NMOP
Internet-Draft
Intended status: Standards Track
Expires: 2 September 2026

TG. Graf
AE. Elhassany
Swisscom
AHF. Huang Feng
INSA-Lyon
BC. Claise
Everything OPS
PL. Lucente
NTT

1 March 2026

YANG Message Keys for Message Broker Integration
draft-ietf-nmop-yang-message-broker-message-key-01

Abstract

This document specifies a mechanism to define a unique Message key for a YANG to Message Broker integration and a topic addressing scheme based on YANG-Push subscription type and YANG Schema Node Identifier. This enables YANG data consumption of a subset of subscribed YANG data, either per specific YANG data node, identifier or telemetry message type, by indexing and organizing in Message Broker topics. It helps top index the information by using data taxonomy and organizes data in partitions and shards of Message Brokers and time series databases.

Table of Contents

1. Introduction

Nowadays network operators are using machine and human readable YANG [RFC7950] to model their configurations and monitor YANG operational data from their networks according to [Mar24].

Most network analytic use cases require real-time data and the delivery of near real-time analytical and actionable insights. This imposes high scalability, resilience and low overhead in the data processing pipeline. Accessing the right data for the right use case with minimal overhead and in the shortest period of time is therefore crucial.

Network operators organize their data in a Data Mesh [Deh22] according to [Bod24] where a Message Broker, such as Apache Kafka [Kaf11] or Apache Pulsar [Pul16], facilitates the exchange of Messages among data processing components in topics and subjects. Typically, data is being stored in Message Broker topics for several hours or days to facilitate resilience in the data processing chain and addressed in Subjects depending on Schema, enabling a data consumer to address and re-consume previously consumed data again if previously lost.

Dimensional data is structured information in a data store. It uses a model of dimension tables to organize business metrics and their descriptive context. This model, developed by Ralph Kimball [Kim96], simplifies data analysis and reporting by creating denormalized, easy-to-understand structures for quick querying. It is optimized for online analytical processing (OLAP) and data warehouses by using the data taxonomy to scale in partitions and shards. YANG [RFC7950] as a data modelling language based on hierarchical tree-based structures facilitates the modelling of dimensional data. This is best shown with YANG Tree Diagrams [RFC8340].

An Architecture for YANG-Push to Message Broker Integration [I-D.ietf-nmop-yang-message-broker-integration] specifies an architecture for integrating YANG-Push with Message Brokers for a Data Mesh architecture. Section 4.5 of [I-D.ietf-nmop-yang-message-broker-integration] describes how the notification messages at a YANG-Push Receiver are being transformed to the Message Broker while Section 3 of [I-D.ietf-nmop-message-broker-telemetry-message] specifies to a Message Schema to contextualize telemetry data. However, neither of these documents addresses how these messages should be indexed in a Message Broker, nor define how topics, partitioning and sharding must be used.

Due to this missing dimensional indexing for Message Broker stored YANG data, all YANG data is stored in one single Topic. This leads to a round robin distribution across multiple Partitions where each YANG Schema ID is defined as a subject within that topic. Therefore, the entire Topic from all Partitions needs to be consumed first before data selection can be applied. This leads to avoidable data processing overhead which in turn impairs scalability and real-time capabilities, required for certain Network Analytics use cases.

YANG telemetry data can be used for several network analytic use cases. Importantly, depending on the use case, only a subset of the subscribed YANG data might be necessary (in time or space). For example, for specific use cases, it is more important to know the current network state, as opposed to have the full series of the state changes over time. In other use cases, instead of consuming data for all network nodes, only a specific network node or network node component requires the YANG monitoring and hence subscription.

This document defines how YANG Messages [I-D.ietf-nmop-message-broker-telemetry-message] should be indexed and organized in Message Broker topics by leveraging the network node hostname, the YANG datastore name and a YANG Item Identifier for indexing. Then, a YANG-Push subscription type and YANG Schema name for a Message Broker topic naming scheme is defined to better organize YANG data.

Network node hostname, YANG datastore name and subtree and xpath filters are part of "ietf-yang-push-telemetry-message" structured YANG data defined in Section 3 of [I-D.ietf-nmop-message-broker-telemetry-message]. YANG data nodes are derived based on YANG Schema tree applied subtree and xpath filters and the content of each telemetry message.

