Workgroup: NETCONF
Internet-Draft:
draft-rogaglia-netconf-trace-ctx-extension-01
Published: 13 January 2023
Intended Status: Standards Track
Expires: 17 July 2023
Authors: R. Gagliano
Cisco Systems
K. Larsson
Deutsche Telekom AG
J. Lindblad
Cisco Systems

NETCONF Extension to support Trace Context propagation

Abstract

This document defines how to propagate trace context information across the Network Configuration Protocol (NETCONF), that enables distributed tracing scenarios. It is an adaption of the HTTP-based W3C specification.

About This Document

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

The latest revision of this draft can be found at TBD. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-rogaglia-netconf-trace-ctx-extension/.

Discussion of this document takes place on the NETCONF Working Group mailing list (mailto:netconf@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/netmod/. Subscribe at https://www.ietf.org/mailman/listinfo/netconf/.

Source for this draft and an issue tracker can be found at https://github.com/TBD.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 17 July 2023.

Table of Contents

1. Introduction

Network automation and management systems commonly consist of multiple sub-systems and together with the network devices they manage, they effectively form a distributed system. Distributed tracing is a methodology implemented by tracing tools to follow, analyze and debug operations, such as configuration transactions, across multiple distributed systems. An operation is uniquely

identified by a trace-id and through a trace context, carries some metadata about the operation. Propagating this "trace context" between systems enables forming a coherent view of the entire operation as carried out by all involved systems.

The W3C has defined two HTTP headers for context propagation that are useful in use case scenarios of distributed systems like the ones defined in [RFC8309]. This document defines an extension to the NETCONF protocol to add the same concepts and enable trace context propagation over NETCONF.

It is worth noting that the trace context is not meant to have any relationship with the data that is carried with a given operation (including configurations, service identifiers or state information).

A trace context also differs from [I-D.lindblad-netconf-transaction-id] in several ways as the trace operation may involve any operation (including for example validate, lock, unlock, etc.) Additionally, a trace context scope may include the full application stack (orchestrator, controller, devices, etc) rather than a single NETCONF server, which is the scope for the transaction-id. The trace context is also complemetary to [I-D.lindblad-netconf-transaction-id] as a given trace-id can can associated to the different transaction-ids as part of the information exported to the collector.

The following enhancement of the reference SDN Architecture from RFC 8309 shows the impact of distributed traces for a network operator.

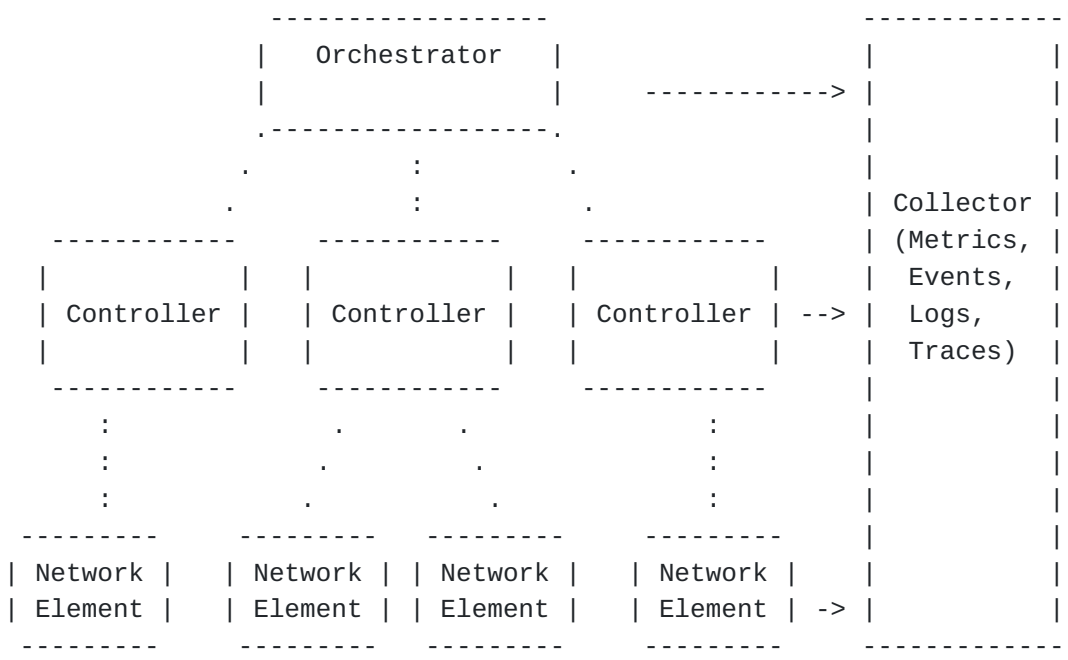

Figure 1: A Sample SDN Architecture from RFC8309 augmented to include the export of metrics, events, logs and traces from the different components to a common collector.

