plane messages sent over the same session as the data-plane?
3. [Issue #3](#) - In which encoding is the first message when the encoding is changed in a subscription through <modify-subscription> rpc?
4. [Issue #4](#) - No tools found for validating YANG-push JSON encoded messages
5. [Discuss #5](#) - How to compile a YANG module (including the YANG-push header and payload) for Kafka messages validation?
6. [Issue #6](#) - YANG-JSON does not define how notifications are encoded, YANG 1.1 defines the notifications in [Section 4.1.10](#)

Issue #1: To which YANG module is referring an Xpath?

- /example:root
- /example:root/actors
- /example:root/foo:singers

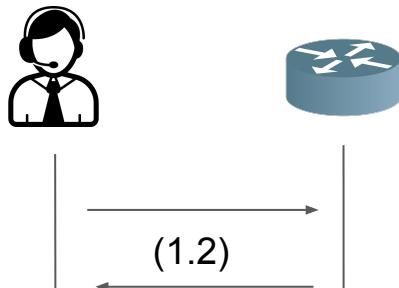
```
<root xmlns:yp="urn:ietf-yang-push" xmlns="urn:example">
  <actors>
    <actor id="1">Christian Bale</actor>
    <actor id="2">Liam Neeson</actor>
    <actor id="3">Michael Caine</actor>
  </actors>
  <foo:singers>
    <foo:singer id="4">Tom Waits</foo:singer>
    <foo:singer id="5">B.B. King</foo:singer>
    <foo:singer id="6">Ray Charles</foo:singer>
  </foo:singers>
</root>
```

“example” is the YANG module name bound to the namespace
“urn:example”

Unwritten rule:

- *An XPath has always the root yang module name*

Issue #1: To which YANG module is referring an Xpath?



```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

After 1.2, there is still no reference to the YANG module name

QUESTION

- *Name of the stream == YANG module name?*
- *The datastore name is associated to a YANG module name by default? How can we find this relation?*

ANSWER

Thomas: *we get the YANG module from the XPath*

Issue #1: To which YANG module is referring an Xpath?

- From RFC8641 Section 3.6

A subscription **must** specify both the selection filters and the datastore[...].

[...]

A publisher **MUST** support at least one type of selection filter.

- YANG Types
 - subtree: anydata
 - xpath: yang>xpath1.0

3.6. Defining the Selection with a Datastore

A subscription must specify both the selection filters and the datastore against which these selection filters will be applied. This information is used to choose and subsequently push data from the publisher's datastore to the receivers.

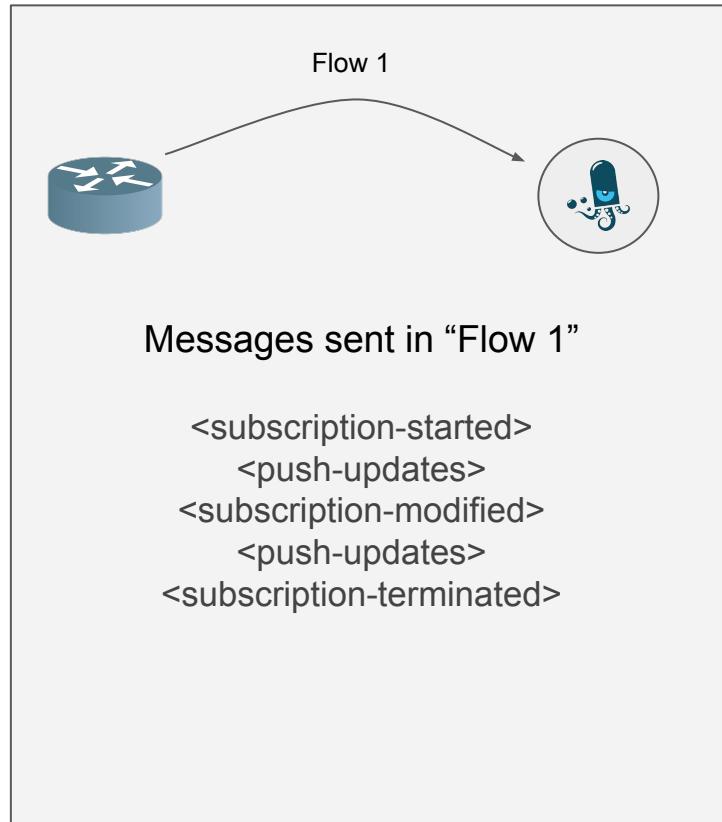
Only a single selection filter can be applied to a subscription at a time. An RPC request proposing a new selection filter replaces any existing filter. The following selection filter types are included in the YANG-Push data model and may be applied against a datastore:

- o subtree: A subtree selection filter identifies one or more datastore subtrees. When specified, update records will only come from the datastore nodes of selected datastore subtree(s). The syntax and semantics correspond to those specified in [\[RFC6241\], Section 6](#).
- o xpath: An "xpath" selection filter is an XPath expression that returns a node set. (XPath is a query language for selecting nodes in an XML document; see [\[XPATH\]](#) for details.) When specified, updates will only come from the selected datastore nodes.

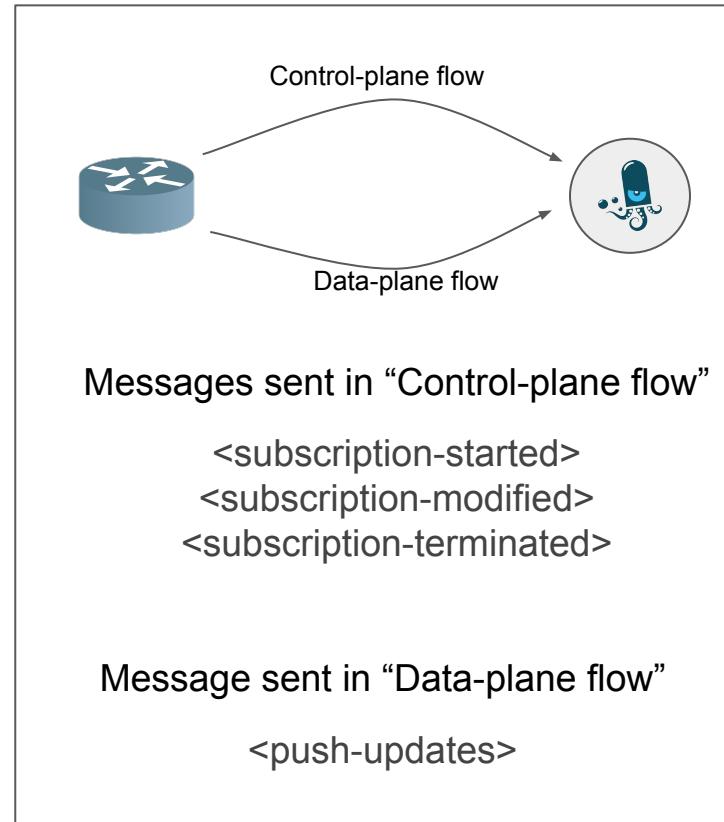
These filters are intended to be used as selectors that define which objects are within the scope of a subscription. A publisher **MUST** support at least one type of selection filter.

We need to assume that the namespace/YANG module name is specified in the Xpath.

Discuss #2 - Are YANG-push control-plane sent in the same session as the data-plane?



VS



Discuss #2 -

Are YANG-push control-plane notifs sent over the same session as the data-plane notifs?

Reading RFC8639 (1)

1. Section 2.7 (Subscription State Change Notifications)

- “[...] they (*subscription state change notifications*) are inserted [...] into the sequence of notification messages sent to a particular receiver.”
- I understand that these notifications are part of the same event notification flow.
- There is no way to separate these flows in the config
 - Though, **dependencies** between subscriptions are allowed (Section 2.3 QoS)

[2.7. Subscription State Change Notifications](#)

In addition to sending event records to receivers, a publisher MUST also send subscription state change notifications when events related to subscription management have occurred.

Subscription state change notifications are unlike other notifications in that they are never included in any event stream. Instead, they are inserted (as defined in this section) into the sequence of notification messages sent to a particular receiver. Subscription state change notifications cannot be dropped or filtered out, they cannot be stored in replay buffers, and they are delivered only to impacted receivers of a subscription. The identification of subscription state change notifications is easy to separate from other notification messages through the use of the YANG extension “`subscription-state-notif`”. This extension tags a notification as a subscription state change notification.

The complete set of subscription state change notifications is described in the following subsections.

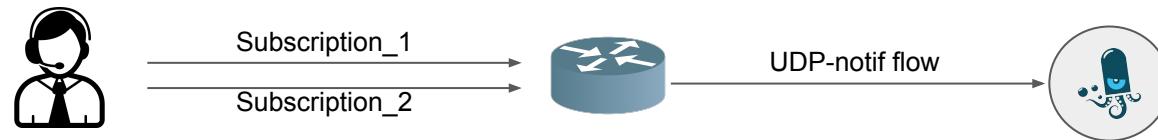
Discuss #2 -

Are YANG-push control-plane notifs sent over the same session as the data-plane notifs?

Reading RFC8639 (2)

2. Section 2.5 (Configured Subscription)

- “[...] *Multiple configured subscriptions MUST be supportable over a single transport session.*”
- This means multiple subscriptions can send messages over the same notification flow?
- Use case: 2 subscriptions sending to the same transport UDP-notif



Agree this is not an issue? Yes

Issue #3 - In which encoding is the first message when the encoding is changed in a subscription through <modify-subscription> rpc?

- Since YANG-push control-plane messages and data-plane are in the same flow, when the user modifies the subscription through the RPC <modify-subscription> changing the encoding, in which encoding does the collector receives the <subscription-modified> notification?
- Other similar corner cases can appear when a subscription is changing. How the collector should manage this changes?

Issue #4 - No tools found for validating YANG-push JSON encoded messages

- During Hackathon 117, Ahmed and Alex did not find any tool to validate YANG encoded data against a YANG module
- Validation of YANG-push messages can be done in XML by using yanglint
 - `yanglint -strict -type nc-notif -f xml notification.yang test.xml`
 - By using this command, a Netconf Notification can be validated as defined in [RFC5277](#)
 - yanglint only validates the presence of `<eventTime>` leaf in the XML
 - Does not work with JSON encoding
- ***Note also that the YANG-push notification header is defined as a structure in [draft-ahuang-netconf-notif-yang](#) and all YANG-push header augmentations will have to augment this structure***
 - Go to [Discuss #5](#) for more details

Discuss #5 - How to compile a YANG module (including the YANG-push header and payload) for Kafka messages validation

- Jean Quilbeuf and Zhouyao Lin implemented a PoC for registering YANG modules in the Kafka schema registry from a Netconf node at the [Hackathon 117](#)
- To validate the YANG message in Kafka Schema registry, a single YANG module needs to be compiled including
 - YANG-push header (and header extensions)
 - Payload (for only the XPath of the subscription and including **augmentations**)
- Proposed solutions from discussions
 - Compile a YANG module including all headers and payloads and register this metadata YANG module to the schema registry
 - Send only the payload without the YANG-push header to Kafka
 - YANG static mount (not even a draft yet)

Issue #6 - YANG-JSON does not define how notifications are encoded, YANG 1.1 defines the notifications in [Section 4.1.10](#)

- YANG 1.1 [RFC7950] defines how a notification should be encoded in Section 4.1.10.
- YANG-JSON [RFC7951] does not define how a YANG notification is encoded in JSON
- Following examples, developers suppose that a notification is encoded as following:

