Network Configuration
Internet-Draft
Intended status: Standards Track
Expires: 4 September 2025

R. Gagliano
Cisco Systems
K. Larsson
Deutsche Telekom AG
J. Lindblad
Cisco Systems

3 March 2025

NETCONF Extension to support Trace Context propagation
draft-ietf-netconf-trace-ctx-extension-04

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 https://datatracker.ietf.org/doc/draft-ietf-netconf-trace-ctx-extension/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-netconf-trace-ctx-extension/.

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

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

Status of This Memo

Copyright Notice

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.ietf-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 complementary to [I-D.ietf-netconf-transaction-id] as a given trace-id can be associated with the different transaction-ids as part of the information exported to the collector.

The following enhancement of the reference SDN Architecture from [RFC8309] shows the impact of distributed traces for a network operator.

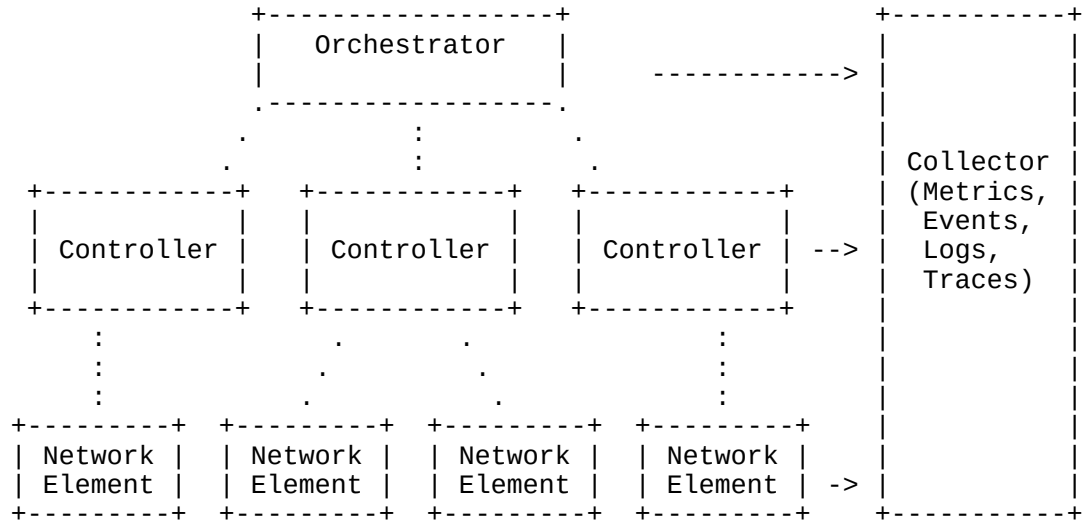

Figure 1: A Sample SDN Architecture from RFC8309 augmented to include the export of metrics, events, logs and traces from the different components to a common collector.

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.

This document does not cover the somewhat related functionality specified in [W3C-Baggage]. Mapping of the Baggage functionality into YANG may be specified in a future document.

1.1. Terminology

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

Additionally, the document utilizes the following abbreviations:

OTLP: OpenTelemetry protocol as defined by [OpenTelemetry]

M.E.L.T.: Metrics, Events, Logs and Traces

gNMI: gRPC Network Management Interface, as defined by [gNMI]

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

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

* xmlns:w3ctc="urn:ietf:params:xml:ns:netconf:w3ctc:1.0" and

* xmlns:ietf-trace-context= "urn:ietf:params:xml:ns:yang:ietf-trace-context"

1.2. Implementation example 1: OpenTelemetry

We will describe an example to show the value of trace context propagation in the NETCONF protocol. In the OTLP Sample Architecture Figure 2 below, we show a deployment based on the RFC8309 sample architecture Figure 1 above, 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.

