OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

J. Quilbeuf
Huawei
B. Claise
Everything OPS
T. Graf
Swisscom
D. Lopez
Telefonica I+D
Q. Sun
China Telecom
3 November 2025

External Trace ID for Configuration Tracing
draft-ietf-netconf-configuration-tracing-06

Abstract

Network equipment are often configured by a variety of network management systems (NMS), protocols, and teams. If a network issue arises (e.g., because of a wrong configuration change), it is important to quickly identify the root cause and obtain the reason for pushing that modification. Another potential network issue can stem from concurrent NMSes with overlapping intents, each having their own tasks to perform. In such a case, it is important to map the respective modifications to its originating NMS.

This document specifies a NETCONF mechanism to automatically map the configuration modifications to their source, up to a specific NMS change request. Such a mechanism is required, in particular, for autonomous networks to trace the source of a particular configuration change that led to an anomaly detection. This mechanism facilitates the troubleshooting, the post-mortem analysis, and in the end the closed loop automation required for self-healing networks. The specification also includes a YANG module that is meant to map a local configuration change to the corresponding trace id, up to the controller or even the orchestrator.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/netconf-wg/configuration-tracing.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Issues arising in the network, for instance violation of some SLAs, might be due to some configuration modification. In the context of automated networks, the assurance system needs not only to identify and revert the problematic configuration modification, but also to make sure that it won't happen again and that the fix will not disrupt other services. To cover the last two points, it is imperative to understand the cause of the problematic configuration change. Indeed, the first point, making sure that the configuration modification will not be repeated, cannot be ensured if the cause for pushing the modification in the first place is not known. Ensuring the second point, not disrupting other services, requires as well knowing if the configuration modification was pushed in order to support new services. Therefore, we need to be able to trace a configuration modification on a device back to the reason that triggered that modification, for instance in a NMS, whether the controller or the orchestrator.

This specification focuses only on configuration pushed via NETCONF [RFC6241] or RESTCONF [RFC8040]. The rationale for this choice is that NETCONF is better suited for normalization than other protocols (SNMP, CLI). Another reason is that the notion of trace context, useful to track configuration modifications, has been ported to NETCONF in [I-D.ietf-netconf-trace-ctx-extension] and RESTCONF in [I-D.ietf-netconf-restconf-trace-ctx-headers].

The same network element, or NETCONF [RFC6241] server, can be configured by different NMSs or NETCONF clients. If an issue arises, one of the starting points for investigation is the configuration modification on the devices supporting the impacted service. In the best case, there is a dedicated user for each client and the timestamp of the modification allows tracing the problematic modification to its cause. In the worst case, everything is done by the same user and some more correlations must be done to trace the problematic modification to its source.

This document specifies a mechanism to automatically map the configuration modifications to their source, up to a specific NMS service request. Practically, this mechanism annotates configuration changes on the configured element with sufficient information to unambiguously identify the corresponding transaction, if any, on the element that requested the configuration modification. It reuses the concept of Trace Context [W3C-Trace-Context] applied to NETCONF as in [I-D.ietf-netconf-trace-ctx-extension] The information needed to trace the configuration is stored in a new YANG module that maps a local configuration change to some additional metadata. The additional metadata contains the trace ID, and, if the local change is not the beginning of the trace, the ID of the client that triggered the local-change. In that sense, it is an instance of the YANG DataStore implementation of the Trace Context as proposed in Section 1.2 of [I-D.ietf-netconf-trace-ctx-extension].

2. Terminology

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

This document uses the terms client and server from [RFC6241].

This document uses the terms transaction and Transaction ID from [I-D.ietf-netconf-transaction-id].

This document uses the term trace ID from [W3C-Trace-Context].

Local Commit ID Identifier of a local configuration change on a Network Equipment, Controller, Orchestrator or any other device or software handling configuration. Such an identifier is usually present in devices that can show a history of the configuration changes, to identify one such configuration change.

3. Use cases

This document was written with autonomous networks in mind. We assume that an existing monitoring or assurance system, such as described in [RFC9417], is able to detect and report network anomalies , e.g. SLA violations, intent violations, network failure, or simply a customer issue. Here are the use cases for the proposed YANG module; they are extensions of the "Provisioning root cause analysis" use case presented in Section 1.3.1 of [I-D.ietf-netconf-trace-ctx-extension].