```
{  
  "ietf-notification:notification": {  
    "eventTime": "2023-03-25T08:30:11.22Z",  
    "ietf-notification-sequencing:sysName": "example-router",  
    "ietf-notification-sequencing:sequenceNumber": 1,  
    "ietf-yang-push:push-update": {  
      "id": 6666,  
      "ietf-yang-push-netobs-timestamping:observation-time": "2023-03-25T08:30:11.22Z",  
      "datastore-contents": {  
        "ietf-interfaces:interfaces": [  
          {  
            "interface": {  
              "name": "eth0",  
              "type": "iana-if-type:ethernetCsmacd",  
              "oper-status": "up"  
            }  
          }  
        ]  
      }  
    }  
  }  
}
```

- Should draft-ahuang-netconf-notif-yang fill this gap?

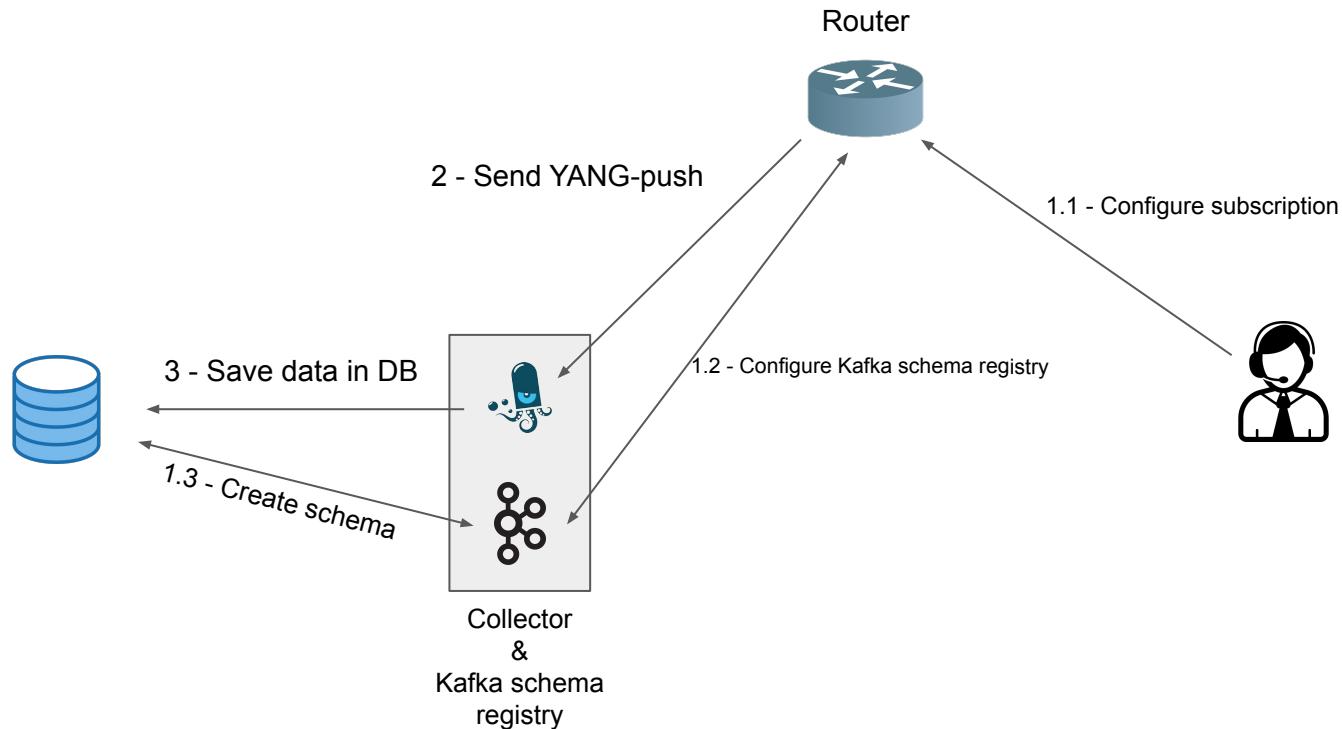
Architecture overview

Focus on integration of
Configured Subscriptions

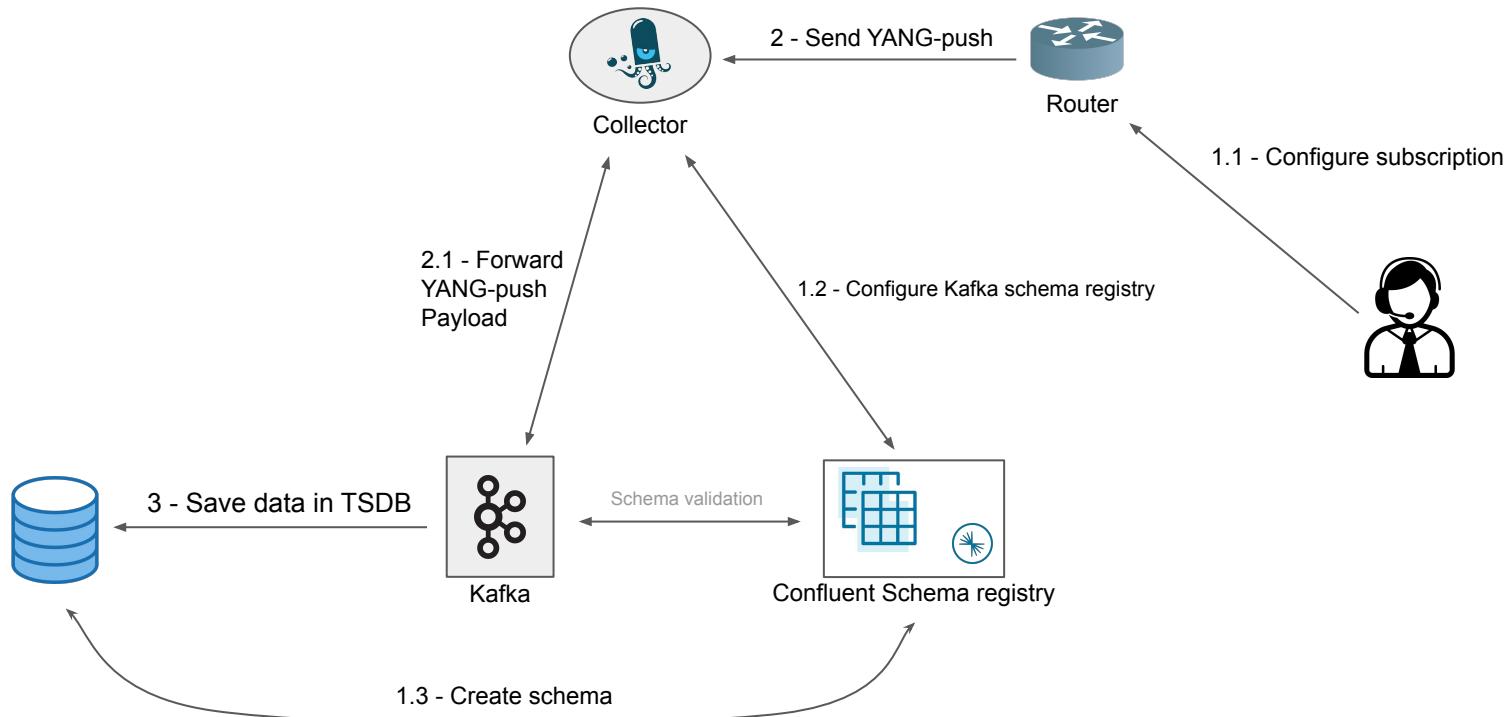
Used tools for demo:

- Kafka
- Kafka schema registry
- pmacct
- TSDB: Druid

High-level Architecture for Configured Subscriptions RFC8639/RFC8641



Implementation-level architecture



Architecture overview

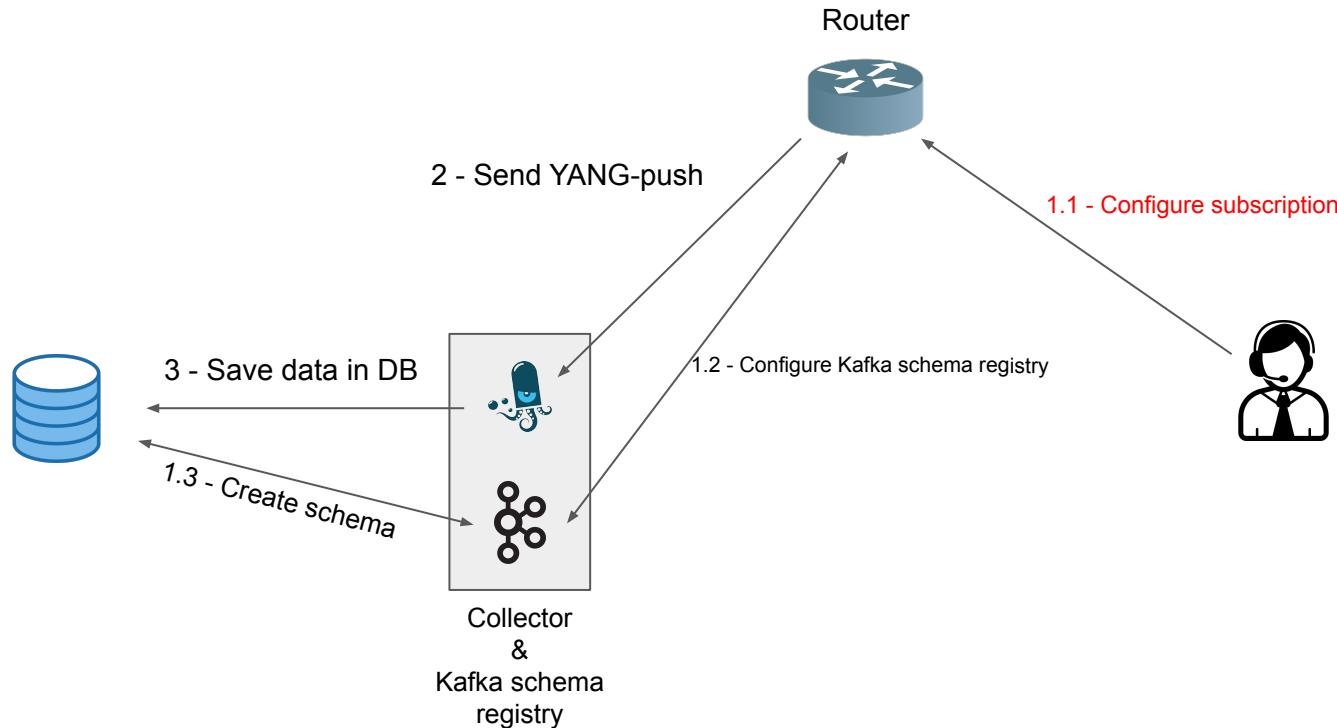
Focus on integration of
Configured Subscriptions

**Configure subscription
(management-plane)**

YANG-Push messages
(control-plane & data-plane)

Tracking YANG-push schema changes

Topology for Configured Subscriptions RFC8639/RFC8641



Configure Subscription with RFC8639: Stream



config tree are wrapped in <edit-config> rpc

(1.1)