2. Conventions and Definitions

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.

2.1. Terminology

The following terms are used as defined in [I-D.ietf-nmop-terminology]:

* Network Telemetry

* Network Analytics

* Value

* State

* Change

The following terms are used as defined in [I-D.ietf-nmop-yang-message-broker-integration]:

* Message Broker

* YANG Message Broker Producer

* YANG Message Broker Consumer

* YANG Schema Registry

* YANG Data Consumer

The following terms are used as defined in Apache Kafka [Kaf11] and Apache Pulsar [Pul16] Message Broker:

* Subject: Corresponds to a unique schema tree within a Schema Registry and is used to identify Messages within a Topic.

* YANG Schema ID: A unique ID referencing the schema tree for a subject in schema registry. It is used by the YANG Message Broker Producer To serialize and the YANG Message Broker Consumer to deserialize the message.

* Topic: A communication channel for publishing and subscribing messages with one or more subjects and partitions.

* Topic Compaction: The act of compressing messages in a topic to the latest state. As used with Apache Pulsar. Apache Kafka uses the term Log Compaction with identical meaning.

* Partition: Messages in a topic are spread over hash buckets where a hash bucket refers to a partition being stored within one message broker node. Message ordering is guaranteed within a partition.

* Shard: The same as Partition but distributed among multiple message broker nodes. In this document, the term Partition is being used primarily but the described indexing concept equally applies also to Shards.

* Message: A piece of structured data sent between data processing components to facilitate communication in a distributed system

* Message Key: Metadata associated with a message to facilitate deterministic hash bucketing for instantiated YANG data.

The following terms are used as defined in The Log-Structured Merge-Tree [One96] scientific paper:

* LSM Tree: Log-Structured Merge-Tree is a data structure with performance characteristics that makes it attractive for providing indexed access to files with high insert volume. LSM trees, like other search trees, maintain key-value pairs.

The following terms are used as defined in Confluent Schema Registry Documentation [ConDoc18]:

* Schema: A formalized, documented structure that defines the shape and content of the messages exchange.

* Schema ID: A unique identifier of a schema associated to a Message Broker subject.

* Schema Registry: A system where schemas are registered, compared and retrieved.

The following terms are used as defined in [RFC8641]:

* Periodic Subscription

* On-change Subscription

* Sync-On-Start

* Xpath Filter

* Subtree Filter

The following terms are used as defined in [I-D.ietf-netconf-notif-envelope]:

* Notification

* Hostname

The following terms are used as defined in [RFC8342]:

* Datastore

The following terms are used as defined in [RFC7950]:

* Schema Node Identifier

* Data Node: Such as container, leaf, leaf-list, list, choice and case elements.

* Schema Tree

3. Solution Design

To identify which network node produced which YANG data instance into which Message Broker Topic, Partition and Subject, YANG Message Keys and Indexes (Section 3.1.1) are being introduced. These keys enable a deterministic distribution of YANG messages accross Topics and Partitions enabling applications to consume only the needed data from specific topics and partitions.

In order to facilitate Message Broker Topic Compaction, a YANG-Push subscription type based topic naming scheme (Section 3.2) is defined. This segregates statistical (Value), State and State change YANG metrics and facilitates a YANG Message Broker Consumer to use the Topic wild card consumption method to select based on YANG-Push subscription type.

3.1. YANG Message Keys and Indexes

For topics that carry YANG telemetry messages as defined in [I-D.ietf-nmop-message-broker-telemetry-message], a Message Key MUST be used. If no Message Key is defined then the Messages are distributed in a round robin fashion across partitions. If a Message Key is defined, then the value of the Message Key is being used as input for the Message Broker Producer hash function to distribute across Partitions. Therefore, Message Keys facilitate Message deterministic distribution.

The Message Key not only used for Message indexing at the Message Producer but also at the Message Broker for topic compaction.

For YANG, the network node hostname, from which YANG datastore the YANG metrics are published from and the YANG data nodes are used to generate the Message Key.

The following sections describe how Message Keys are used in both Message producers and Message consumers.

3.1.1. YANG Message Broker Producer

YANG data nodes are uniquely identifiable within the YANG Schema tree. Section 6.5 of [RFC7950] defines with "absolute-schema-nodeid" how absolute YANG Schema node identifiers are being crafted locally unique to the YANG module and how YANG data nodes are associated.