The network automation, management and control architectures are distributed in nature. In order to "manage the managers", operators would like to use the same techniques as any other distributed systems in their IT environment. Solutions for analysing Metrics, Events, Logs and Traces (M.E.L.T) are key for the successful monitoring and troubleshooting of such applications. Initiatives such as the OpenTelemetry [OpenTelemetry] enable rich ecosystems of tools that NETCONF-based applications would want to participate in.

With the implementation of this trace context propagation extension to NETCONF, backend systems behind the M.E.L.T collector will be able to correlate information from different systems but related to a common context.

1.1. Implementation example 1: OpenTelemetry

We will describe an example to show the value of trace context propagation in the NETCONF protocol. In Figure 2, we show a deployment based on Figure 1 with a single controller and two network elements. In this example, the NETCONF protocol is running between the Orchestrator and the Controller. NETCONF is also used between the Controller and the Network Elements.

Let's assume an edit-config operation between the orchestrator and the controller that results (either synchronously or asynchronously) in corresponding edit-config operations from the Controller towards

the two network elements. All trace operations are related and will create M.E.L.T data.

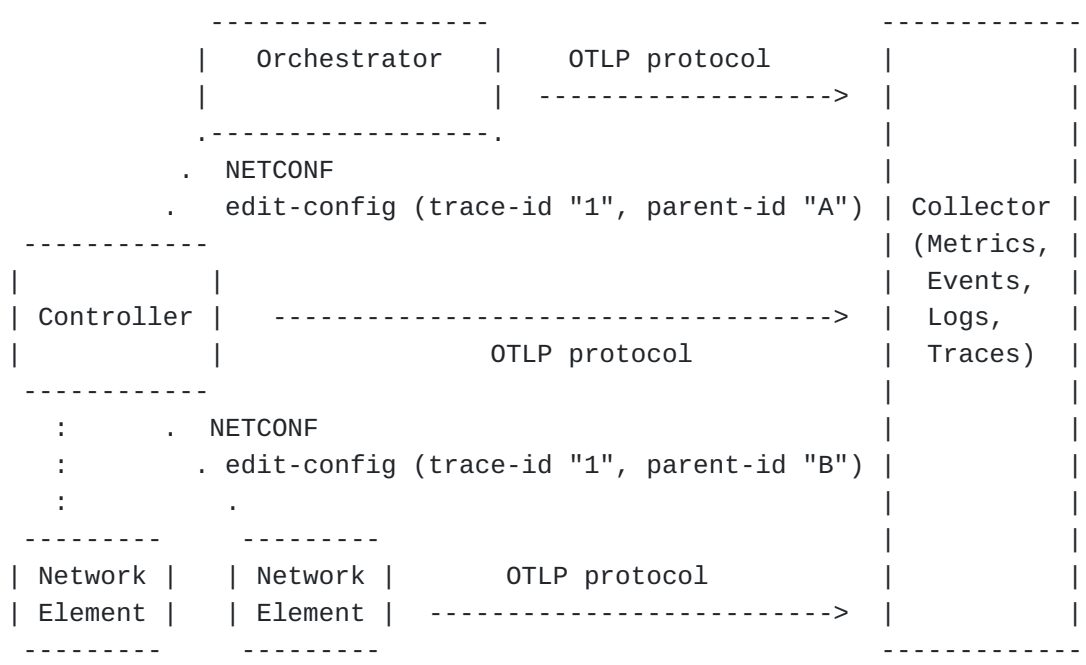

Figure 2: An implementation example where the NETCONF protocol is used between the Orchestrator and the Controller and also between the Controller and the Network Elements. Every component exports M.E.L.T information to the collector using the OTLP protocol.