```
<?xml version='1.0' encoding='UTF-8'?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <subscription>
      <id>6666</id>
      <stream-subtree-filter>some-subtree-filter</stream-subtree-filter>
      <stream>some-stream</stream>
      <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
      <encoding>encode-json</encoding>
      <receivers>
        <receiver>
          <name>subscription-specific-receiver-def</name>
          <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
        </receiver>
      </receivers>
    </subscription>
    <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">
      <receiver-instance>
        <name>global-udp-notif-receiver-def</name>
        <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
          <address>192.0.5.1</address>
          <port>12345</port>
          <enable-segmentation>false</enable-segmentation>
          <max-segment-size/>
        </udp-notif-receiver>
      </receiver-instance>
    </receiver-instances>
  </subscriptions>
</config>
```

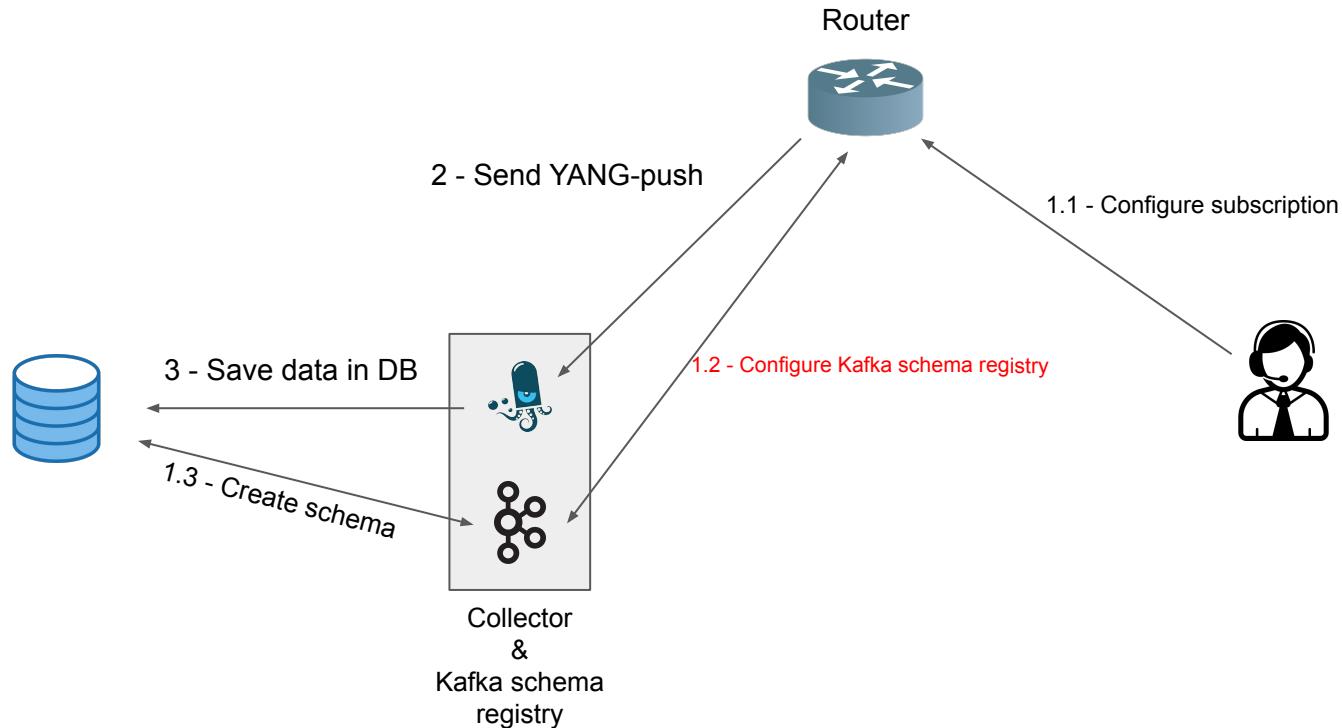
Configure Subscription with RFC8641: Datastore



config tree are wrapped in <edit-config> rpc

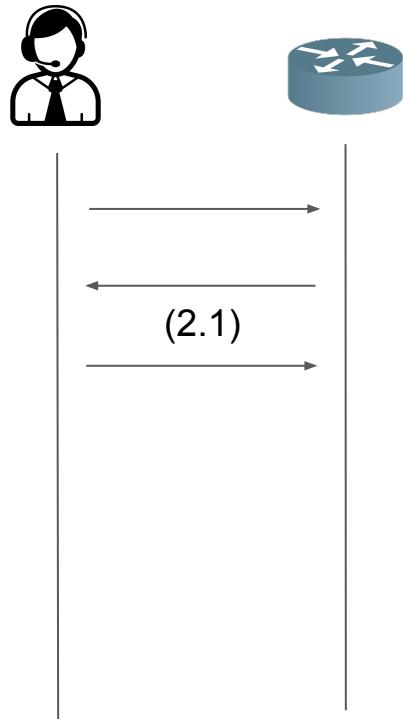
```
<?xml version='1.0' encoding='UTF-8'?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <subscription>
      <id>6666</id>
      <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datastore>
      <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">/some/path</datastore-xpath-filter>
      <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
      <encoding>encode-json</encoding>
      <receivers>
        <receiver>
          <name>subscription-specific-receiver-def</name>
          <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
        </receiver>
      </receivers>
    </subscription>
    <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">
      <receiver-instance>
        <name>global-udp-notif-receiver-def</name>
        <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
          <address>192.0.5.1</address>
          <port>12345</port>
          <enable-segmentation>false</enable-segmentation>
          <max-segment-size/>
        </udp-notif-receiver>
      </receiver-instance>
    </receiver-instances>
  </subscriptions>
</config>
```

Topology for Configured Subscriptions RFC8639/RFC8641



Configure Subscription RFC8639/RFC8641

Obtaining available schemas

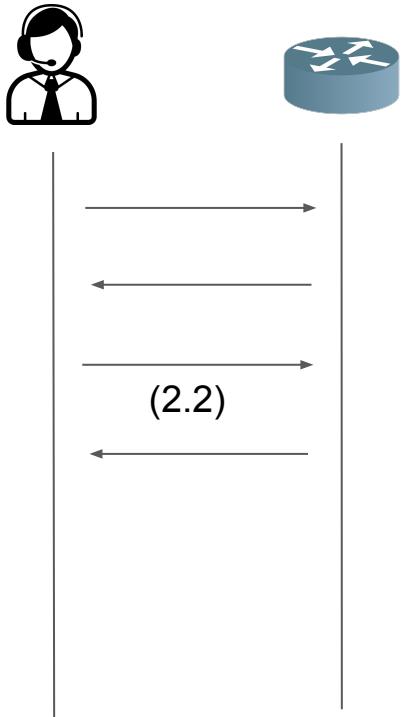


Discovering available YANGs

```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
        |   <schemas/>
      </netconf-state>
    </filter>
  </get>
</rpc>
```

Configure Subscription RFC8639/RFC8641

Obtaining available schemas

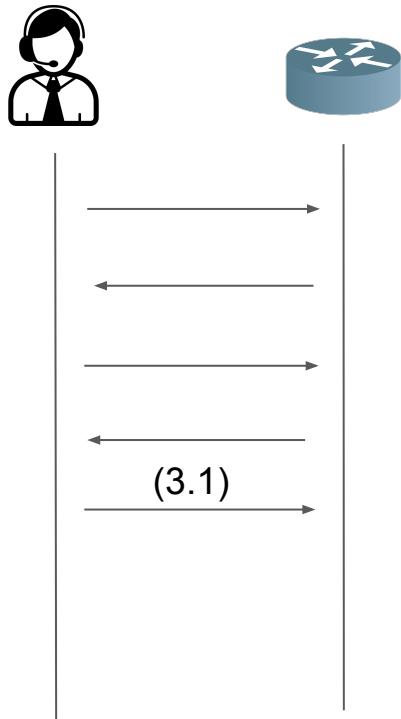


Discovering available YANGs

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
      <schemas>
        <schema>
          <identifier>foo</identifier>
          <version>2020-10-10</version>
          <format>yang</format>
          <namespace>http://example.com/foo</namespace>
          <location>
            http://example.com/schema/foo@2020-10-10.yang
          </location>
          <location>NETCONF</location>
        </schema>
        <schema>
          <identifier>bar-types</identifier>
          <version>2008-06-01</version>
          <format>yang</format>
          <namespace>http://example.com/bar</namespace>
          <location>
            http://example.com/schema/bar-types@2008-06-01.yang
          </location>
          <location>NETCONF</location>
        </schema>
      </schemas>
    </netconf-state>
  </data>
</rpc-reply>
```

Configure Subscription RFC8639/RFC8641

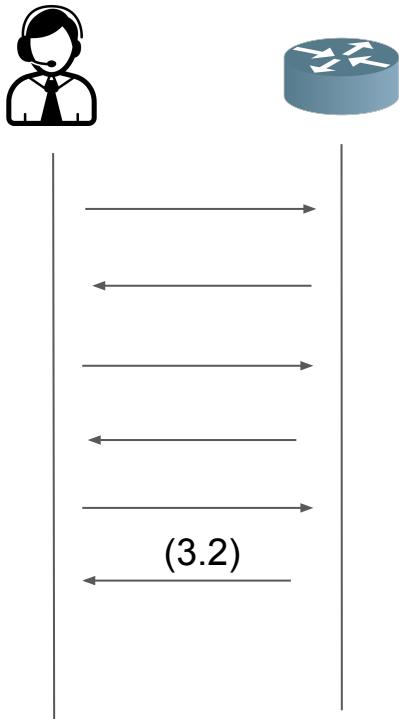
Get schema



```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    <identifier>foo</identifier>
    <version>1.0</version>
    <format>yang</format>
  </get-schema>
</rpc>
```