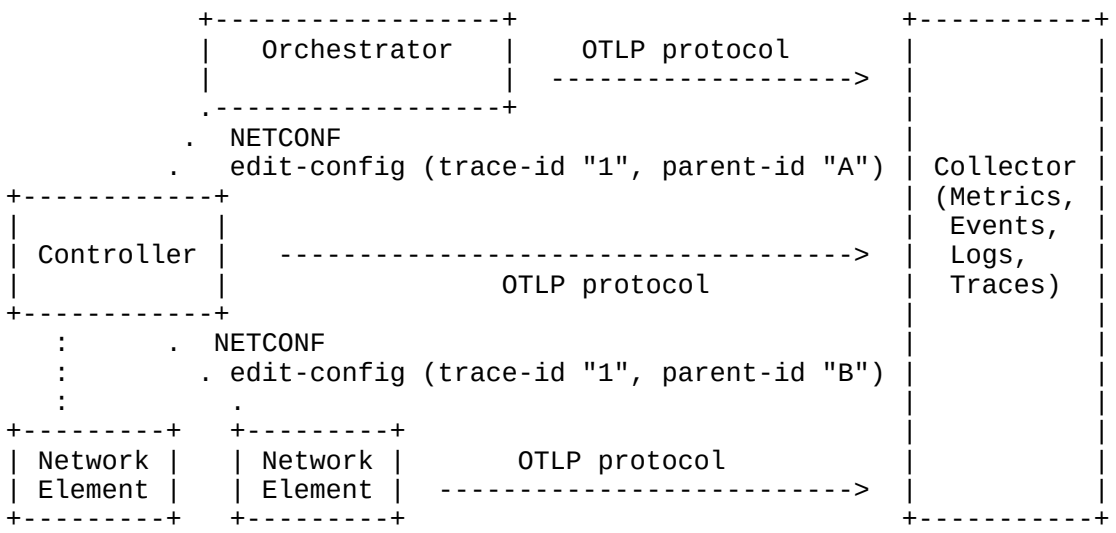

Figure 2: An implementation example where the NETCONF protocol is used between the Orchestrator and the Controller and also between the Controller and the Network Elements. Every component exports M.E.L.T information to the collector using the OTLP protocol.

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 monitoring and troubleshooting operations for the full application stack.

1.3. 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 storing traces in a YANG datastore (particularly the operational datastore.) Implementations should consider the use of circular buffers to avoid resource exhaustion. External systems could access traces (and particularly past traces) via NETCONF, RESTCONF, gNMI or other polling mechanisms. Finally, storing traces in a YANG datastore enables the use of YANG-Push [RFC8641] or gNMI Telemetry as 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.

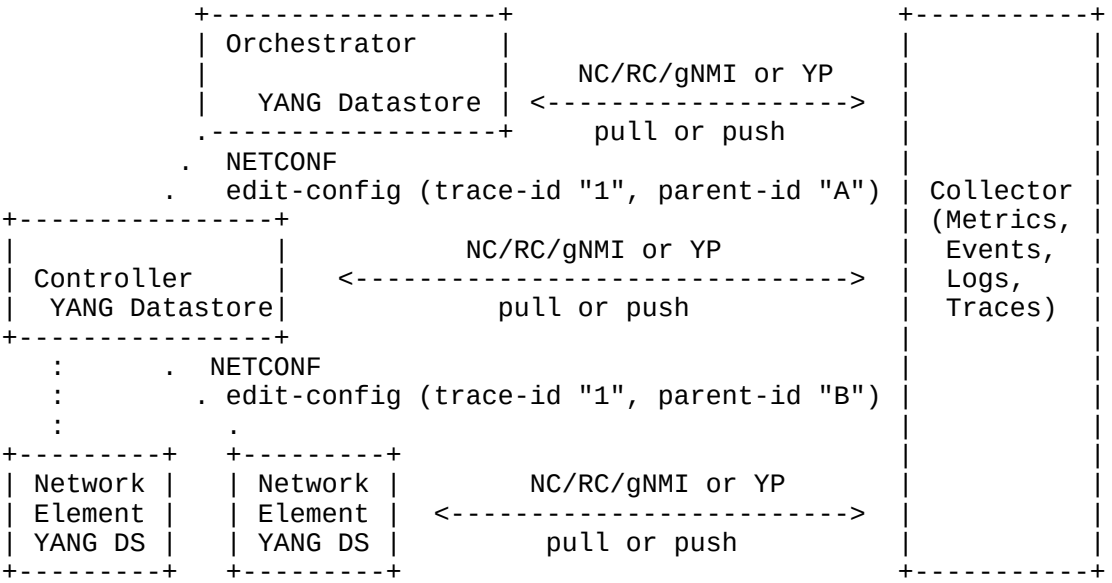

Figure 3: An implementation example where the NETCONF protocol is used between the Orchestrator and the Controller and also between the Controller and the Network Elements. M.E.L.T. information is stored in local YANG Datastores and accessed by the collector using "pull" mechanisms using the NETCONF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

1.4. Use Cases

1.4.1. 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 facilitate the performance of a root cause analysis due to a network changes. The trace information could also be included as an optional resource with the different NETCONF transaction ids described in [I-D.ietf-netconf-transaction-id].