3.1. Configuration Mistakes

Taking into account that many network anomalies are due to configuration mistakes, this mechanism allows to find out whether the offending configuration modification was triggered by a tracing-enabled client/NMS. In such a case, we can map the offending configuration modification id on a server/NE to a local configuration modification id on the client/NMS. Assuming that this mechanism (the YANG module) is implemented on the controller, we can recursively find, in the orchestrator, the latest (set of of) service request(s) that triggered the configuration modification. Whether this/those service request(s) are actually the root cause needs to be investigated. However, they are a good starting point for troubleshooting, post-mortem analysis, and in the end the closed loop automation, which is absolutely required for self-healing networks.

3.2. Concurrent NMS Configuration

Building on the previous use case is the situation where two NMS’s, unaware of each other, are configuring a common router, each believing that they are the only NMS for the common router. So one configuration executed by the NMS1 is overwritten by the NMS2, which in turn is overwritten by NMS1, etc.

3.3. Conflicting Intents

Autonomous networks will be solved first by assuring intent per specific domain; for example data center, core, cloud, etc. This last use case is a more specific "Concurrent NMS configuration" use case where assuring domain intent breaks the entire end to end service, even if the domain-specific controllers are aware of each other.

3.4. Not a use case: Onboarding

During onboarding, a newly added device is likely to receive a multiple configuration message, as it needs to be fully configured. Our use cases focus more on what happens after the initial configuration is done, i.e. when the "stable" configuration is modified.

4. Relying on W3C Trace Context to Trace Configuration Modifications

4.1. Existing configuration metadata on device

This document assumes that NETCONF clients or servers (orchestrators, controllers, devices, ...) have some kind of mechanism to record the modifications done to the configuration. For instance, devices typically have a history of configuration changes and this history associates a locally unique identifier to some metadata, such as the timestamp of the modification, the user doing the modification or the protocol used for the modification. Such a locally unique identifier is a Local Commit ID, we assume that it exists on the platform. This Local Commit ID is the link between the module presented in this draft and the device-specific way of storing configuration changes.

4.2. Client ID

This document assumes that each NETCONF client for which configuration must be traced (for instance orchestrator and controllers) has a unique client ID among the other NETCONF clients in the network. Such an ID could be an IP address or a host name. The mechanism for providing and defining this client ID is out of scope of the current document.

4.3. Instantiating the YANG module

[I-D.ietf-netconf-trace-ctx-extension] defines a NETCONF extension providing the trace context from [W3C-Trace-Context]. Using this mechanism, the NETCONF server captures the trace-id, when available, and maps it to a local commit ID, by populating the YANG module.

The trace context from W3C provides a parent-id. This parent-id does not identify a particular server or NMS but rather one request in the chain of HTTP requests constituting the trace. Similarly to the passing of the trace context in [I-D.ietf-netconf-trace-ctx-extension], we propose an XML attribute on NETCONF messages to pass the client-id. The attribute name is "client-id" and the namespace is the namespace of the YANG module from Section 5, namely ’urn:ietf:params:xml:ns:yang:ietf-external-transaction-id’. An example of a commit message including the client-id is shown in Figure 4.

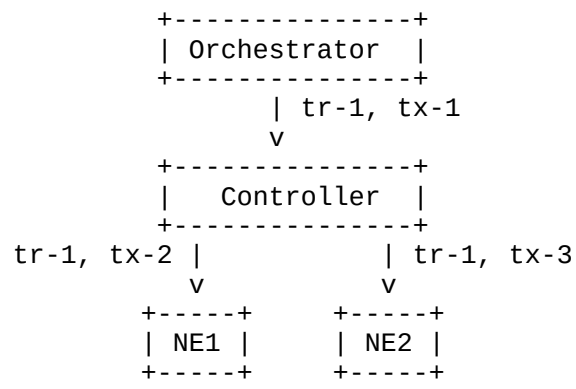

Figure 1: Example of Hierarchical Configuration. tx: transaction. tr: trace.

In Figure 1, the transactions 'tx-1', 'tx-2' and 'tx-3' are sent via NETCONF. The NETCONF RPC used, most likely 'commit' in our use case, is annotated with the 'traceparent' annotation as defined in [I-D.ietf-netconf-trace-ctx-extension]. The traceparent annotation has the same trace id 'tr-1' for each of these transactions. Additionally, for each transaction the client id is passed via the 'client-id' annotation. For 'tx-1' the client-id is the id of the Orchestrator. For 'tx-2' and 'tx-3', the client is the id of the Controller.

