Modular RELAX NG Schema of NETCONF RPC and Protocol Operations
draft-lhotka-netconf-relaxng-00

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 3, 2009.

Copyright Notice

Copyright (c) 2009 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.
Abstract

This memo presents a schema for NETCONF RPC and protocol operations expressed in RELAX NG (compact syntax). The schema is modular and cleanly separates the server and client part of the NETCONF vocabulary and also the schema extensions provided by optional capabilities. The modular structure improves readability but also enables selecting certain modules and assembling them into a grammar that can be used for validation of NETCONF protocol data units.

Table of Contents

1. Introduction ......................................................... 3
2. Problems of the NETCONF XSD Schema .............................. 5
3. Description of the RELAX NG Schemas ............................. 6
4. Assembling the Modules into Validation Schemas ................. 8
5. Conclusions ......................................................... 9
6. References .......................................................... 10

Appendix A. RELAX NG Schema Modules ................................. 11
   A.1. Common patterns ................................................. 11
   A.2. Base Client Schema ............................................. 12
   A.3. Base Server Schema ............................................. 14
   A.5. Capability :candidate ........................................... 20
   A.6. Capability :confirmed-commit .................................. 21
   A.7. Capability :rollback-on-error .................................. 21
   A.8. Capability :validate ............................................. 21
   A.9. Capability :startup .............................................. 22
   A.10. Capability :url ................................................ 22
   A.11. Capability :xpath .............................................. 23

Author’s Address ....................................................... 24
1. Introduction


However, the NETCONF protocol vocabulary actually consists of two distinct parts - one for the NETCONF client and the other for the NETCONF server - that can never appear together in the same NETCONF PDU. The overlap of these two parts is small and even if the same element, such as <hello>, is allowed in both, the content model of each version may differ. Moreover, the PDU contents in every particular case depend on the capabilities supported by the server and negotiated in the <hello> messages. The all-encompassing approach of the NETCONF XSD schema cannot take these differences into account and the grammar is thus in many places too liberal. As a result, the XSD schema can hardly be used for serious validation of NETCONF PDUs.

This report introduces a new modular schema for the same NETCONF protocol vocabulary expressed in RELAX NG [4], a simple but powerful schema language that became Part 2 of the international standard ISO/IEC 19757 [5]. RELAX NG has two official syntaxes, XML and compact, the latter, being designed primarily for human readers without special training, is considerably more suitable for inclusion in standardization documents such as RFCs than XML-based syntaxes.

The goal of this work is twofold:

1. Demonstrate that RELAX NG is a sound alternative to XSD for the given purpose in that it is able to express the same (or even more detailed) grammar and data-typing rules as the existing NETCONF XSD schema and do it in a way that is considerably more human-readable.

2. Use the extensibility framework of RELAX NG for dividing the schema into smaller modules according to the logic of the NETCONF protocol. Such a decomposition will not only further aid readability and make the parts of the schema easier to maintain, but also allow for using the modules directly for NETCONF PDU validation.

The report is organized as follows: In the Section 2 we summarize the problems of the NETCONF XSD schema, Section 3 describes the structure and main design features of the modular RELAX NG schema, Section 4 shows how to use the modules for effective validation of NETCONF PDUs in specific contexts and finally Section 5 concludes the reports.
Appendix A contains full listing of the RELAX NG modules in the compact syntax.
2. Problems of the NETCONF XSD Schema

The NETCONF XSD schema in Appendix B of [1] can serve well as a set of formalized guidelines for implementers of the NETCONF protocol, especially the RPC layer. However, this schema is much less useful for the second major purpose of XML schemas, namely for validating documents, in our case NETCONF PDUs. The reason for this deficiency is that many of the constraints in the schema are rather lax or even logically inconsistent. The latter problem is manifested, for example, in the <session-id> parameter of the <hello> element: the schema declares it as optional (minOccurs="0") but, in fact, this content model is appropriate neither for server’s nor for client’s version of <hello> - it is mandatory in the former but not allowed at all in the latter.

Similarly, the borderline case of a <rpc-reply> to an invalid <rpc> lacking the "message-id" parameter led the schema designers to declaring this parameter as optional. As an unfortunate result, any <rpc-reply> with "message-id" missing will be found valid.

Strict validation of NETCONF PDUs in a particular context must also take into account the set of capabilities supported by the concrete server. The XSD schema is not designed to allow such specialization.