1.4.2. System performance profiling

When operating a distributed system such as the one shown in Figure 2, operators are expected to benchmark 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.

1.4.3. Billing and auditing

In certain circumstances, we could envision tracing information 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.

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 and MAY add one w3ctc:tracestate 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 XML 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>
```

2.1. Error handling

The NETCONF server SHOULD follow the "Processing Model for Working with Trace Context" as specified in [W3C-Trace-Context]. Based on this processing model, it is NOT RECOMMENDED to reject an RPC because of the trace context attribute values.

If the server still decides to reject the RPC because of the trace context attribute values, the server MUST return a NETCONF rpc-error with the following values:

```
  error-tag:      operation-failed
  error-type:     protocol
  error-severity: error
```

Additionally, the error-info tag MUST contain a trace-context-error-info structure with relevant details about the error. This structure is defined in the module ietf-netconf-context.yang. Example of a badly formated trace context extension:

```
<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=
       "Bad Format"
     w3ctc:tracestate=
       "value-with-quotes=&quot;Quoted string&quot;,other-value=123">
  <get-config/>
</rpc>
```

This might give the following error response:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
            xmlns:w3ctc="urn:ietf:params:xml:ns:netconf:w3ctc:1.0"
            xmlns:ietf-trace-context=
            "urn:ietf:params:xml:ns:yang:ietf-trace-context"
            message-id="1">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message>
      Context traceparent attribute incorrectly formatted
    </error-message>
    <error-info>
      <ietf-trace-context:trace-context-error-info>
        <ietf-trace-context:meta-name>
          w3ctc:traceparent
        </ietf-trace-context:meta-name>
        <ietf-trace-context:meta-value>
          Bad Format
        </ietf-trace-context:meta-value>
        <ietf-trace-context:error-type>
          ietf-trace-context:bad-format
        </ietf-trace-context:error-type>
      </ietf-trace-context:trace-context-error-info>
    </error-info>
  </rpc-error>
</rpc-reply>
```

2.2. Trace Context extension versioning

This extension refers to the [W3C-Trace-Context] trace context capability. The W3C traceparent and tracestate headers include the notion of versions. It would be desirable for a NETCONF client to be able to discover the one or multiple versions of these headers supported by a server. We would like to achieve this goal avoiding the definition of new NETCONF capabilities for each headers' version.

We define a pair YANG modules (ietf-trace-ctx-traceparent-1.0.yang and ietf-trace-ctx-tracestate-1.0.yang) that MUST be included in the YANG library per [RFC8525] of the NETCONF server supporting the NETCONF Trace Context extension. These capabilities that will refer to the headers' supported versions. Future updates of this document could include additional YANG modules for new headers' versions.

