

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 02, 2014

R. Varga
Pantheon Technologies SRO
March 3, 2014

Efficient XML Interchange Capability for NETCONF
draft-varga-netconf-exi-capability-02

Abstract

The Network Configuration Protocol (NETCONF) provides mechanisms to install, manipulate, and delete the configuration of network devices via exchange of XML messages in textual representation. Efficient XML Interchange (EXI) is a W3C-recommended binary representation of XML Information Set, which is more efficient from both CPU and bandwidth utilization perspective. This document defines a capability-based extension to the NETCONF protocol that allows peers to agree to exchange protocol messages using EXI encoding.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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 <http://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 September 02, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. EXI Capability	3
3.1. Overview	3
3.2. Dependencies	3
3.3. Capability Identifier	3
3.4. Dynamic Schema-informed Encoding Negotiation	4
3.5. New Mandatory Operations	5
3.5.1. <start-exi>	5
3.5.2. <stop-exi>	8
3.6. New Optional Operations	8
3.6.1. <enable-schema-encoding>	8
3.6.2. <disable-schema-encoding>	9
4. YANG module for <start-exi> and <stop-exi> Operations	9
5. IANA considerations	14
6. Security Considerations	14
7. Acknowledgements	14
8. Normative References	14
Author's Address	14

1. Introduction

The NETCONF protocol [RFC6241] is defined in terms of two peers, client and server, exchanging XML messages in an RPC pattern. These messages are encoded as XML text documents, which makes the exchange easily understandable by human operators by simply observing them on the wire. Unfortunately, this feature takes its toll on both computation and network resources, as the representation contains redundant information and verbose names.

Efficient XML Interchange [W3C.REC-exi-20110310] is a W3C Recommendation which defines a more efficient way of encoding XML Information Set than the usual text representation. This is achieved by a combination of reduced verbosity, binary encoding and, optionally, pruning of non-essential information like comments.

It seems advantageous to allow clients and servers participating on a NETCONF session to sacrifice human readability to increase processing efficiency, especially in environments with high transactional activity and/or limited computing resources.

2. Terminology

This document uses the following terms defined in [RFC6241]:

- o capability
- o client
- o message
- o protocol operation
- o remote procedure call
- o server

3. EXI Capability

3.1. Overview

The :exi capability indicates that the peer supports EXI message encoding and is willing to use it. The capability has no effect on data being handled by the NETCONF protocol, nor does it affect protocol message exchanges.

3.2. Dependencies

EXI-encoded documents are binary data, this capability may only be used when the underlying transport is 8-bit clean and preserves message boundaries in face of arbitrary binary data. Notably this requires use of Chunked Framing mechanism as described in [RFC6242] when used with SSH transport.

The optional Dynamic Schema-informed Encoding Negotiation mechanism relies on NETCONF Monitoring as defined in [RFC6022].

3.3. Capability Identifier

The EXI capability is identified by the following capability string:

urn:ietf:params:netconf:capability:exi:1.0

The identifier MAY have a the following parameters:

compression: This indicates that the sender is willing to perform EXI compression. The parameter MUST contain a positive integral value, which indicates maximum compression block size which the sender can process.

schemas: This indicates that the sender can use schema-informed grammars for EXI encoding. The parameter MUST contain a value, which has to be one of "builtin", "base:1.1" or "dynamic".

builtin Indicates the ability to use the XML schema built into the EXI specification.

base:1.1 Superset of "builtin", indicates that the sender additionally supports schema-informed EXI encoding, based on netconf.xsd schema published in [RFC6241].

dynamic Superset of "base:1.1", indicates that the sender additionally supports dynamic schema-informed encoding negotiation outlined below.

Examples:

urn:ietf:params:netconf:capability:exi:1.0?compression=1000000

urn:ietf:params:netconf:capability:exi:1.0?schemas=builtin

urn:ietf:params:netconf:capability:exi:1.0?schemas=base:1.1

urn:ietf:params:netconf:capability:exi:1.0?compression=20000&schemas=builtin

urn:ietf:params:netconf:capability:exi:1.0?schemas=dynamic

3.4. Dynamic Schema-informed Encoding Negotiation

The core of this extension relies on shared knowledge between the server and the client where schema-informed encoding is concerned. This limits the encoding efficiency as the actual data transferred over the session is encoded using the equivalent of the builtin schema. Alleviating this limitation requires a mechanism for discovering data schemas and a protocol for synchronizing their activation.