It is technically possible that several clients push configuration to the candidate configuration datastore and only one of them commits the changes to the running configuration datastore. From the running configuration datastore perspective, which is the effective one, there is a single modification, but caused by several clients, which means that this modification should have several corresponding client-ids. Although, this case is technically possible, it is a bad practice. We won't cover it in this document. In other terms, we assume that a given configuration modification on a server is caused by a single client, and thus has a single corresponding client-id.

4.4. Using the YANG module

The YANG module defined below enables tracing a configuration change in a Network Equipment back to its origin, for instance a service request in an orchestrator. To do so, the Anomaly Detection System (ADS) should have, for each client-id, access to some credentials enabling read access to the YANG module for configuration tracing on that client. It should as well have access to the network equipment in which an issue is detected.

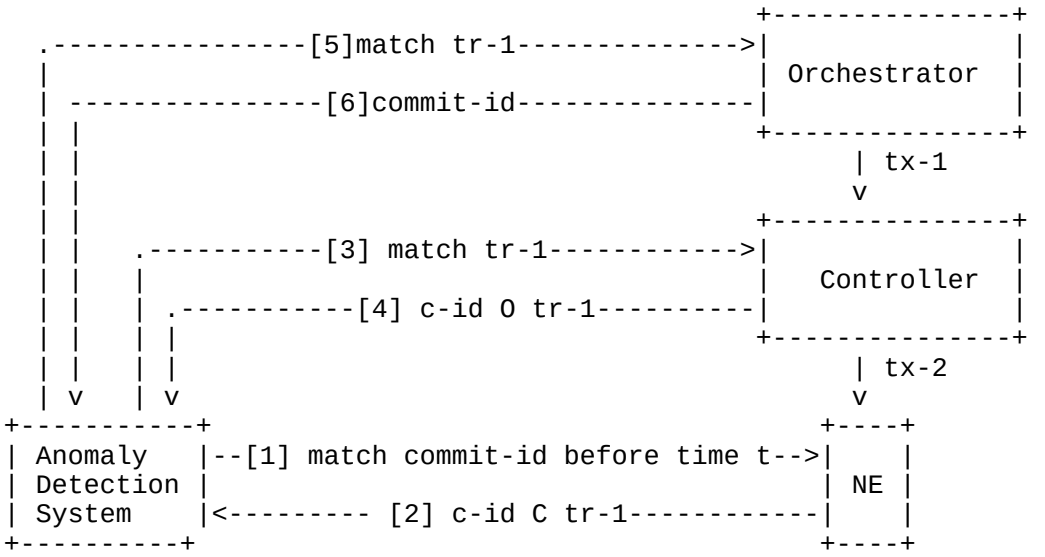

Figure 2: Example of Configuration Tracing. tr: trace-id, C: Controller, O: orchestrator. The number between square brackets refer to steps in the listing below.

The steps for a software to trace a configuration modification in a Network Equipment back to a service request are illustrated in Figure 2. They are detailed below.

1. The Anomaly Detection System (ADS) identifies the commit-id that created an issue, for instance by looking for the last commit-id occurring before the issue was detected. The ADS queries the NE for the trace id and client id associated to the commit-id.

2. The ADS receives the trace-id and the client-id. In Figure 2, that step would receive the trace-id tr-1 and the id of the Controller as a result. If there is no associated client-id, the change was not done by a client compatible with the present draft, and the investigation stops here.

3. The ADS queries the client identified by the client-id found at the previous step, looking for a match of the trace-id from the previous step. In Figure 2, for that step, the software would look for the trace-id tr-1 stored in the Controller.

4. From that query, the ADS knows the local-commit-id on the client (Controller in our case). Since the local-commit-id is associated to a client-id pointing to the Orchestrator, the ADS continues the investigation.

5. The ADS queries the Orchestrator, trying to find a match for the trace-id tr-1.

6. Finally, the ADS receives the commit-id from the Orchestrator that ultimately caused the issue in the NE. Since there is no associated client-id, the investigation stops here. The modification associated to the commit-id, for instance a service request, is now available for further manual or automated analysis, such as analyzing the root cause of the issue.

Note that step 5 and 6 are actually a repetition of step 3 and 4. The general algorithm is to continue looking for a client until no more client-id can be found in the current element.

5. YANG module

We present in this section the YANG module for modelling the information about the configuration modifications.