3. YANG Modules

3.1. YANG module for trace-context-error-info structure

```
<CODE BEGINS> file "ietf-trace-context@2024-11-07.yang"
module ietf-trace-context {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-trace-context";
  prefix ietf-trace-context;

  import ietf-yang-structure-ext {
    prefix sx;
    reference
      "RFC8791: YANG Data Structure Extensions";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Authors:  Roque Gagliano
               <mailto:rogaglia@cisco.com>

               Jan Lindblad
               <mailto:jlindbla@cisco.com>

               Kristian Larsson
               <mailto:kll@dev.terastrm.net>
    ";
  description
```

```
       "When propagating tracing information across applications,
        client and servers needs to share some specific contextual
        information. This  extensions aligns the NETCONF and RESTCONF
        protocols to the W3C trace-context document:
        https://www.w3.org/TR/2021/REC-trace-context-1-20211123

        Copyright (c) 2024 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
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
        for full legal notices

        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 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.
       ";

     revision 2024-11-07 {
       description
         "Initial revision";
       reference
         "RFC XXXX:
          NETCONF Extension to support Trace Context propagation";
     }

     identity meta-error {
       description
         "Base identity for trace context attribute errors.";
     }

     identity missing {
       base meta-error;
       description
         "Indicates an attribute or header that is required
          (in the current situation) is missing.";
     }

     identity bad-format {
```

```
    base meta-error;
    description
      "Indicates an attribute or header value where the
       value is incorrectly formatted.";
  }

  identity processing-error {
    base meta-error;
    description
      "Indicates that the server encountered a processing
       error while processing the attribute or header value.";
  }

  typedef meta-error-type {
    type identityref {
      base meta-error;
    }
    description
      "Error type";
  }

  sx:structure trace-context-error-info {
    container trace-context-error-info {
      description
        "This error is returned by a NETCONF or RESTCONF server
         when a client sends a NETCONF RPC with additonal
         attributes or RESTCONF RPC with additional headers
         for trace context processing, and there is an error
         related to them.  The server has aborted the RPC.";
      leaf meta-name {
        type string;
        description
          "The name of the problematic or missing meta information.
           In NETCONF, the qualified XML attribute name.
           In RESTCONF, the HTTP header name.
           If a client sent a NETCONF RPC with the attribute
           w3ctc:traceparent='incorrect-format'
           this leaf would have the value 'w3ctc:traceparent'";
      }
      leaf meta-value {
        type string;
        description
          "The value of the problematic meta information received
           by the server.
           If a client sent a NETCONF RPC with the attribute
           w3ctc:traceparent='incorrect-format'
           this leaf would have the value 'incorrect-format'.";
      }
```

```
        leaf error-type {
          type meta-error-type;
          description
            "Indicates the type of meta information problem
             detected by the server.
             If a client sent an RPC annotated with the attribute
             w3ctc:traceparent='incorrect-format'
             this leaf might have the value
             'ietf-trace-context:bad-format'.";
        }
      }
    }
  }
  <CODE ENDS>
```

3.2. YANG module for traceparent header version 1.0

```
  <CODE BEGINS> file "ietf-trace-ctx-traceparent-1.0@2024-11-07.yang"
  module ietf-trace-ctx-traceparent-1.0 {
    namespace
    "urn:ietf:params:xml:ns:yang:ietf-trace-ctx-traceparent-1.0";
    prefix ietf-trace-ctx-traceparent-1.0;

    organization
      "IETF NETCONF (Network Configuration) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
       WG List:  <mailto:netconf@ietf.org>

       Authors:  Roque Gagliano
                 <mailto:rogaglia@cisco.com>

                 Jan Lindblad
                 <mailto:jlindbla@cisco.com>

                 Kristian Larsson
                 <mailto:kll@dev.terastrm.net>
      ";
    description
      "This module documents the support for trace context traceparent
       versiom 1.0 in alignment with W3C versions:
       https://www.w3.org/TR/2021/REC-trace-context-1-20211123

       Copyright (c) 2024 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
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
        for full legal notices

        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 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.
      ";

    revision 2024-11-07 {
      description
        "Initial revision";
      reference
        "RFC XXXX:
         NETCONF Extension to support Trace Context propagation";
    }
  }
  <CODE ENDS>
```

3.3. YANG module for tracestate header version 1.0

```
  <CODE BEGINS> file "ietf-trace-ctx-tracestate-1.0@2024-11-07.yang"
  module ietf-trace-ctx-tracestate-1.0 {
    namespace
      "urn:ietf:params:xml:ns:yang:ietf-trace-ctx-tracestate-1.0";
    prefix ietf-trace-ctx-tracestate-1.0;

    organization
      "IETF NETCONF (Network Configuration) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
       WG List:  <mailto:netconf@ietf.org>

       Authors:  Roque Gagliano
                 <mailto:rogaglia@cisco.com>

                 Jan Lindblad
                 <mailto:jlindbla@cisco.com>

                 Kristian Larsson
                 <mailto:kll@dev.terastrm.net>
```

```
      ";
    description
      "This module documents the support for trace context tracestate
       versiom 1.0 in alignment with W3C versions:
       https://www.w3.org/TR/2021/REC-trace-context-1-20211123

       Copyright (c) 2024 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
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices

       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 (RFC 2119) (RFC 8174) when, and only when,
       they appear in all capitals, as shown here.
      ";

    revision 2024-11-07 {
      description
        "Initial revision";
      reference
        "RFC XXXX:
         NETCONF Extension to support Trace Context propagation";
    }
  }
  <CODE ENDS>
```

4. Security Considerations

The YANG modules specified in this document are used to flag capabilities define and define an error information structure that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040].