Configure Subscription RFC8639/RFC8641

Using the schema

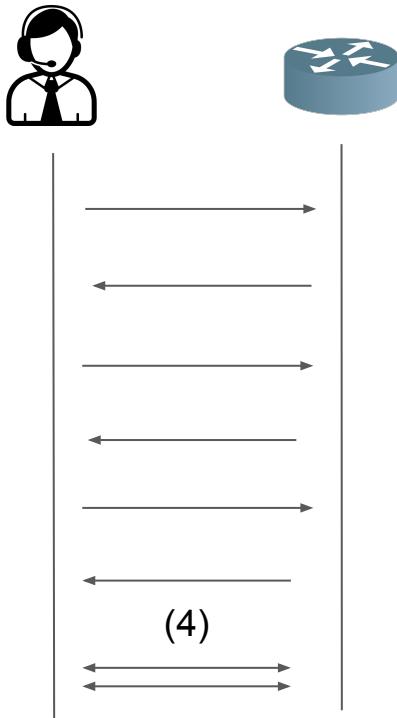


```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    module foo {
      //default format (yang) returned
      //bar version 2020-10-10 yang module
      //contents here ...
    }
  </data>
</rpc-reply>
```

*Once got the YANG module and the Xpath
→ Get recursively the imported YANG modules*

Configure Subscription RFC8639/RFC8641

Recursively get YANG modules



Get all imported YANG modules

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    <identifier>foo</identifier>
    <version>1.0</version>
    <format>yang</format>
  </get-schema>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    module foo {
      //default format (yang) returned
      //bar version 2020-10-10 yang module
      //contents here ...
    }
  </data>
</rpc-reply>
```

Once all the YANG modules (and their Xpath) are obtained
→ Push all these YANG modules to the Schema registry
→ Build one “true” YANG module capturing the actual notification header and message format(s) for the consumer/producer

Architecture overview

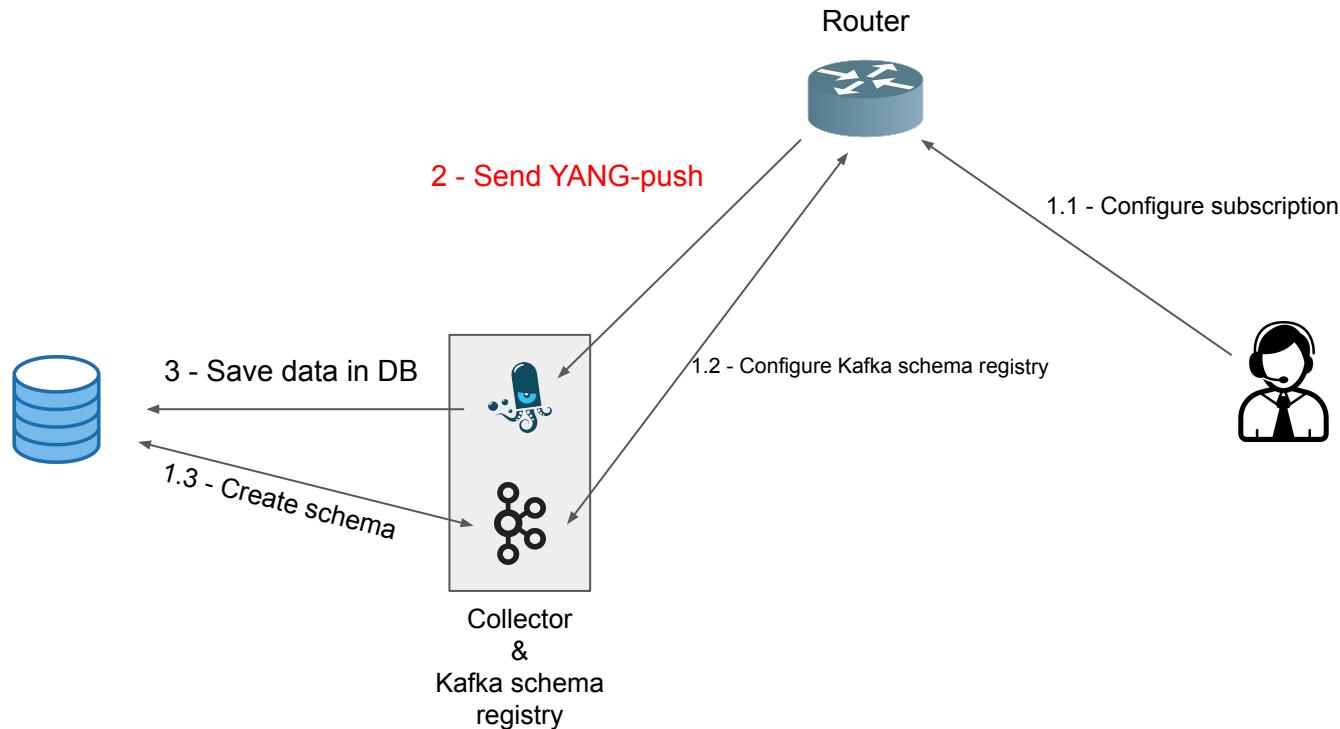
Focus on integration of
Configured Subscriptions

Configure subscription
(management-plane)

**YANG-Push messages
(control-plane & data-plane)**

Tracking YANG-push schema changes

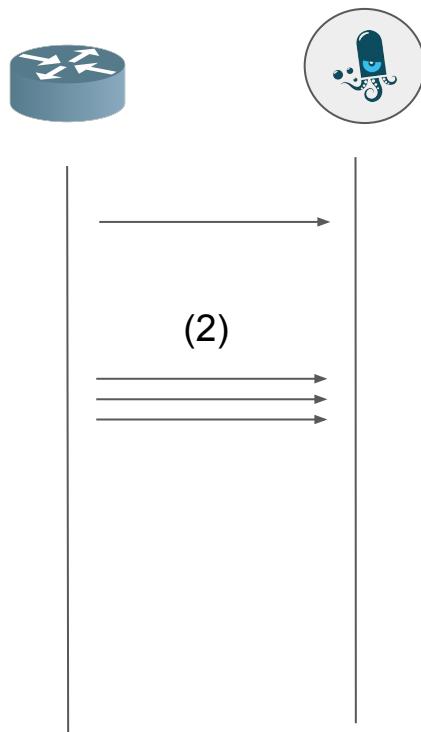
Topology for Configured Subscriptions RFC8639/RFC8641



Sending YANG-push to the collector: Subscription started

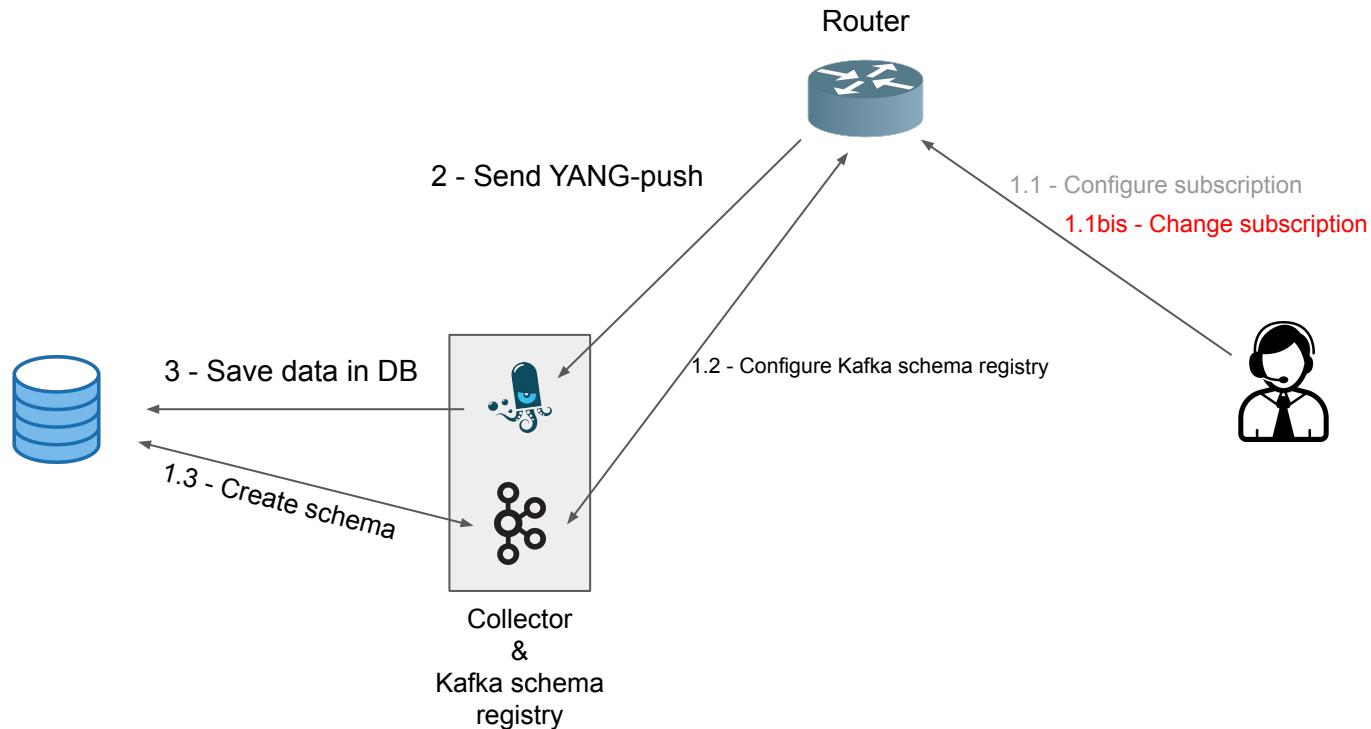


2 - Sending YANG-push to the collector: Updates



```
{  
    "ietf-notification:notification": {  
        "eventTime": "2023-03-25T08:30:11.22Z",  
        "ietf-yang-push:push-update": {  
            "id": 666,  
            "datastore-contents": {  
                "ietf-interfaces:interfaces": [  
                    {  
                        "interface": {  
                            "name": "eth0",  
                            "oper-status": "up"  
                        }  
                    }  
                ]  
            }  
        }  
    }  
}
```

Topology for Configured Subscriptions RFC8639/RFC8641



Change Subscription RFC8639/RFC8641 - Router



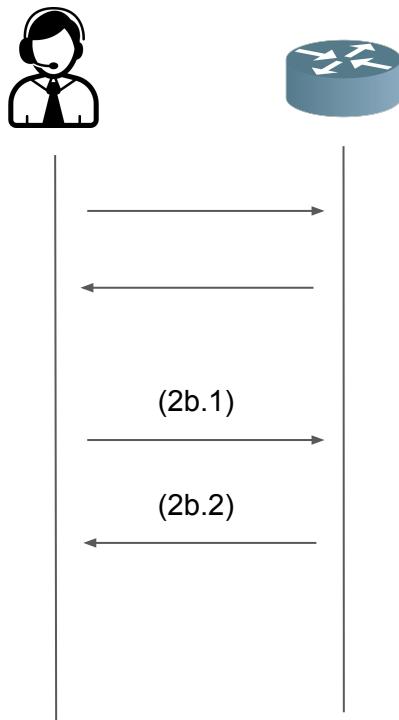
Example: Change encoding