5.1. Overview

The tree representation [RFC8340] of our YANG module is depicted in Figure 3

```
module: ietf-external-transaction-id
  +--ro external-transactions-id
     +--ro configuration-change* [local-commit-id]
        +--ro local-commit-id    string
        +--ro timestamp?         yang:date-and-time
        +--ro trace-parent
        |  +--ro version?       hex-digits
        |  +--ro trace-id?      hex-digits
        |  +--ro parent-id?     hex-digits
        |  +--ro trace-flags?   hex-digits
        +--ro client-id?         string
```

Figure 3: Tree representation of ietf-external-transaction-id YANG module

The local-commit-id represents the local id of the configuration changes, which is device-specific. It can be used to retrieve the local configuration changes that happened during that transaction.

The trace-parent is present to identify the trace associated to the local-commit-id. This trace-parent can be transmitted by a client or created by the current server. In Section 4.4, the most important field in trace-parent is the trace-id. We also included the other fields for trace-parent as defined in [W3C-Trace-Context] for the sake of completion. In some cases, for instance direct configuration of the device, the device may choose to not include the trace-id.

The presence of a client-id indicates that the trace-parent has been transmitted by that client. If the trace is initiated by the current server, there is no associated client-id.

Even if this document focuses only on NETCONF or RESTCONF, the use cases defined in Section 3 are not specific to NETCONF or RESTCONF and the mechanism described in this document could be adapted to other configuration mechanisms. For instance, a configuration modification pushed via CLI can be identified via a label, which could contain the trace-parent. As such cases are difficult to standardize, we won't cover them in this document.

5.2. YANG module ietf-external-transaction-id

```
<CODE BEGINS> file "ietf-external-transaction-id@2021-11-03.yang"
module ietf-external-transaction-id {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-external-transaction-id";
  prefix ext-txid;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types, Section 3";
  }

  organization
    "IETF NETCONF Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf  <mailto:jean.quilbeuf@huawei.com>";
  description
    "This module enables tracing of configuration changes in a
     network for the sake of automated correlation between
     configuration changes and the external request that triggered
     that change.
```

```
        The module stores the identifier of the trace, if any, that
        triggered the change in a device. If that trace-id was provided
        by a client, (i.e. not created locally by the server), the id
        of that client is stored as well to indicated which client
        triggered the configuration change.

        Copyright (c) 2025 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Revised BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (https://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see the
        RFC itself for full legal notices.  ";

    revision 2022-10-20 {
      description
        "Initial revision";
      reference
        "RFC xxxx: External Trace ID for Configuration Tracing";
    }

    typedef hex-digits {
      type string {
        pattern '[0-9a-f]*';
      }
      description
        "A string composed of hexadecimal digits. Digits represented by
         letters are restricted to lowercase so that a single
         representation of a given value is allowed. This enables using
         the string equality to check equality of the represented
         values.";
    }

    grouping trace-parent-g {
      description
        "Trace parent frow the W3C trace-context recommandation.
         Follows the format version 00.";
      leaf version {
        type hex-digits {
          length "2";
        }
        must "../version = '00'";
        description
          "Version of the trace context. Must be 00 to match the
```

```
            format described in this module.";
        }
        leaf trace-id {
          type hex-digits {
            length "32";
          }
          must "../trace-id != '00000000000000000000000000000000'";
          description
            "Trace ID that is common for every transaction that is
             part of the configuration chain. This value can be used
             to match a local commit id to a commit local to another
             system.";
        }
        leaf parent-id {
          type hex-digits {
            length "16";
          }
          description
            "ID of the request (client-side) that lead to configuring
             the server hosting this module.";
        }
        leaf trace-flags {
          type hex-digits {
            length "2";
          }
          description
            "Flags enabled for this trace. See W3C reference for the
             details about flags.";
        }
      }

      container external-transactions-id {
        config false;
        description
          "Contains the IDs of configuration transactions that are
           external to the device.";
        list configuration-change {
          key "local-commit-id";
          description
            "List of configuration changes, identified by their
             local-commit-id";
          leaf local-commit-id {
            type string;
            description
              "Stores the identifier as saved by the server. Can be used
               to retrieve the corresponding changes using the server
               mechanism if available.";
          }
```

```
        leaf timestamp {
          type yang:date-and-time;
          description
            "A timestamp that can be used to further filter change
             events.";
        }
        container trace-parent {
          description
            "Trace parent associated to the local-commit-id. If a
             client ID is present as well, the trace context was
             transmitted by that client. If not, the trace context was
             created locally.

             This trace-parent must come from the trace context of the
             request actually modifying the running configuration
             datastore. This request might be an edit-config or a
             commit depending on whether the candidate datastore is
             used.";
          uses trace-parent-g;
        }
        leaf client-id {
          type string;
          description
            "ID of the client that originated the modification, to
             further query information about the corresponding
             change.

             This data node is present only when the configuration was
             pushed by a compatible system.";
        }
      }
    }
  }
  <CODE ENDS>
```