Section 3.6 of [RFC8641] defines how YANG data nodes can be subscribed with subtree and xpath selection filters. A YANG-Push publisher publishes with "subscription-started" state notifications for each subscription which filter and filter type is being used to the YANG-Push receiver.

To generate the Message Key, the "absolute-schema-nodeid" (see Section 6.5 of [RFC7950]) must be extracted from the YANG-Push subtree or xpath subscription filter in use. If the identifier refers to a YANG list (see Section 7.8 of [RFC7950]) the list key (Section 7.8.2 of [RFC7950]) is appended to the identifier, separated by a slash.

For example, given the XPath filter shown in Figure 1, the "absolute-schema-nodeid" is "interfaces/interface". Because the interface list has a key named name, the resulting keys for the Message Key are "interfaces/interface/name" plus the YANG data node name of the list which is in this case the name of the interface.

```
ietf-interface:interfaces/interface[type='ianaift:ethernetCsmacd']
```

Figure 1: YANG-Push ietf-interface Xpath Filter Example

For example, if the following subtree filter is being used, the "absolute-schema-nodeid" is "hardware/component/state". Therefore, the keys used for the Message Key generation are "hardware/component/name/state" plus the YANG data node name of the list which is in this case the name of the component.

```
<get>
  <filter type="subtree">
    <hardware xmlns="urn:ietf:params:xml:ns:yang:ietf-hardware">
      <component>
        <state/>
      </component>
    </hardware>
  </filter>
</get>
```

Figure 2: YANG-Push ietf-hardware Subtree Filter Example

When the Message is being produced to the Message Broker, the Network node hostname and YANG datastore name is used from the structured YANG data defined in "ietf-yang-push-telemetry-message" Section 3 of [I-D.ietf-nmop-message-broker-telemetry-message] where the YANG "absolute-schema-nodeid" with the optional list key is derived from subtree and xpath filters, respectively from their YANG Schema tree.

3.1.2. YANG Message Broker Consumer

The consumer hashes the Message寰ッKey, applies modulo with the number of partitions, and determines the partition from which it should consume messages bearing that Message寰ッKey.

At a YANG data store, such as a Time Series database or stream processor, the YANG data could than be ingested into tables according to topic names and indexed per Message Key. If Topic Compaction is enabled, only current state is consumed.

3.1.3. Time Series Database

Depending if the YANG Data Consumer knows the Message Key from the YANG Message Broker Consumer or the YANG Schema from the YANG Schema Registry the network telemetry messages can be indexed in a Time series database. The Message Key could serve as the primary key, while keys from the YANG data taxonomy can be reflected in the indexing scheme using primary and secondary keys in a time series database. Implementation examples can be found under Section 5.

3.2. YANG-Push Message Broker Topic Naming

YANG data can be subscribed "periodic", on-change" or "on-change" with "sync-on-start". Periodic subscriptions are used for obtaining statistical metrics. On-Change subscriptions are used for obtaining State Changes and on-change with sync-on-start is used for obtaining States.

Message Brokers topics are addressed with a unique name. Usually topics are named hierarchically similar to the DNS namespace where "." delimits hierarchies.

This document defines "statistics", "states" and "state-changes" in the topic name as the first part to denote the types of data. Followed by "yang" to denote YANG data. Followed by the Section 6.5 of YANG prefix [RFC7950] and Section 7.1.4 of absolute-schema-nodeid [RFC7950] where all subsequent "/" are substituted by "_".

For example, if the "ietf-interface:interfaces/interface" xpath filter is being used, the Message Broker topic name would be as following. In the example the project name and environment (prod, dev, test etc.) is prefixed.

project.environment.statistics.yang.if.interfaces_interface

Figure 3: YANG-Push ietf-interface Topic Name Example

3.2.1. YANG Message Broker Producer

For Message寰ッBroker topic creation, the "periodic", "on-change", and "sync-on-start" update triggers contain data defined by the 寰徸etf-subscribed-notifications寰YANG module (Section 4.1 of [RFC8641]). Subscription state notifications MUST be used to derive the subscribed YANG data when it represents "statistics", "states", or "state-changes". The YANG "absolute-schema-nodeid" MUST be derived from subtree and xpath filter data of subscription state notifications, respectively from their YANG Schema tree.