As such, these YANG modules do not contain any configuration data, state data or RPC definitions, which makes their security implications very limited. The additional attributes specified in this document (but not in YANG modules, since YANG cannot be used to specify attributes) are worth mentioning, however.

The traceparent and tracestate attributes make it easier to track the flow of requests and their downstream effect on other systems. This is indeed the whole point with these attributes. This knowledge could also be of use to bad actors that are working to build a map of the managed network.

The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

5. 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://www.rfc-editor.org/rfc/rfc3688.html).

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

Registrant Contact: The IETF IESG.

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

This document registers three module names in the 'YANG Module Names' registry, defined in RFC 6020:

```
     name: ietf-trace-ctx-traceparent-1.0

     prefix: ietf-trace-ctx-traceparent-1.0

     namespace:
     urn:ietf:params:xml:ns:yang:ietf-trace-ctx-traceparent-1.0

     RFC: XXXX

   and

     name: ietf-trace-ctx-tracestate-1.0

     prefix: ietf-trace-ctx-tracestate-1.0

     namespace:
     urn:ietf:params:xml:ns:yang:ietf-trace-ctx-tracestate-1.0

     RFC: XXXX

   and

     name: ietf-trace-context

     prefix: ietf-trace-context

     namespace: urn:ietf:params:xml:ns:yang:trace-context

     RFC: XXXX
```

6. Acknowledgments

The authors would like to acknowledge the valuable implementation feedback from Christian Rennerskog and Per Andersson. Many thanks to Raul Rivas Felix, Alexander Stoklasa, Luca Relandini and Erwin Vrolijk for their help with the demos regarding integrations. The help and support from Jean Quilbeuf and Benoit Claise has also been invaluable to this work.

7. References

7.1. 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/rfc/rfc2119>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/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/rfc/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/rfc/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/rfc/rfc8446>.

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

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

7.2. Informative References

[gNMI] "gNMI - gRPC Network Management Interface", 4 November 2024, <https://github.com/openconfig/gnmi>.

[I-D.ietf-netconf-transaction-id] Lindblad, J., "Transaction ID Mechanism for NETCONF", Work in Progress, Internet-Draft, draft-ietf-netconf-

transaction-id-08, 20 February 2025,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-transaction-id-08>.

[OpenTelemetry]
"OpenTelemetry Cloud Native Computing Foundation project", 4 November 2024, <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/rfc/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/rfc/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 (to be deleted by RFC Editor)

A.1. From version 03 to version 04

* WGLC data change.

A.2. From version 02 to 03

* Changed document Abbreviation
* trace-context-error-info is a container in example

A.3. From version 01 to 02

* Enhanced Terminology and moved it up in the document.
* Changed namespaces and module names to map WGLC comments and IETF requirements

A.4. From version 00 to 01

* Added Security considerations
* Added Acknowledgements
* Added several Normative references

* Updated link to latest document on github
* Firmed up error handling and YANG-library to MUST-requirements

A.5. From version 03 to draft-ietf-netconf-trace-ctx-extension-00

* Adopted by NETCONF WG
* Moved repository to NETCONF WG
* Changed build system to use martinthomson’s excellent framework
* Ran make fix-lint to remove white space at EOL etc.
* Added this change note. No other content changes.

A.6. From version 02 to 03

* Changed IANA section to IESG per IANA email
* Created sx:structure and improved error example
* Added ietf-netconf-otlp-context.yang for the sx:structure
* Created a dedicated section for the YANG modules

A.7. From version 01 to 02

* Added Error Handling intial section
* Added how to manage versioning by defining YANG modules for each traceparent and trastate versions as defined by W3C.
* Added ’YANG Module Names’ to IANA Considerations

A.8. 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.ietf-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. 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