```
<?xml version='1.0' encoding='UTF-8'?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <subscription>
      <id>6666</id>
      <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datastore>
      <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">/some/path</datastore-xpath-filter>
      <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
      <encoding>encode-xml</encoding>
      <receivers>
        <receiver>
          <name>subscription-specific-receiver-def</name>
          <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
        </receiver>
      </receivers>
    </subscription>
    <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">
      <receiver-instance>
        <name>global-udp-notif-receiver-def</name>
        <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
          <address>192.0.5.1</address>
          <port>12345</port>
          <enable-segmentation>false</enable-segmentation>
          <max-segment-size/>
        </udp-notif-receiver>
      </receiver-instance>
    </receiver-instances>
  </subscriptions>
</config>
```

Change Subscription RFC8639/RFC8641 - Router



Change Subscription RFC8639/RFC8641 - Router



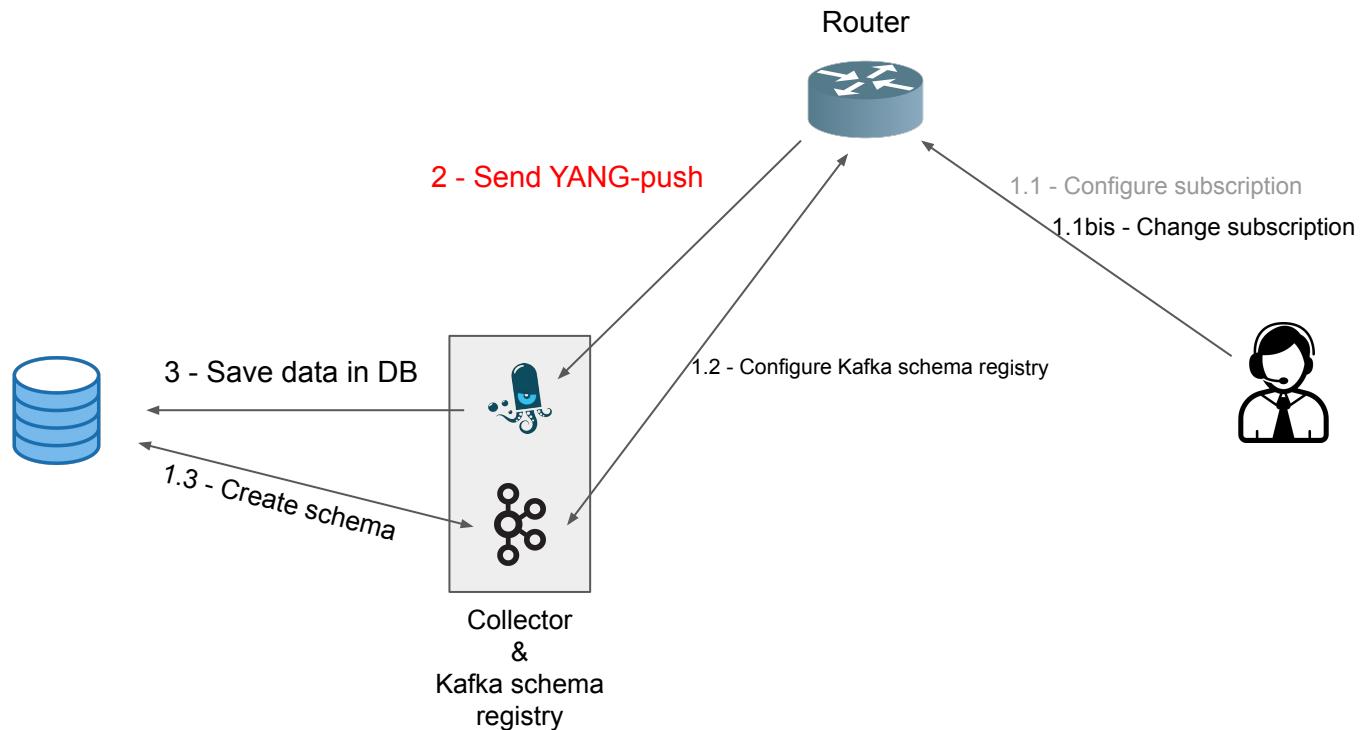
```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    <identifier>foo</identifier>
    <version>1.0</version>
    <format>yang</format>
  </get-schema>
</rpc>
```



```
<rpc-reply message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    module foo {
      //default format (yang) returned
      //bar version 2020-10-10 yang module
      //contents here ...
    }
  </data>
</rpc-reply>
```

Pushing new YANG module to Schema registry

Topology for Configured Subscriptions RFC8639/RFC8641



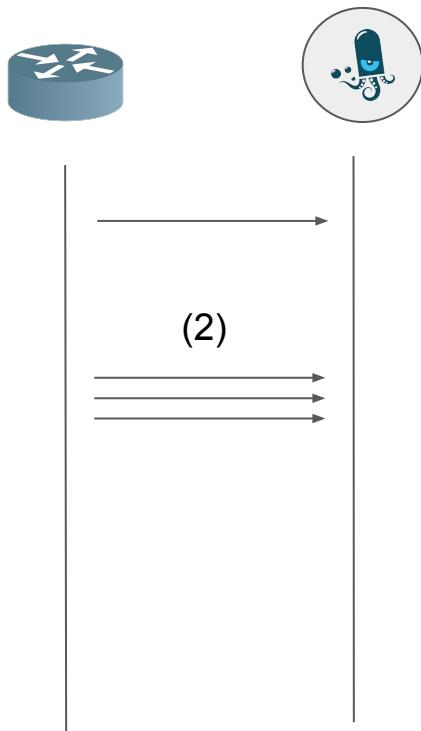
2b - Sending YANG-push to collector



```
{  
    "ietf-notification:notification": {  
        "eventTime": "2023-03-25T08:30:11.22Z",  
        "ietf-subscribed-notification:subscription-modified": {  
            "id": 6666,  
            "target": {  
                "ietf-yang-push: datastore": "ds:operational"  
            },  
            "transport": "ietf-udp-notif-transport:udp-notif",  
            "encoding": "encode-json",  
            "ietf-yang-push:periodic": {  
                "ietf-yang-push:period": 100  
            }  
        }  
    }  
}
```

*This first message is in json or in xml when there is a change on the subscription encoding?
Tracked at Issue #3*

2 - Sending YANG-push to collector



```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-09-02T10:59:55.32Z</eventTime>
  <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>6666</id>
    <datastore-contents>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>eth0</name>
          <oper-status>up</oper-status>
        </interface>
      </interfaces>
    </datastore-contents>
  </push-update>
</notification>
```

Architecture overview

Focus on integration of
Configured Subscriptions

Configure subscription
(management-plane)

YANG-Push messages
(control-plane & data-plane)

Tracking YANG-push schema changes

Tracking YANG model changes from the publisher

2 Options:

- Using a periodic subscription to the XPath **/sn:subscriptions**
- Using on-change subscription to the Xpath **/sn:subscriptions**

Using a periodic subscription to <subscriptions> container



```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
        <subscription>
          <id>2222</id>
          <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datastore>
          <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
            xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">if:interfaces</datastore-xpath-filter>
          <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
          <encoding>encode-xml</encoding>
          <receivers>
            <receivers>
              <name>subscription-specific-receiver-def</name>
              <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
            </receivers>
          </receivers>
          <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
            <period>30000</period>
          </periodic>
        </subscription>
        <subscription>
          <id>6666</id>
          <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datastore>
          <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
            xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">if:interfaces</datastore-xpath-filter>
          <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
          <encoding>encode-json</encoding>
          <receivers>
            <receivers>
              <name>subscription-specific-receiver-def</name>
              <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
            </receivers>
          </receivers>
          <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
            <period>6000</period>
          </periodic>
        </subscription>
        <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">
          <receiver-instance>
            <name>global-udp-notif-receiver-def</name>
            <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
              <address>192.0.5.1</address>
              <port>12345</port>
              <enable-segmentation>false</enable-segmentation>
              <max-segment-size></max-segment-size>
            </udp-notif-receiver>
          </receiver-instance>
        </receiver-instances>
      </subscriptions>
    </config>
  </edit-config>
</rpc>
```

See full example in <https://github.com/network-analytics/udp-notif-scapy/blob/main/src/resources/xml/subscription/edit-config-yang-push-tracking.xml>

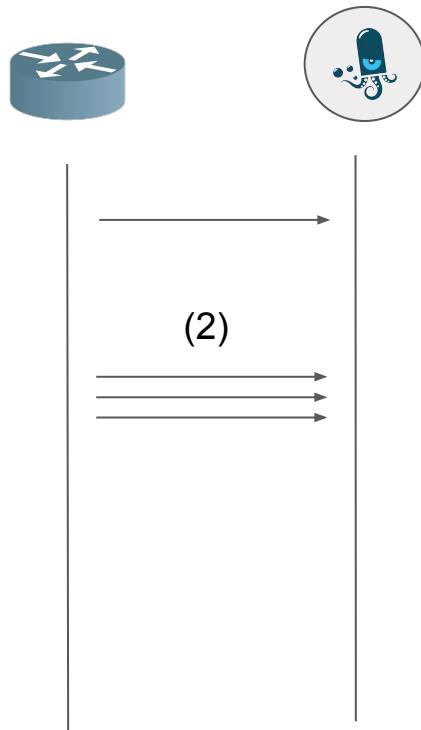
Using a on-change subscription to <subscriptions> container