3.2.2. YANG Message Broker Consumer

The consumer can use a wildcard (*) in the topic name to subscribe to multiple topics.

For example, if YANG states should be consumed and indexed in Time Series database or stream processor than below Topic Name could be used, and the YANG data could be ingested into tables according to topic names and indexed per Message Key. If Topic Compaction is enabled, only current state is consumed.

project.environment.states.yang.*

Figure 4: YANG-Push Wildcard Topic Name Example

4. Message Broker Implementations

Topic, Partitioning and Message Keys are generic concepts of Message Brokers. There are two known Message Broker implementations supporting all features described in this document.

4.1. Apache Kafka

Apache Kafka supports Message Keys, Partitioning and Log Compaction.

With the following example from the Apache Kafka admin client API https://kafka.apache.org/41/javadoc/org/apache/kafka/clients/admin/Admin.html a new compacted Topic can be created.

```
Properties props = new Properties();
props.put(AdminClientConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");

try (Admin admin = Admin.create(props)) {
 String topicName = "my-topic";
 int partitions = 12;
 short replicationFactor = 3;
 // Create a compacted topic
 CreateTopicsResult result = admin.createTopics(Collections.singleton(
  new NewTopic(topicName, partitions, replicationFactor)
   .configs(Collections.singletonMap(TopicConfig.CLEANUP_POLICY_CONFIG,/
   TopicConfig.CLEANUP_POLICY_COMPACT))));

 // Call values() to get the result for a specific topic
 // KafkaFuture<Void> future = result.values().get(topicName);

 // Call get() to block until the topic creation is complete or has
 // failed if creation failed the ExecutionException wraps the
 // underlying cause. future.get();
}
```

The most important configuration items from https://kafka.apache.org/41/configuration/topic-configs/ are "topicName" defines the Topic name, "partitions" the amount of partitions, "replicationFactor" how many times the partition is being replicated.

With "compact" in "cleanup.policy" the log compaction can be turned on per topic. With "min.cleanable.dirty.ratio" and "delete.retention.ms" how often and when Log Compaction should occur per topic. Where with "retention.bytes" and with "retention.ms" the topic specific compaction configurations can be limited how often the topics are compacted.

The topic names are constrained to 249 character length and the following characters: "a-z", "A-Z", "0-9", ".", "_" and "-". Topics can be created on the fly by producing into a new Topic when "auto.create.topics.enable" has been configured prior. Topics should be deleted at the end of the lifecycle through the "kafka-topics.sh" command.

The Partition count for a given Topic can be increased but not decreased. Consumer groups are automatically re-joined and partitions are being rebalanced on Message Broker nodes when Partition count changed.

4.2. Apache Pulsar

Apache Pulsar supports Message Keys, Partitioning and Topic Compaction.

With "brokerServiceCompactionThreshold" when Topic Compaction should occur is being configured.

The topic names allow all characters except: "/". Topics can be created on the fly by producing into a new Topic when "allowAutoTopicCreation" has been configured prior. Topics should be deleted at the end of the lifecycle through pulsar-admin or pulsarctl tools.

The Partition count for a given Topic can be increased but not decreased. Consumer groups are automatically re-joined and partitions are being rebalanced on Message Broker nodes when Partition count changed.

5. Time Series Database Implementations

Tables, partition and keys are generic concepts of time series databases. With ClickHouse, this document provides examples of how YANG message keys can be obtained from the Message Broker and used for indexing.

5.1. ClickHouse

5.1.1. Data Model

Unlike other realtime analytics databases, ClickHouse does not (necessarily) rely on partitioning data by timestamp. ClickHouse represents data in the MergeTree format, which is similar to a LSM tree:

A table consists of data parts sorted by primary key.

When data is inserted in a table, separate data parts are created and each of data part is lexicographically sorted by primary key. For example, if the primary key is ("MessageKey", "Date"), the data in the part is sorted by "MessageKey", and within each "MessageKey", it is ordered by "Date".

Data belonging to different partitions are separated into different parts. In the background, ClickHouse merges data parts for more efficient storage. Parts belonging to different partitions are not merged. The merge mechanism does not guarantee that all rows with the same primary key will be in the same data part.