Each of the components in this example (Orchestrator, Controller and Network Elements) is exporting M.E.L.T information to the collector using the OpenTelemetry Protocol (OTLP).

For every edit-config operation, the trace context is included. In particular, the same trace-id "1" (simplified encoding for documentation) is included in all related NETCONF messages, which enables the collector and any backend application to correlate all M.E.L.T messages related to this transaction in this distributed stack.

Another interesting attribute is the parent-id. We can see in this example that the parent-id between the orchestrator and the controller ("A") is different from the one between the controller and the network elements ("B"). This attribute will help the collector and the backend applications to build a connectivity graph to understand how M.E.L.T information exported from one component relates to the information exported from a different component.

With this additional metadata exchanged between the components and exposed to the M.E.L.T collector, there are important improvements

to the monitor and troubleshooting operations for the full application stack.

1.2. Implementation example 2: YANG DataStore

OpenTelemetry implements the "push" model for data streaming where information is sent to the back-end as soon as produced and is not required to be stored in the system. In certain cases, a "pull" model may be envisioned, for example for performing forensic analysis while not all OTLP traces are available in the back-end systems.

An implementation of a "pull" mechanism for M.E.L.T. information in general and for traces in particular, could consist of storying traces in a yang datastore (particularly the operational datastore.) Implementations should consider the use of circular buffers to avoid resources exhaustion. External systems could access traces (and particularly past traces) via NETCONF, RESTCONF, gNMI or other polling mechanisms. Finally, storying traces in a YANG datastore enables the use of YANG-Push [RFC8641] or gNMI Telemetry as an additional "push" mechanisms.

This document does not specify the YANG module in which traces could be stored inside the different components. That said, storing the context information described in this document as part of the recorded traces would allow back-end systems to correlate the information from different components as in the OpenTelemetry implementation.

Note to be removed in the future: Some initial ideas are under discussion in the IETF for defining a standard YANG data model for traces. For example see: I-D.quilbeuf-opsawg-configuration-tracing which focusses only on configuration change root cause analysis use case (see bellow the use case desciption.). This ideas are complementary to this draft.