The base schema discovery mechanism is already present in [RFC6022]. This document extends the /netconf-state/schemas/schema subtree with a new leaf, exi-useable, which indicates whether the server supports the use of that particular schema in the EXI schema-informed encoding process.

The negotiation of use of a particular schema for encoding has multiple aspects. First and foremost is the concern of constrained environments, which may have limited resources and thus their ability to dedicate them to improving encoding efficiency may change over lifetime of a NETCONF session. The second issue comes from the need to synchronize the values used in the "schema" EXI header field. Both end of the session need to map names to the same schemas, otherwise the decoding process will not succeed. This name is carried verbatim in the stream, so it should be as concise as possible.

When the peers have both indicated support for Dynamic Schema-informed Encoding, encoding starts in base:1.1 mode. The client then queries the server for the list of schemas, looking for schemas which have the `exi-useable` leaf set to true. It then selects the schemas it can use in EXI encoding process, potentially requesting them from the server. Finally it prioritizes them and sends a `<enable-schema-encoding>` request for each of them. Once the server has assigned a EXI schema-id and communicated it back to the client, both parties can use this schema in EXI encoding. The client can request the end of use of a particular schema via the `<disable-schema-encoding>` RPC, which the server SHOULD NOT fail.

3.5. New Mandatory Operations

3.5.1. `<start-exi>`

Description: The `<start-exi>` operation requests that the message encoding be switched to EXI. The operation is invoked by the client and validated by the server. If the server finds the parameters acceptable, it will issue a positive response in the current session encoding. It MUST encode all subsequent messages using EXI encoding with the supplied parameters. It will also expect all incoming messages to be EXI-encoded. The client MUST NOT send any messages to the server between the time it sends this request and the time it receives a response. Once it receives a positive reply, it MUST encode all subsequent messages using the EXI encoding with the parameters supplied in the RPC. If the operation fails, the session message encoding remains unchanged.

Parameters:

`alignment`: Requested EXI alignment. If this parameter is not present, bit-packed is assumed. The following values are valid:

`bit-packed`: Set EXI alignment to bit-packed.

`byte-aligned`: Set EXI alignment to byte-aligned.

`pre-compression`: Set EXI alignment to pre-compression.

`compressed`: Do not specify EXI alignment, but perform EXI compression instead.

`fidelity`: Requested EXI fidelity options. If this parameter is not present or empty, all fidelity options are disabled. The

following items may be specified:

<comments/>: Preserve.comments EXI Fidelity option

<dtd/>: Preserve.dtd EXI Fidelity option

<lexical-values/>: Preserve.lexicalValues EXI Fidelity option

<pis/>: Preserve.pis EXI Fidelity option

<prefixes/>: Preserve.prefixes EXI Fidelity option

schema: Optional parameter. This specifies what schema options should be enabled in the EXI encoding process. The following values are valid:

none Do not use schema-informed grammars at all. This translates to using schemaId of <xsd:nil>true</xsd:nil> in the EXI Options header.

builtin Do not use schema-informed grammars, but use the built-in XML data types. This translates to using an empty schemaId in the EXI Options header.

base:1.1 Use schema-informed grammar based on netconf.xsd as published in [RFC6241] in non-strict mode. The value "base:1.1" should be carried in the schemaId field in the EXI Options.

dynamic Same as base:1.1 with the additional support for dynamically modifying which schemas are available for schema-informed encoding.

Positive Response: If the device was able to satisfy the request, an <rpc-reply> is sent that contains an <ok> element.

Negative Response: An <rpc-error> element is included in the <rpc-reply> if the request cannot be completed for any reason.