Each data part is logically divided into granules. A granule is the smallest indivisible data set that ClickHouse reads when selecting data. ClickHouse does not split rows or values, so each granule always contains an integer number of rows. The first row of a granule is marked with the value of the primary key for the row. For each data part, ClickHouse creates an index file that stores the marks. For each column, whether it's in the primary key or not, ClickHouse also stores the same marks. These marks let you find data directly in column files.

Thus, it is possible to quickly run queries on one or many ranges of the primary key.

5.1.2. Message Broker Integration

ClickHouse integrates with Message Brokers through Integration Table Engines.

Reading (selecting) data through Kafka Table Engine follows Apache Kafka semantics of advancing the offset, so subsequent reads will start at the offset the previous read left off.

It is the responsibility of the data model designer to transfer data to a regular table:

* Use the engine to create a Kafka consumer and consider it a data stream.

Example:

```
CREATE TABLE queue (
    timestamp UInt64,
    level String,
    message String
)
ENGINE = Kafka
SETTINGS kafka_broker_list = 'localhost:9092',
    kafka_topic_list = 'topic',
    kafka_group_name = 'group1',
    kafka_format = 'JSONEachRow',
    kafka_num_consumers = 4;
```

* Create a table with the desired structure.

Example:

```
CREATE TABLE messages (
    key String,
    timestamp UInt64,
    level String,
    message String
)
ENGINE = MergeTree
ORDER BY (key, timestamp);
```

* Create a materialized view that converts data from the engine and puts it into a previously created table.

```
CREATE MATERIALIZED VIEW mv_messages TO messages AS
SELECT
    _key AS key,
    timestamp,
    level,
    message
FROM queue;
```

The Message Key and partition ID are available as virtual (read only) columns _key and _partition.

5.1.3. Message Formats

ClickHouse supports numerous Message formats natively. The example above uses the JSON Lines format but other (binary) formats, such as Apache Avro or Protobuf, are supported as well.

5.1.4. Schema Registry

ClickHouse has built in Schema Registry support. For Apache Avro, the Schema Registry and authentication are encoded in additional parameters to the Apache Kafka consumer.

For formats such as Confluent JSON_SR, use the "kafka_schema_registry_skip_bytes" parameter to skip reading the Schema Registry preamble. The Schema can then be encoded explicitly.

6. IANA Considerations

This document includes no request to IANA.

7. Security Considerations

This document should not affect the security of the Internet.

8. Operational Considerations

The YANG Message Broker Producer of a YANG-Push receiver should have three config knobs facilitate the features described in this document as optional:

* Topic Distribution: Select between "topic" and "subject" distribution. Default is subject to remain backward compatibility to [I-D.ietf-nmop-yang-message-broker-integration].

* Distribution Type: Select between "none" and "YANG-Push subscription type".

* YANG Message Key: Select between "enable" and "disable".

Subject distribution enables message ordering for a set of YANG Message Keys on each partition. Where in topic distribution messages are randomly being distributed among partitions.

To accommodate for potential date loss throughout the data processing pipeline, periodic update of the current State for State metrics is RECOMMENDED. This can be accommodated with YANG-Push as defined in [RFC8641] by complementing "on-change sync on start" subscriptions with "periodic" subscriptions. Alternatively, in YANG-Push Lite defined in Section 7.6 of [I-D.wilton-netconf-yang-push-lite] this simplified in one subscription.

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>.

[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>.

[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>.

[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>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

[I-D.ietf-nmop-message-broker-telemetry-message] Elhassany, A., Graf, T., and P. Lucente, "Extensible YANG Model for Network Telemetry Messages", Work in Progress, Internet-Draft, draft-ietf-nmop-message-broker-telemetry-message-04, 18 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-message-broker-telemetry-message-04>.

[I-D.ietf-netconf-notif-envelope] Feng, A. H., Francois, P., Graf, T., and B. Claise, "Extensible YANG Model for YANG-Push Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-notif-envelope-04, 6 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-notif-envelope-04>.

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>.

[I-D.ietf-nmop-terminology] Davis, N., Farrel, A., Graf, T., Wu, Q., and C. Yu, "Some Key Terms for Network Fault and Problem Management", Work in Progress, Internet-Draft, draft-ietf-nmop-terminology-23, 18 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-terminology-23>.