```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
        <subscription>
          <id>2222</id>
          <datasource xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datasource>
          <datasource-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
            xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">/sn:subscriptions</datasource-xpath-filter>
          <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
          <encoding>encode-xml</encoding>
          <receivers>
            <receiver>
              <name>subscription-specific-receiver-def</name>
              <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
            </receiver>
          </receivers>
          <on-change xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
            <dampening-period>0</dampening-period>
            <sync-on-start>true</sync-on-start>
            <excluded-change>
              <!-- # entries: 0.. -->
            </excluded-change>
            <on-change>
          </subscription>
          <subscription>
            <id>6666</id>
            <datasource xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datasource>
            <datasource-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
              xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">/if:interfaces</datasource-xpath-filter>
            <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
            <encoding>encode-json</encoding>
            <receivers>
              <receiver>
                <name>subscription-specific-receiver-def</name>
                <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
              </receiver>
            </receivers>
            <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
              <period>6000</period>
            </periodic>
          </subscription>
          <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">
            <receiver-instance>
              <name>global-udp-notif-receiver-def</name>
              <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
                <address>192.0.5.1</address>
                <port>12345</port>
                <enable-segmentation>false</enable-segmentation>
                <max-segment-size></max-segment-size>
              </udp-notif-receiver>
            </receiver-instance>
          </receiver-instances>
        </subscriptions>
      </config>
    </edit-config>
  </rpc>
```

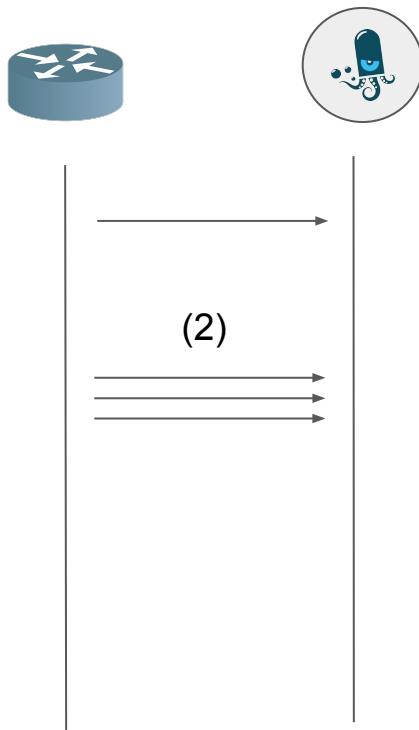
See full example in <https://github.com/network-analytics/udp-notif-scapy/blob/main/src/resources/xml/subscription/edit-config-yang-push-tracking-onchange.xml>

2 - Sending Telemetry YANG-push to collector



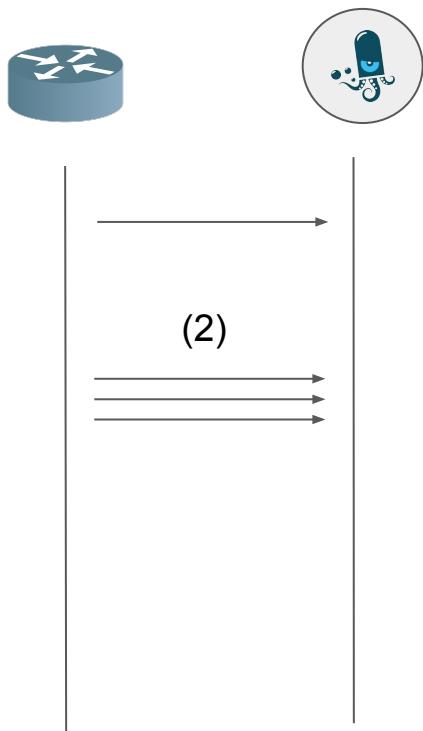
```
{  
    "ietf-notification:notification": {  
        "eventTime": "2023-03-25T08:30:11.22Z",  
        "ietf-yang-push:push-update": {  
            "id": 6666,  
            "datastore-contents": {  
                "ietf-interfaces:interfaces": [  
                    {  
                        "interface": {  
                            "name": "eth0",  
                            "oper-status": "up"  
                        }  
                    }  
                ]  
            }  
        }  
    }  
}
```

2 - Sending Subscription changes periodically



```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
<eventtime>2022-09-02T10:59:55.32Z</eventTime>
<push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
<id>2222</id>
<datastore-contents>
<subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
<subscription>
<id>2222</id>
<datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datastore>
<datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
| xmlns:sn="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">/sn:subscriptions</datastore-xpath-filter>
<transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
<encoding>encode-xml</encoding>
<receivers>
<receiver>
<name>subscription-specific-receiver-def</name>
<receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
</receiver>
</receivers>
<periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
<period>30000</period>
</periodic>
</subscription>
<subscription>
<id>6666</id>
<datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">ds:operational</datastore>
<datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
| xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">/if:interfaces</datastore-xpath-filter>
<transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
<encoding>encode-json</encoding>
<receivers>
<receiver>
<name>subscription-specific-receiver-def</name>
<receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
</receiver>
</receivers>
<periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
<period>6000</period>
</periodic>
</subscription>
<receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">
<receiver-instance>
<name>global-udp-notif-receiver-def</name>
<udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
<address>192.0.5.1</address>
<port>12345</port>
<enable-segmentation>false</enable-segmentation>
<max-segment-size/>
</udp-notif-receiver>
</receiver-instance>
</receiver-instances>
</subscriptions>
</datastore-contents>
</push-update>
</notification>
```

2 - Sending Subscription change using on-change notifications



```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
<eventTime>2022-09-02T10:59:53Z</eventTime>
<push-change-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
  <id>2222</id>
  <datastore-changes>
    <yang-patch>
      <patch-id>patch_54</patch-id>
      <comment>Changing encoding to JSON and increasing the period to 10 minutes</comment>
      <edit>
        <edit-id>id_change_1</edit-id>
        <operation>merge</operation>
        <target xmlns:sn="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">/sn:subscriptions/subscription[id=2222]</target>
        <value>
          <encoding>encode-json</encoding>
          <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
            <period>60000</period>
          </periodic>
        </value>
      </edit>
    </yang-patch>
  </datastore-changes>
</push-change-update>
</notification>
```

YANG-push Open-source projects

- UDP-notif mock generator:
<https://github.com/network-analytics/udp-notif-scapy>
- Kafka Schema Registry: (to be open-sourced)
- UDP-notif c-collector and c-publisher:
<https://github.com/insa-unyte/udp-notif-c-collector>
- HTTPS-notif c-collector:
<https://github.com/network-analytics/https-notif-c-collector>

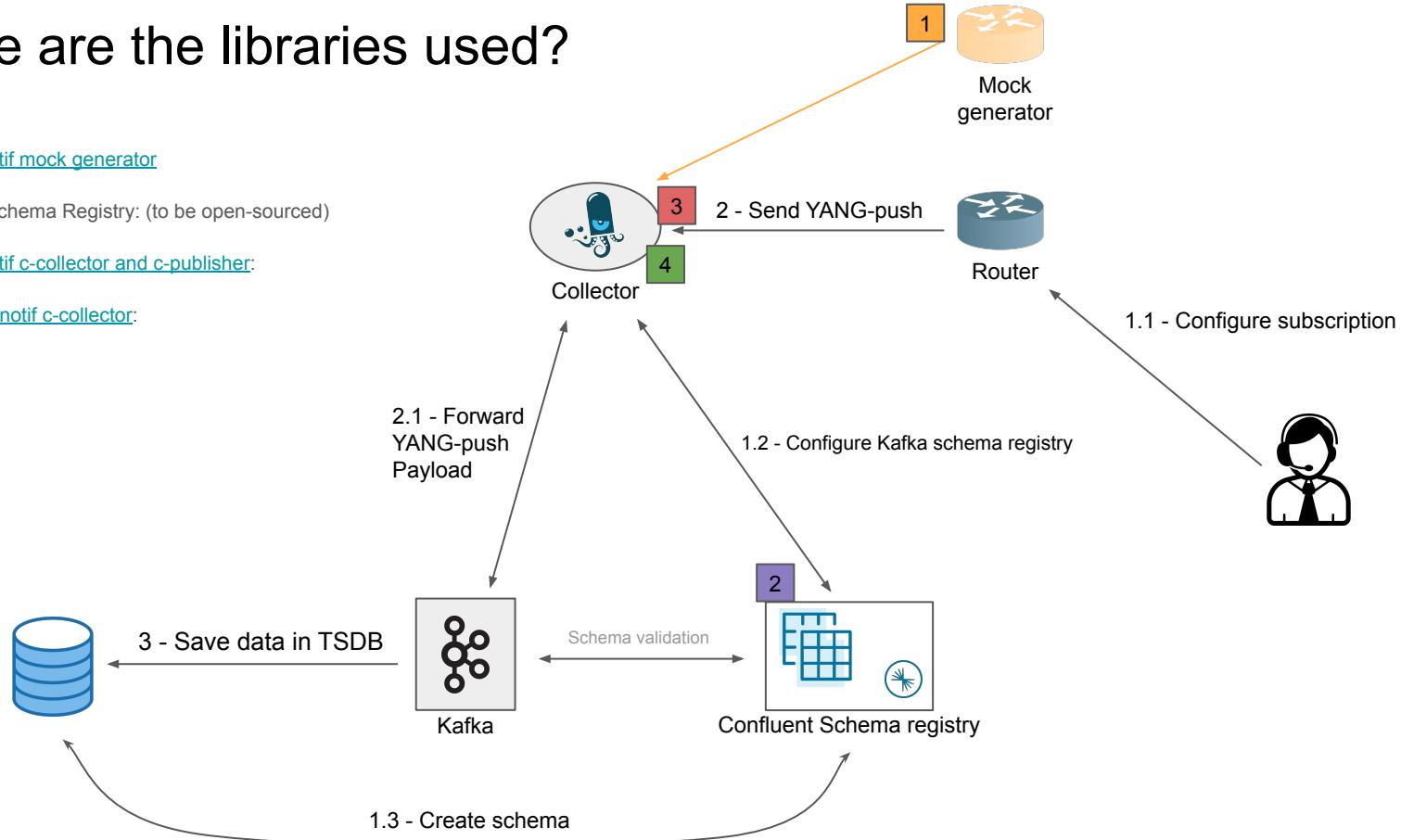
Where are the libraries used?

1 [UDP-notif mock generator](#)

2 Kafka Schema Registry: (to be open-sourced)

3 [UDP-notif c-collector and c-publisher](#):

4 [HTTPS-notif c-collector](#):



YANG-push Open-source projects

- **UDP-notif mock generator:**
<https://github.com/network-analytics/udp-notif-scapy>
- Kafka Schema Registry: (to be open-sourced)
- UDP-notif c-collector and c-publisher:
<https://github.com/insa-unyte/udp-notif-c-collector>
- HTTPS-notif c-collector:
<https://github.com/network-analytics/https-notif-c-collector>

UDP-Notif mock generator

<https://github.com/network-analytics/udp-notif-scapy>



- Goals
- Current Support
- Planned Support
- Implementation details
 - Used YANG module
 - Implemented JSON messages
 - Implemented XML messages



UDP-Notif YANG-push mock generator: Goals

- Mock messages for YANG-push using UDP-Notif transport
 - Data-plane
 - Ctrl-plane
- Support new YANG-push drafts for integration in the global architecture
- Support multiple encodings
 - XML
 - JSON
 - (CBOR)
- Based on Scapy
- MIT License



YANG-push mock generator: Current support

- YANG-push flows
 - Subscription started → Push Updates → Subscription terminated
 - Subscription started → Push Updates → Subscription modified → Push Updates → Subscription terminated
- Supported drafts in YANG-push notifications:
 - draft-ietf-netconf-distributed-notif
 - draft-ahuang-netconf-notif-yang
 - draft-tgraf-netconf-notif-sequencing
 - draft-tgraf-yang-push-observation-time
 - draft-tgraf-netconf-yang-notifications-versioning
 - RFC8639 - Subscribed Notifications
 - RFC8641 - YANG-push
 - RFC7223 - A YANG Data Model for Interface Management
- Transport:
 - UDP-notif (-09): <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-notif-09>
 - UDP pub channel (-05): <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-pub-channel-05>
- Encoding
 - YANG-JSON (RFC7951) and XML (RFC7950) supported



YANG-push Scapy mock generator: Planned

- Encoding
 - **RFC7950 YANG 1.1 XML** (supported)
 - **RFC7951 YANG-JSON** (supported)
 - **RFC9254 YANG-CBOR** (ongoing)

Implementation Details

Used YANG module: **ietf-interfaces.yang [RFC8343]**

ietf-interfaces@2018-02-20.yang