Example:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <start-exi>
    <alignment>pre-compression</alignment>
    <fidelity>
      <dtd/>
      <lexical-values/>
    </fidelity>
  </start-exi>
```

```
</rpc>

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

3.5.2. <stop-exi>

Description: The <stop-exi> operation requests that the message encoding be switched to textual XML. The operation is invoked by the client and validated by the server. If the server is able to switch the encoding to XML, it will issue a positive response in the current session encoding. It MUST encode all subsequent messages using standard XML encoding. It will also expect all incoming messages to be XML-encoded. The client MUST NOT send any messages to the server between the time it sends this request and the time it receives a response. Once it receives a positive reply, it MUST encode all subsequent messages using the standard XML encoding. If the operation fails, the session message encoding remains unchanged. If the session currently uses XML encoding, this RPC is a no-operation and SHOULD NOT fail.

Positive Response: If the device was able to satisfy the request, an <rpc-reply> is sent that contains an <ok> element.

Negative Response: An <rpc-error> element is included in the <rpc-reply> if the request cannot be completed for any reason.

Example:

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

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

3.6. New Optional Operations

3.6.1. <enable-schema-encoding>

Description: The <enable-schema-encoding> requests the device assign a numeric identifier for use of a specific schema for EXI Schema-informed encoding.

Parameters:

identifier: Schema identifier, as defined in [RFC6022].

version: Schema version, as defined in [RFC6022].

format: Schema format, as defined in [RFC6022].

Positive Response: If the device was able to satisfy the request, an `<rpc-reply>` is sent that contains an `<exi-schema-id>` element, which contains the numeric identifier which should be used in the schemaId EXI header field. This identifier has to be unique.

Negative Response: An `<rpc-error>` element is included in the `<rpc-reply>` if the request cannot be completed for any reason.

3.6.2. `<disable-schema-encoding>`

Description: The `<disable-schema-encoding>` requests the device to deallocate the schema ID from use on this session and stop using it for encoding data towards the client.

Parameters:

exi-schema-id: EXI Schema ID, as assigned by a previous `<enable-schema-encoding>` call.

Positive Response: If the device was able to satisfy the request, an `<rpc-reply>` is sent that contains an `<ok>` element.

Negative Response: An `<rpc-error>` element is included in the `<rpc-reply>` if the request cannot be completed for any reason.