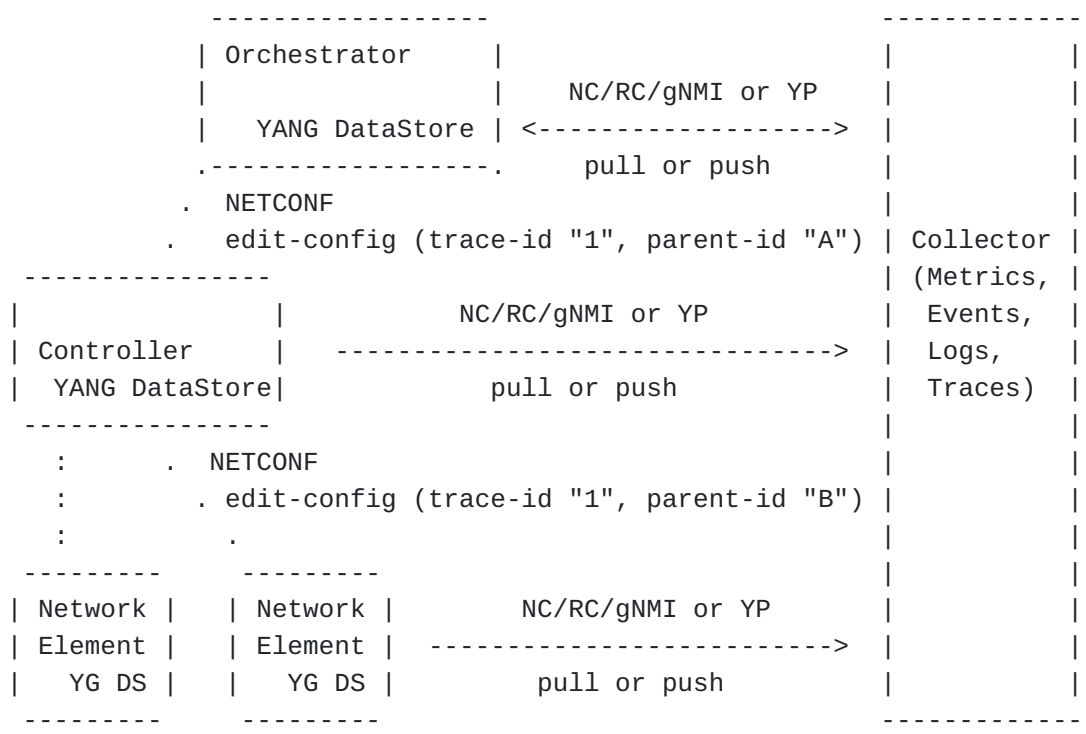

Figure 3: An implementation example where the NETCONF protocol is used between the Orchestrator and the Controller and also between the Controller and the Network Elements. M.E.L.T. information is stored in local Yang Datastores and accessed by the collector using "pull" mechanisms using the NETCONF (NC), RESTCONF (RC) or gNMI protocols. A "push" strategy is also possible via YANG-Push or gNMI.

1.3. Use Cases

1.3.1. Provisioning root cause analysis

When a provisioning activity fails, errors are typically propagated northbound, however this information may be difficult to troubleshoot and typically, operators are required to navigate logs across all the different components.

With the support for trace context propagation as described in this document for NETCONF, the collector will be able to search every trace, event, metric, or log in connection to that trace-id and faciliate the performance of a root cause analysis due to a network changes. The trace information could also include as an optional resource the different NETCONF transaction ids described in [I-D.lindblad-netconf-transaction-id].

1.3.2. System performaednce profiling

When operating a distributed system such as the one shown in Figure 2, operators are expected to benchmark what are the Key Performance Indicators (KPIs) for the most common tasks. For example, what is the typical delay when provisioning a VPN service across different controllers and devices.

Thanks to Application Performance Management (APM) systems, from these KPIs, an operator can detect a normal and abnormal behaviour of the distributed system. Also, an operator can better plan any upgrades or enhancements in the platform.

With the support for context propagation as described in this document for NETCONF, much richer system-wide KPIs can be defined and used for troubleshooting as the metrics and traces propagated by the different components share a common context. Troubleshooting for abnormal behaviours can also be troubleshot from the system view down to the individual element.

1.3.3. Billing and auditing

In certain circuntances, we could perceive that tracing infomration could be used as additional inputs to billing systems. In particular, trace context information could be used to validate that a certain northbound order was carried out in southbound systems.

1.4. Terminology

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

The XML prefixes used in this document are mapped as follows:

*xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0",

*xmlns:notif="urn:ietf:params:xml:ns:netconf:notification:1.0",

*xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-patch" and

*xmlns:ypatch="urn:ietf:params:xml:ns:yang:ietf-yang-patch".

2. NETCONF Extension

When performing NETCONF operations by sending NETCONF RPCs, a NETCONF client MAY include trace context information in the form of XML attributes. The [W3C-Trace-Context] defines two HTTP headers;

traceparent and tracestate for this purpose. NETCONF clients that are taking advantage of this feature MUST add one w3ctc:traceparent attribute to the nc:rpc tag.

A NETCONF server that receives a trace context attribute in the form of a w3ctc:traceparent attribute SHOULD apply the mutation rules described in [W3C-Trace-Context]. A NETCONF server MAY add one w3ctc:traceparent attribute in the nc:rpc-reply response to the nc:rpc tag above. NETCONF servers MAY also add one w3ctc:traceparent attribute in notification and update message envelopes: notif:notification, yp:push-update and yp:push-change-update.

For example, a NETCONF client might send:

```
<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"
     w3ctc:traceparent=
       "00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01">
  <get-config/>
</rpc>
```

In all cases above where a client or server adds a w3ctc:traceparent attribute to a tag, that client or server MAY also add one w3ctc:tracestate attribute to the same tag.

The proper encoding and interpretation of the contents of the w3ctc:traceparent attribute is described in [W3C-Trace-Context] section 3.2 except 3.2.1. The proper encoding and interpretation of the contents in the w3ctc:tracestate attribute is described in [W3C-Trace-Context] section 3.3 except 3.3.1 and 3.3.1.1. A NETCONF tag can only have zero or one w3ctc:tracestate attributes, so its content MUST always be encoded as a single string. The tracestate field value is a list of list-members separated by commas (,). A list-member is a key/value pair separated by an equals sign (=). Spaces and horizontal tabs surrounding list-members are ignored. There is no limit to the number of list-members in a list.