```
module: ietf-interfaces
++-rw interfaces
  +-rw interface* [name]
    +-rw name
    +-rw description?
    +-rw type
    +-rw enabled?
    +-rw link-up-down-trap-enable?
    +-ro admin-status
    +-ro oper-status
    +-ro last-change?
    +-ro if-index
    +-ro phys-address?
    +-ro higher-layer-if*
    +-ro lower-layer-if*
    +-ro speed?
    +-ro statistics
      +-ro discontinuity-time
      +-ro in-octets?
      +-ro in-unicast-pkts?
      +-ro in-broadcast-pkts?
      +-ro in-multicast-pkts?
      +-ro in-discards?
      +-ro in-errors?
      +-ro in-unknown-protos?
      +-ro out-octets?
      +-ro out-unicast-pkts?
      +-ro out-broadcast-pkts?
      +-ro out-multicast-pkts?
      +-ro out-discards?
      +-ro out-errors?
```



ietf-interfaces@2023-04-30.yang

```
module: ietf-interfaces
++-rw interfaces
  +-rw interface* [name]
    +-rw name
    +-rw description?
    +-rw mtu
    +-rw type
    +-rw enabled?
    +-rw link-up-down-trap-enable?
    +-ro admin-status
    +-ro oper-status
    +-ro last-change?
    +-ro if-index
    +-ro phys-address?
    +-ro higher-layer-if*
    +-ro lower-layer-if*
    +-ro speed?
    +-ro statistics
      +-ro discontinuity-time
      +-ro in-octets?
      +-ro in-unicast-pkts?
      +-ro in-broadcast-pkts?
      +-ro in-multicast-pkts?
      +-ro in-discards?
      +-ro in-errors?
      +-ro in-unknown-protos?
      +-ro out-octets?
      +-ro out-unicast-pkts?
      +-ro out-broadcast-pkts?
      +-ro out-multicast-pkts?
      +-ro out-discards?
      +-ro out-errors?
```

New leaf: mtu

UDP-Notif mock generator

<https://github.com/network-analytics/udp-notif-scapy>



- Goals
- Current Support
- Planned Support
- Implementation details
 - Used YANG module
 - **Implemented JSON messages**
 - Implemented XML messages

Implementation Details

Implemented JSON messages – <subscription-started>

```
{  
    "ietf-notification:notification": {  
        "eventTime": "2023-03-25T08:30:11.22Z", } → draft-ahuang-netconf-notif-yang  
        "ietf-notification-sequencing:sysName": "example-router", } → draft-tgraf-netconf-notif-sequencing  
        "ietf-notification-sequencing:sequenceNumber": 1, } → RFC8639 - Subscribed notifications  
        "ietf-subscribed-notification:subscription-started": {  
            "id": 6666, } → RFC8641 - YANG-push  
            "ietf-yang-push:datastore": "ietf-datastores:operational", } → RFC8641 - YANG-push  
            "ietf-yang-push:datastore-xpath-filter": "/if:interfaces", } → draft-tgraf-netconf-yang-notifications-versioning  
            "ietf-yang-push-revision:revision": "2014-05-08", } → draft-ietf-netconf-distributed-notif  
            "ietf-distributed-notif:message-observation-domain-id": [1,2], } → RFC8639 - Subscribed notifications  
            "transport": "ietf-udp-notif-transport:udp-notif", } → RFC8641 - YANG-push  
            "encoding": "encode-json", } → draft-ietf-netconf-distributed-notif  
            "ietf-yang-push:periodic": {  
                "ietf-yang-push:period": 100 } } → RFC8641 - YANG-push  
    } }
```

Implementation Details

Implemented JSON messages – <push-update>

```
{  
    "ietf-notification:notification": {  
        "eventTime": "2023-03-25T08:30:11.22Z",  
        "ietf-notification-sequencing:sysName": "example-router",  
        "ietf-notification-sequencing:sequenceNumber": 1,  
        "ietf-yang-push:push-update": {  
            "id": 6666,  
            "ietf-yang-push-netobs-timestamping:observation-time": "2023-03-25T08:30:11.22Z",  
            "datastore-contents": {  
                "ietf-interfaces:interfaces": [  
                    {  
                        "interface": {  
                            "name": "eth0",  
                            "type": "iana-if-type:etherNetCsmacd",  
                            "oper-status": "up"  
                        }  
                    }  
                ]  
            }  
        }  
    }  
}
```

The diagram illustrates the structure of a YANG-based notification message. Braces group different parts of the message, which are then connected by arrows to their respective specifications:

- A brace groups the top-level notification object, which points to [draft-ahuang-netconf-notif-yang](#).
- A brace groups the 'datastore-contents' field under the 'push-update' object, which points to [RFC8641 - YANG-push](#).
- A brace groups the 'interface' object within the 'interfaces' list, which points to [Monitored Yang: ietf-interfaces.yang](#).
- A brace groups the entire 'push-update' object, which points to [RFC8639 - Subscribed notifications](#).
- A brace groups the 'datastore-contents' field under the 'push-update' object, which points to [draft-tgraf-yang-push-observation-time](#).
- A brace groups the top-level notification object, which also points to [draft-tgraf-netconf-notif-sequencing](#).

Implementation Details

Implemented JSON messages – <subscription-modified>

```
{  
    "ietf-notification:notification": { }, → draft-ahuang-netconf-notif-yang  
    "eventTime": "2023-03-25T08:30:11.22Z", →  
    "ietf-notification-sequencing:sysName": "example-router", → draft-tgraf-netconf-notif-sequencing  
    "ietf-notification-sequencing:sequenceNumber": 1, →  
    "ietf-subscribed-notification:subscription-modified": { }, → RFC8639 - Subscribed notifications  
        "id": 6666, →  
        "ietf-yang-push:datastore": "ietf-datastores:operational", → RFC8641 - YANG-push  
        "ietf-yang-push:datastore-xpath-filter": "/if:interfaces", →  
        "ietf-yang-push:revision": "2014-05-08", → draft-tgraf-netconf-yang-notifications-versioning  
        "ietf-distributed-notif:message-observation-domain-id": [1,2], → draft-ietf-netconf-distributed-notif  
        "transport": "ietf-udp-notif-transport:udp-notif", →  
        "encoding": "encode-json", → RFC8639 - Subscribed notifications  
        "ietf-yang-push:periodic": { }, →  
            "ietf-yang-push:period": 100 } → RFC8641 - YANG-push  
    }  
}
```

Implementation Details

Implemented JSON messages – Updated <push-update>

```
{  
    "ietf-notification:notification": {  
        "eventTime": "2023-03-25T08:30:11.22Z",  
        "ietf-notification-sequencing:sysName": "example-router",  
        "ietf-notification-sequencing:sequenceNumber": 1,  
        "ietf-yang-push:push-update": {  
            "id": 6666,  
            "ietf-yang-push-netobs-timestamping:observation-time": "2023-03-25T08:30:11.22Z",  
            "datastore-contents": {  
                "ietf-interfaces:interfaces": [  
                    {  
                        "interface": {  
                            "name": "eth0",  
                            "type": "iana-if-type:etherNetCsmacd",  
                            "oper-status": "up",  
                            "mtu": 1500  
                        }  
                    }  
                ]  
            }  
        }  
    }  
}
```

The diagram illustrates the structure of an implemented JSON message for an updated <push-update>. The message is a complex object with nested arrays and objects. Callout arrows point from specific parts of the JSON structure to their corresponding specifications:

- "eventTime": "2023-03-25T08:30:11.22Z", "ietf-notification-sequencing:sysName": "example-router", "ietf-notification-sequencing:sequenceNumber": 1, "ietf-yang-push:push-update": { → draft-ahuang-netconf-notif-yang
- "ietf-yang-push:push-update": { → draft-tgraf-netconf-notif-sequencing
- "id": 6666, "ietf-yang-push-netobs-timestamping:observation-time": "2023-03-25T08:30:11.22Z", "datastore-contents": { → RFC8639 - Subscribed notifications
- "datastore-contents": { → draft-tgraf-yang-push-observation-time
- "datastore-contents": { → RFC8641 - YANG-push
- Monitored Yang: ietf-interfaces-02.yang

Scapy mock generator

<https://github.com/network-analytics/udp-notif-scapy>



- Goals
- Current Support
- Planned Support
- Implementation details
 - Used YANG module
 - Implemented JSON messages
 - **Implemented XML messages**

Implementation Details

Implemented XML messages – <subscription-started>

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-09-02T10:59:55.3Z</eventTime>
  <sysName xmlns="urn:ietf:params:xml:ns.yang:ietf-notification-sequencing">example-router</sysName>
  <sequenceNumber xmlns="urn:ietf:params:xml:ns.yang:ietf-notification-sequencing">1</sequenceNumber>
  <subscription-started xmlns="urn:ietf:params:xml:ns.yang:ietf-subscribed-notification">
    <id>6666</id>
    <datastore xmlns="urn:ietf:params:xml:ns.yang:ietf-yang-push"
      xmlns:ds="urn:ietf:params:xml:ns.yang:ietf-datastores">ds:operational</datastore>
    <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns.yang:ietf-yang-push"
      xmlns:if="urn:ietf:params:xml:ns.yang:ietf-ifaces">if:interfaces</datastore-xpath-filter>
    <revision xmlns="urn:ietf:params:xml:ns.yang:ietf-yang-push-revision">2014-05-08</revision>
    <message-observation-domain-id xmlns="urn:ietf:params:xml:ns.yang:ietf-distributed-notif">1</message-observation-domain-id>
    <message-observation-domain-id xmlns="urn:ietf:params:xml:ns.yang:ietf-distributed-notif">2</message-observation-domain-id>
    <transport xmlns:unt="urn:ietf:params:xml:ns.yang:ietf-udp-notif-transport">unt:udp-notif</transport>
    <encoding>encode-xml</encoding>
    <periodic xmlns="urn:ietf:params:xml:ns.yang:ietf-yang-push">
      <period>100</period>
    </periodic>
  </subscription-started>
</notification>
```