4. YANG module for `<start-exi>` and `<stop-exi>` Operations

The following YANG module defines the new operations introduced in this document. The YANG language is defined in [RFC6020]. Every NETCONF speaker that supports the :exi capability MUST implement this YANG module.

```
<CODE BEGINS> file "ietf-netconf-exi@2014-03-03.yang"

module ietf-netconf-exi {
  // vi: set et smarttab sw=4 tabstop=4:
  namespace "urn:ietf:params:xml:ns:netconf:exi:1.0";

  prefix exi;

  import ietf-netconf-monitoring {
    prefix ncm;
    revision-date "2010-10-04";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "Robert Varga <robert.varga@pantheon.sk>";

  description
    "NETCONF Protocol Operations for Efficient XML Interchange.";

  revision 2014-03-03 {
    description
      "Updated with dynamic schema negotiation.";
    reference
      "I-D.varga-netconf-exi-capability-02";
  }

  revision 2013-10-21 {
    description
      "Initial revision";
    reference
      "I-D.varga-netconf-exi-capability-01";
  }

  typedef exi-alignment {
    type enumeration {
      enum bit-packed {
        description
          "Use bit-packed EXI alignment";
      }
      enum byte-aligned {
        description
          "Use byte-aligned EXI alignment";
      }
      enum pre-compression {
        description
          "Use pre-compression EXI alignment";
      }
      enum compressed {
        description
```

```
        "Do not set EXI alignment, use EXI compression
        instead";
    }
}

description "EXI alignment specification.";
}

typedef exi-fidelity {
    type enumeration {
        enum comments {
            description
                "Preserve.comments EXI Fidelity option";
        }
        enum dtd {
            description
                "Preserve.dtd EXI Fidelity option";
        }
        enum lexical-values {
            description
                "Preserve.lexicalValues EXI Fidelity option";
        }
        enum pis {
            description
                "Preserve.pis EXI Fidelity option";
        }
        enum prefixes {
            description
                "Preserve.prefixes EXI Fidelity option";
        }
    }
}

description "EXI fidelity options.";
}

rpc start-exi {
    description
        "Start encoding protocol messages in Efficient XML
        Interchange format.";

    reference "I-D.varga-netconf-exi-capability";

    input {
        leaf alignment {
            type exi-alignment;
            default "bit-packed";
            description "EXI alignment to use.";
        }

        leaf-list fidelity {
            type exi-fidelity;
            description "EXI fidelity options to use.";
        }
    }
}
```

```
    }

    rpc stop-exi {
        description
            "Stop encoding protocol messages in Efficient XML
            Interchange format. Revert back to using the usual text
            XML encoding.";
    }

    grouping schema-identifier {
        description
            "The globally-unique identifier of a schema. This
            grouping contains the verbatim transcription of arguments
            to <get-schema> RPC as defined in RFC6022, except all
            leaves are made mandatory.";

        leaf identifier {
            type string;
            mandatory true;
            description
                "Identifier for the schema list entry.";
        }

        leaf version {
            type string;
            description
                "Version of the schema requested.  If this parameter
                is not present, and more than one version of the
                schema exists on the server, a 'data-not-unique'
                error is returned, as described above.";
        }

        leaf format {
            type identityref {
                base ncm:schema-format;
            }
            description
                "The data modeling language of the schema.  If this
                parameter is not present, and more than one formats
                of the schema exists on the server, a
                'data-not-unique' error is returned, as described
                above.";
        }
    }

    typedef exi-schema-id {
        type uint16;
        description
            "Schema identifier for use in the EXI stream header.";
    }

    augment "/ncm:netconf-state/ncm:schemas/ncm:schema" {
        description
            "Additional information about schemas useful for EXI
```

```
        encoding";

    leaf exi-useable {
        type boolean;
        default false;
        description
            "Set to true if the device can use the schema for EXI
            Schema-informed encoding.";
    }
    leaf exi-schema-id {
        type exi-schema-id;
        description
            "The EXI schema ID currently assigned to this schema.
            This value has meaning only within the session and
            may differ on other sessions.";
    }
}

rpc enable-schema-encoding {
    description
        "Request the use of specificied schema in EXI message
        encoding. This request is sent by the client to the
        server. If the server is able to transition into using
        the schema, it assigns it a unique EXI integer
        identifier. This identifier is to be used in the EXI
        header as schema identifier.

        The server may start using the identifier as soon as it
        enqueus the response. The client may start using the
        identifier as soon as it sees this RPC complete.";
    input {
        uses schema-identifier;
    }
    output {
        leaf exi-schema-id {
            type exi-schema-id;
            mandatory true;
            description
                "The EXI Schema ID assigned to this schema for
                encoding purposes.";
        }
    }
}

rpc disable-schema-encoding {
    description
        "This RPC is send by the client when it stops using a
        particular exi-schema-id.";
    input {
        leaf exi-schema-id {
            type exi-schema-id;
            mandatory true;
            description
                "The EXI Schema ID which should be disabled.";
        }
    }
}
```

```
    }  
  }  
}
```

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:exi:1.0

6. Security Considerations

The compression option present in EXI specification may increase CPU and memory requirements for encoding the response. Devices should ensure this overhead is acceptable before agreeing to using EXI encoding, such that no operational risks are introduced.

7. Acknowledgements

The author would like to thank Anton Tkacik, Miroslav Miklus and Stefan Kobza for their contributions to this document.

8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6022] Scott, M. and M. Bjorklund, "YANG Module for NETCONF Monitoring", RFC 6022, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [W3C.REC-exi-20110310] Schneider, J. and T. Kamiya, "Efficient XML Interchange (EXI) Format 1.0", World Wide Web Consortium Recommendation REC-exi-20110310, March 2011, <<http://www.w3.org/TR/2011/REC-exi-20110310>>.

Author's Address

Robert Varga
Pantheon Technologies SRO
Mlynske Nivy 56
Bratislava 821 05
Slovakia

Email: robert.varga@pantheon.sk