For example, a NETCONF client might send:

```
<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"
     w3ctc:tracestate="rojo=00f067aa0ba902b7,congo=t61rcWkgMzE"
     w3ctc:traceparent=
       "00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01">
  <get-config/>
</rpc>
```

As in all XML documents, the order between the attributes in an XML tag has no significance. Clients and servers MUST be prepared to handle the attributes no matter in which order they appear. The

tracestate value MAY contain double quotes in its payload. If so, they MUST be encoded according to XML rules, for example:

```
<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"
     w3ctc:traceparent=
       "00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01"
     w3ctc:tracestate=
       "value-with-quotes=&quot;Quoted string&quot;,other-value=123">
  <get-config/>
</rpc>
```

TBD Errors ....

3. Security Considerations

TODO Security

4. IANA Considerations

This document registers the following capability identifier URN in the 'Network Configuration Protocol (NETCONF) Capability URNs' registry:

urn:ietf:params:netconf:capability:w3ctc:1.0

This document registers one XML namespace URN in the 'IETF XML registry', following the format defined in [RFC3688] (https://tools.ietf.org/html/rfc3688).

URI: urn:ietf:params:xml:ns:netconf:w3ctc:1.0

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

5. Acknowledgments

TBD

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[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>.

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

6.2. Informative References

[I-D.lindblad-netconf-transaction-id] Lindblad, J., "Transaction ID Mechanism for NETCONF", Work in Progress, Internet-Draft, draft-lindblad-netconf-transaction-id-02, 8 June 2022, <https://www.ietf.org/archive/id/draft-lindblad-netconf-transaction-id-02.txt>.

[OpenTelemetry] "OpenTelemetry Cloud Native Computing Foundation project", 29 August 2022, <https://opentelemetry.io>.

[RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <https://www.rfc-editor.org/info/rfc8309>.

[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>.

[W3C-Baggage] "W3C Propagation format for distributed context Baggage", 23 November 2021, <https://www.w3.org/TR/baggage/#examples-of-http-headers>.

Appendix A. Changes

A.1. From version 00 to 01

*Added new section: Implementation example 2: YANG DataStore

*Added new use case: Billing and auditing

*Added in introduction and in "Provisioning root cause analysis" the idea that the different transaction-ids defined in [I-D.lindblad-netconf-transaction-id] could be added as part of

the tracing information to be exported to the collectors, showing how the two documents are complementary.

Appendix B. TO DO List (to be deleted by RFC Editor)

- *Manage versioning of the trace-context specification
- *Error handling specification
- *We intend to extend the trace-concext capability to RESTCONF in a future draft
- *The W3C is working on a draft document to introduce the concept of "baggage" [W3C-Baggage] that we expect part of a future draft for NETCONF and RESTCONF

Appendix C. XML Attributes vs RPCs input augmentations discussion (to be deleted by RFC Editor)

There are arguments that can be raised regarding using XML Attribute or to augment NETCONF RPCs.

We studied Pros/Cons of each option and decided to propose XML attributes:

XML Attributes Pro:

- *Literal alignment with W3C specification
- *Same encoding for RESTCONF and NETCONF enabling code reuse
- *One specification for all current and future rpcs

XML Attributes Cons:

- *No YANG modeling, multiple values represented as a single string
- *Dependency on W3C for any extension or changes in the future as encoding will be dictated by string encoding

RPCs Input Augmentations Pro:

- *YANG model of every leaf
- *Re-use of YANG toolkits
- *Simple updates by augmentations on existing YANG module
- *Possibility to express deviations in case of partial support

RPCs Input Augmentations Cons:

*Need to augment every rpc, including future rpcs would need to consider these augmentations, which is harder to maintain

*There is no literal alignment with W3C standard. However, as mentioned before most of the time there will be modifications to the content

*Would need updated RFP for each change at W3C, which will make adoption of new features slower

Authors' Addresses

Roque Gagliano
Cisco Systems
Avenue des Uttins 5
CH-1180 Rolle
Switzerland

Email: rogaglia@cisco.com

Kristian Larsson
Deutsche Telekom AG

Email: kll@dev.terastrm.net

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com