The diagram illustrates the structure of an XML notification message for 'subscription-started'. The message is annotated with several IETF standards, each indicated by a blue brace and a corresponding arrow pointing to its definition:

- Event Time:** draft-ahuang-netconf-notif-yang
- System Name:** draft-tgraf-netconf-notif-sequencing
- Sequence Number:** RFC8639 - Subscribed notifications
- Datastore Configuration:** RFC8641 - YANG-push
- Message Observation Domains:** draft-tgraf-netconf-yang-notifications-versioning
- Transport:** draft-ietf-netconf-distributed-notif
- Encoding:** RFC8639 - Subscribed notifications
- Periodic Updates:** RFC8641 - YANG-push

Implementation Details

Implemented XML messages – <push-update>

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-09-02T10:59:55.32Z</eventTime>
  <sysName xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-sequencing">example-router</sysName>
  <sequenceNumber xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-sequencing">1</sequenceNumber>
  <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>6666</id>
    <observation-time xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push-netobs-timestamping">
      2022-09-02T10:59:55.32Z
    </observation-time>
    <datastore-contents>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>eth0</name>
          <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
          <oper-status>up</oper-status>
        </interface>
      </interfaces>
    </datastore-contents>
  </push-update>
</notification>
```

The diagram illustrates the structure of an XML notification message with annotations pointing to relevant standards and documents:

- Top-level notification structure:** Points to [draft-ahuang-netconf-notif-yang](#).
- eventTime:** Points to [draft-tgraf-netconf-notif-sequencing](#).
- sysName and sequenceNumber:** Points to [RFC8639 - Subscribed notifications](#).
- push-update structure:** Points to [draft-tgraf-yang-push-observation-time](#).
- id:** Points to [RFC8641 - YANG-push](#).
- observation-time timestamp:** Points to [Monitored Yang: ietf-interfaces.yang](#).
- datastore-contents / interfaces / interface / name:** Points to [Monitored Yang: ietf-interfaces.yang](#).

Implementation Details

Implemented XML messages – <subscription-modified>

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-09-02T10:59:55.32Z</eventTime>
  <sysName xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-sequencing">example-router</sysName>
  <sequenceNumber xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-sequencing">1</sequenceNumber>
  <subscription-modified xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notification">
    <id>6666</id>
    <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">ds:operational</datastore>
    <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">/if:interfaces</datastore-xpath-filter>
    <revision xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push-revision">2014-05-08</revision>
    <message-observation-domain-id xmlns="urn:ietf:params:xml:ns:yang:ietf-distributed-notif">1</message-observation-domain-id>
    <message-observation-domain-id xmlns="urn:ietf:params:xml:ns:yang:ietf-distributed-notif">2</message-observation-domain-id>
    <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
    <encoding>encode-xml</encoding>
    <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
      <period>100</period>
    </periodic>
  </subscription-modified>
</notification>
```

The diagram illustrates the structure of an XML notification message for 'subscription-modified'. Braces on the left side group various sections of the XML code, which are then connected by arrows to their respective standards or drafts on the right. The groups include: 1) General header information (eventTime, sysName, sequenceNumber); 2) The main notification payload (datastore, datastore-xpath-filter, revision); 3) Message observation domain identifiers; 4) Transport configuration (encoding, periodic); and 5) The subscription-modified element itself.

- draft-ahuang-netconf-notif-yang
- draft-tgraf-netconf-notif-sequencing
- RFC8639 - Subscribed notifications
- RFC8641 - YANG-push
- draft-tgraf-netconf-yang-notifications-versioning
- draft-ietf-netconf-distributed-notif
- RFC8639 - Subscribed notifications
- RFC8641 - YANG-push

Implementation Details

Implemented XML messages – Updated <push-update>

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"> } → draft-ahuang-netconf-notif-yang
  <eventTime>2022-09-02T10:59:55.32Z</eventTime>
  <sysName xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-sequencing">example-router</sysName> } → draft-tgraf-netconf-notif-sequencing
  <sequenceNumber xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-sequencing">1</sequenceNumber>
  <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"> } → RFC8639 - Subscribed notifications
    <id>6666</id>
    <observation-time xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push-netobs-timestamping"> } → draft-tgraf-yang-push-observation-time
      2022-09-02T10:59:55.32Z
    </observation-time>
    <datastore-contents>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>eth0</name>
          <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type> } → Monitored Yang: ietf-interfaces-02.yang
          <oper-status>up</oper-status>
          <mtu>1500</mtu>
        </interface>
      </interfaces>
    </datastore-contents>
  </push-update>
</notification>
```

YANG-push Open-source projects

- UDP-notif mock generator:
<https://github.com/network-analytics/udp-notif-scapy>
- **Kafka Schema Registry**: (to be open-sourced)
- UDP-notif c-collector and c-publisher:
<https://github.com/insa-unyte/udp-notif-c-collector>
- HTTPS-notif c-collector:
<https://github.com/network-analytics/https-notif-c-collector>



Kafka Schema Registry

YANG integration

(To be open-sourced)

- Context
- Goals
- Current Support
- Ongoing Development
- Use-cases to test





Kafka Schema Registry: Context

- Schema registry allows the validation of Kafka messages at the producer and consumer level
- Supports schema versioning and backwards compatibility checks
- Current supported schema encodings:
 - AVRO
 - ProtoBuf
 - JSON Schema

Schema Registry documentation: <https://docs.confluent.io/platform/current/schema-registry/fundamentals/index.html>



Kafka Schema Registry - YANG integration: Goals

- Use YANG as a schema encoding for validating Kafka messages
- Serialization and Deserialization of YANG encodings
 - YANG-JSON ([RFC7951](#))
 - YANG-CBOR ([RFC9254](#))

Kafka Schema Registry: Current Support



- Registration of YANG module developed by Ahmed Elhassany (Swisscom)
- Schema Registry is capable of
 - Register a new YANG module as a new **subject**
 - Validate dependencies from the YANG module before registering the **subject**
- Used YANG parsing library: Yangkit (<https://github.com/yang-central/yangkit>)



Kafka Schema Registry: Ongoing Development

- Serialization and Deserialization of YANG encodings messages
 - YANG-JSON (Ongoing development lead by Alex Huang Feng)
 - YANG-CBOR
- Validation of the YANG message using [Yangkit](#)

Kafka Schema Registry: Use-cases validation @Yangkit

Usecase #1

YANG module: container with 2 leaves

```
module insa-custom {  
    yang-version 1;  
    namespace "urn:ietf:params:xml:ns:yang:insa-yang";  
    prefix iny;  
  
    contact "alex.huang-feng@insa-lyon.fr";  
    organization "INSA Lyon";  
    description "insa-custom yang module";  
  
    revision 2019-09-09 {  
        description "Initial version.";  
    }  
  
    container insa-container {  
        config false;  
        leaf computer {  
            type string;  
            description "computer";  
        }  
        leaf router {  
            type uint8;  
            description "router";  
        }  
    }  
}
```

Message: valid JSON encoded message

```
{  
    "insa-custom:insa-container": {  
        "computer": "computer",  
        "router": 234  
    }  
}
```

Is XPath one of the parameters for Schema Registry?

YANG-push Open-source projects

- UDP-notif mock generator:
<https://github.com/network-analytics/udp-notif-scapy>
- Kafka Schema Registry: (to be open-sourced)
- **UDP-notif c-collector and c-publisher:**
<https://github.com/insa-unyte/udp-notif-c-collector>
- HTTPS-notif c-collector:
<https://github.com/network-analytics/https-notif-c-collector>

UDP-notif C-collector & C-publisher

<https://github.com/insa-unyte/udp-notif-c-collector>

- Goals
- Current Support
- Planned Support



UDP-notif C-collector & C-publisher: Goals



- Reference implementation for the collection of UDP-notif messages in C
- Implementation as a library to ease third parties integration
- MIT Licence



UDP-notif C-collector & C-publisher: Current Support

- Serialization and Deserialization of UDP-notif header
- Support of current and legacy protocol:
 - draft-ietf-netconf-udp-notif (06): <https://datatracker.ietf.org/doc/draft-ietf-netconf-udp-notif/>
 - draft-ietf-netconf-udp-pub-channel (05): <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-pub-channel-05>
- Segmentation reassembly
- IPv4 and IPv6 supported
- Receive multiple message at once using `recvmsg`
- Socket file descriptor as a parameter
- Monitoring of packet drops by the lib

UDP-notif C-collector & C-publisher: Planned Support



- DTLS1.3 encryption as defined in [Section 6](#)



UDP-notif C-collector & C-publisher: Milestones

- First performance tests presented at IETF Hackathon 110
- Library integrated to [pmacct](#)

YANG-push Open-source projects

- UDP-notif mock generator:
<https://github.com/network-analytics/udp-notif-scapy>
- Kafka Schema Registry: (to be open-sourced)
- UDP-notif c-collector and c-publisher:
<https://github.com/insa-unyte/udp-notif-c-collector>
- **HTTPS-notif c-collector:**
<https://github.com/network-analytics/https-notif-c-collector>

HTTPS-notif C-collector

<https://github.com/network-analytics/https-notif-c-collector>

- Goals
- Current Support





HTTPS-notif C-collector: Goals

- Reference implementation for the collection of HTTPS-notif messages in C
- Implementation as a library to ease third parties integration
- MIT License



HTTPS-notif C-collector: Current Support

- Deserialization of HTTPS-notif header as defined in
<https://datatracker.ietf.org/doc/draft-ietf-netconf-https-notif/>
- IPv4 and IPv6 supported
- HTTPS decoding with `microhttpd`

References

- YANG-push
 - RFC8639: Subscription to YANG Notifications
 - RFC8641: Subscription to YANG Notifications for Datastore Updates
- Definition of a Datastore:
 - RFC 8342: Network Management Datastore Architecture (NMDA)
- Encoding:
 - RFC7950: The YANG 1.1 Data Modeling Language (for XML)
 - RFC7951: JSON Encoding of Data Modeled with YANG
 - RFC9254: Encoding of Data Modeled with YANG in CBOR
- YANG-push RPCs and notifications in XML:
 - <https://github.com/network-analytics/udp-notif-scapy/tree/main/src/resources/xml>
- YANG-push notification in JSON:
 - <https://github.com/network-analytics/udp-notif-scapy/tree/main/src/resources/json/notifications>

Some feedbacks from discussions

- One stream is a set of datastore changes
- If we want to change a subscription, it is better to terminate the subscription and create a new one
- No opinion in “on-change” subscription yet
- Scapy implementation: think about not only changing the version, but changing the xpath we are subscribed to