6. Security Considerations

This section is modeled after the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The "ietf-external-transaction-id" module defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF[RFC8040]. These protocols have to use a secure transport layer (e.g., SSH [RFC6242], TLS [RFC8446] and QUIC [RFC9000]) and have to use mutual authentication.

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

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. Specifically, the following subtrees and data nodes have particular sensitivities/ vulnerabilities:

* external-transactions-id/configuration-change exposes information about which user or external system can configure the device and could help an attacker to send its own configuration to the device. It could also give some information about the architecture of the configuration, i.e. what are the controllers and the orchestrators.

7. IANA Considerations

RFC Ed.: replace XXXX with actual RFC number and remove this note.

IANA is requested to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-external-transaction-id
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

IANA is requested to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

```
Name: ietf-external-transaction-id
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-external-transaction-id
Prefix: ext-txid
Reference: RFC XXXX
```

8. Contributors

9. Open Issues / TODO

This section is to be removed before publishing as an RFC.

None

10. Normative References

[I-D.ietf-netconf-restconf-trace-ctx-headers]
Gagliano, R., Larsson, K., and J. Lindblad, "RESTCONF Extension to Support Trace Context Headers", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-trace-ctx-headers-07, 19 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-trace-ctx-headers-07>.

[I-D.ietf-netconf-trace-ctx-extension]
Gagliano, R., Larsson, K., and J. Lindblad, "NETCONF Extension to support Trace Context propagation", Work in Progress, Internet-Draft, draft-ietf-netconf-trace-ctx-extension-05, 19 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trace-ctx-extension-05>.

[I-D.ietf-netconf-transaction-id]
Lindblad, J., "Transaction ID Mechanism for NETCONF", Work in Progress, Internet-Draft, draft-ietf-netconf-transaction-id-07, 19 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-transaction-id-07>.

[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>.

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

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

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

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

[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>.

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

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

[W3C-Trace-Context] "W3C Recommendation on Trace Context", 23 November 2021, <https://www.w3.org/TR/2021/REC-trace-context-1-20211123/>.

11. Informative References

[I-D.ietf-netmod-rfc8407bis] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

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

Appendix A. Changes between revisions

This section is to be removed before publishing as an RFC.

05 -> 06

* Change Benoit’s affiliation

04 -> 05

* Fix security considerations template

03 -> 04

* Add security and IANA considerations

01 -> 02

* Remove YANG specific annotation for the mechanism to pass the client-id.

* Align with NETCONF Trace context draft.

00(WG adoption) -> 01

* Define mechanism to pass the client-id.

01 -> 02

* Switch to trace-parent instead of transaction id for tracing configuration

00 -> 01

* Define Parent and Child Transaction

* Context for the "local-commit-id" concept

* Feedback from Med, both in text and YANG module

Appendix B. Example of NETCONF message

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:w3ctc="urn:ietf:params:xml:ns:netconf:w3ctc:1.0"
     xmlns:ext-txid=
       "urn:ietf:params:xml:ns:yang:ietf-external-transaction-id"
     w3ctc:traceparent=
       "00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01"
     ext-txid:client-id="controller-01">
  <commit/>
</rpc>
```

Figure 4: Example of NETCONF commit RPC with annotations

In Figure 4, we present an RPC annotated with the traceparent and the client-id. The traceparent example is taken from [I-D.ietf-netconf-trace-ctx-extension]. The client-id annotation is defined in our YANG module. Here the client-id passed is 'controller-01'.

Acknowledgements

The authors would like to thank Mohamed Boucadair, Jan Linblad and Roque Gagliano for their reviews and propositions.

Authors' Addresses

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

Benoit Claise
Everything OPS
Email: benoit@everything-ops.net

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

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain
Email: diego.r.lopez@telefonica.com

Qiong Sun
China Telecom
Email: sunqiong@chinatelecom.cn