[I-D.ietf-nmop-yang-message-broker-integration] Graf, T. and A. Elhassany, "An Architecture for YANG-Push to Message Broker Integration", Work in Progress, Internet-Draft, draft-ietf-nmop-yang-message-broker-integration-10, 18 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-yang-message-broker-integration-10>.

[I-D.wilton-netconf-yang-push-lite]
Wilton, R., Keller, H., Claise, B., Aries, E., Cumming, J., and T. Graf, "YANG Datastore Telemetry (YANG Push Lite)", Work in Progress, Internet-Draft, draft-wilton-netconf-yang-push-lite-02, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-wilton-netconf-yang-push-lite-02>.

[Mar24] Martinez-Casanueva, I. D., Gonzalez-Sanchez, D., Bellido, L., Fernandez, D., and D. R. Lopez, "Toward Building a Semantic Network Inventory for Model-Driven Telemetry", IEEE, DOI 10.1109/MCOM.001.2200222, February 2024, <https://arxiv.org/html/2402.06511v1>.

[Bod24] Bode, J., K端hl, N., Kreuzberger, D., and C. Holtmann, "Toward Avoiding the Data Mess: Industry Insights From Data Mesh Implementations", IEEE, DOI 10.1109/ACCESS.2024.3417291, January 2024, <https://arxiv.org/html/2302.01713v4>.

[Deh22] Dehghani, Z., "Data Mesh", O'Reilly Media, ISBN 9781492092391, March 2022, <https://www.oreilly.com/library/view/data-mesh/9781492092384/>.

[Kim96] Kimball, R. and M. Ross, "The Data Warehouse Toolkit", Wiley, DOI 10.1007/s002360050048, 1996, <https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/books/data-warehouse-dw-toolkit/>.

[One96] O'Neil, P., Cheng, E., Gawlick, D., and E. O'Neil, "The Log-Structured Merge-Tree", Acta Informatica, ISBN 9781118530801, 1996, <https://www.cs.umb.edu/~poneil/lsmtree.pdf>.

[Kaf11] Narkhede, N., "Apache Kafka", Apache Software Foundation, January 2011, <https://kafka.apache.org/>.

[Pul16] Guo, S. and M. Merli, "Apache Pulsar", Apache Software Foundation, January 2016, <https://pulsar.apache.org/>.

[ConDoc18] Yokota, R., "Confluent Schema Registry Documentation", Confluent Community and Apache Software Foundation, December 2018, <https://docs.confluent.io/platform/current/schema-registry/>.

Acknowledgements

Thanks to Camilo Cardona, Rob Wilton, Holger Keller, Reshad Rahman, Nigel Davis, Olga Havel and Michael Mackey for their comments and reviews.

We also like to thank Victor Lopez for the initial idea on the network controller use case. Ashley Woods, Sivakumar Sundaravadivel and Rafael Julio for the idea of grouping topics by YANG-Push subscription type and insisting that Topic Compaction is a key enabler for inventory metrics and YANG data consumer integration and should be supported day 1. Nigel Davis for confirming that Topic Compaction simplifies indeed data processing system architecture and Lo誰c Monney for the operational configuration and monitoring details on Apache Kafka.

Contributors

Many thanks goes to Hellmar Becker who contributed Section 3.1.3 and Section 5 on how YANG Message Keys can be obtained from Message Broker, how time series databases can use it for indexing YANG data and example implementation in ClickHouse.

Hellmar Becker
ClickHouse
601 Marshall Street
Redwood City, CA 94063
United States of America
Email: hellmar.becker@clickhouse.com

Authors' Addresses

Thomas Graf
Swisscom
Binzring 17
CH-8045 Zurich
Switzerland
Email: thomas.graf@swisscom.com

Ahmed Elhassany
Swisscom
Binzring 17
CH-8045 Zurich
Switzerland
Email: ahmed.elhassany@swisscom.com

Alex Huang Feng
INSA-Lyon
Lyon
France
Email: alex.huang-feng@insa-lyon.fr

Beno樽t Claise
Everything OPS
Liege
Belgium
Email: benoit@everything-ops.net

Paolo Lucente
NTT
Veemweg 23
3771 Barneveld
Netherlands
Email: paolo@ntt.net