Finally, Appendix A of [1] defines the standard NETCONF errors and specifies constraints on their contents. For example, some combinations or "error-tag" and "error-type" parameters are not permitted, some errors have prescribed content of "error-info" while others cannot have this parameter, and all these standard errors have severity "error". Again, the XSD schema doesn’t take these constraints into account and allows all possible combinations of error parameters.
3. Description of the RELAX NG Schemas

Modular RELAX NG schema is designed for much stricter validation of NETCONF PDUs than the XSD schema. This is achieved by utilizing auxiliary information that is typically available, such as the originator of the PDU - server or client - and list of capabilities supported by the server.

The RELAX NG schema is divided into 11 modules, each of them stored in a separate file:

- nc-base-common (Appendix A.1)
- nc-base-client (Appendix A.2)
- nc-base-server (Appendix A.3)
- nc-writable-running-client (Appendix A.4)
- nc-candidate-client (Appendix A.5)
- nc-confirmed-commit-client (Appendix A.6)
- nc-rollback-on-error-client (Appendix A.7)
- nc-validate-client (Appendix A.8)
- nc-startup-client (Appendix A.9)
- nc-url-client (Appendix A.10)
- nc-xpath-client (Appendix A.11)

The file names are composed of the module name and the appropriate extension - either ".rng" or ".rnc", depending on whether the file contains RELAX NG in the XML or compact syntax.

The main division line in the schema goes between the client and server part. Hence, modules "nc-base-client" and "nc-base-server" contain the client and server vocabulary, respectively. However, this is only the bare-bones NETCONF, without any optional capabilities. Few parameters and RELAX NG patterns are common to both the client and server schemas and these are contained in the "nc-base-common" module.

Further, extensions to the schema provided by optional capabilities are defined in separate modules. In general, there would be two modules per capability - one with the server and the other with the
client extensions. However, all the standard extensions defined in [1] only affect the client vocabulary, so the server module is not needed. Nevertheless, other capabilities may extend or change the server vocabulary, too, for example by introducing new error types.

The schema modules are designed for maximum extensibility. Therefore, their RELAX NG patterns mainly use the "content-oriented" style that allows new content to be easily added without redefining entire patterns. Due to this design, the capability modules can be generally very short and easy to understand. However, taking the content-oriented design to the extreme would make the schemas unwieldy, so we use it only in places where extensions are likely.
4. Assembling the Modules into Validation Schemas

Two of the RELAX NG modules - "nc-base-client" and "nc-base-server" - are full-fledged grammars that can be readily used for validating NETCONF PDUs generated by the client and server, respectively. So the following PDU (taken from [1]) will successfully validate against the "nc-base-client" schema:

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     message-id="101">
  <get-config>
    <source>
      <running/>
    </source>
  </get-config>
</rpc>
```

However, if we replace the `<running/>` element in the "source" parameter with `<startup/>`, the validation fails since the optional capabilities such as :startup are not supported by the "nc-base-client" schema. We can allow `<startup/>` as the value of the "source" parameter simply by using the following schema:

```xml
include "nc-base-client.rnc"
include "nc-startup-client.rnc"
```

In the same way, we can include any combination of capabilities as long as they are not mutually exclusive.

For the server PDUs the procedure is analogical but generally simpler since none of the standard capabilities defined in [1] affects the server schema, so the "nc-base-server" schema is sufficient for validating server PDUs.
5. Conclusions

The set of RELAX NG schemas presented in this report provide a modular grammar that can be used as a formal definition and documentation of the NETCONF vocabulary, but also for validating NETCONF PDUs. As a validation vehicle, though, these schemas are considerably stricter than the W3C XML Schema from [1]. This is mainly due to the fact that each of the RELAX NG schema modules deals with a certain specific part of the vocabulary – client or server side of a NETCONF channel, or individual capabilities. The modules can be assembled in many different ways into validation schemas that may be used for effective NETCONF PDU validation.

The fact that the modular RELAX NG schema is stricter and more validation-oriented than its XSD counterpart doesn’t mean that it is less readable. On the contrary, separating the server and client vocabularies and factoring out the optional capabilities resulted in base server and client schemas that are simple and easy to comprehend. Likewise, most capability schemas are almost trivial but still show quite precisely where and how each capability extends the vocabulary.

The simplicity and limited scope of the schema modules also enables their inclusion in larger validation frameworks such as DSDL [5] and/or combination with a NETCONF concrete data model expressed in RELAX NG or a different data modelling language.

As a matter of fact, most improvements described in this report could, and probably should, be applied to the NETCONF XSD schema in [1], too. The only notable exception is the compact syntax, which is special to RELAX NG.
6. References


Appendix A. RELAX NG Schema Modules

This appendix contains listings of the RELAX NG modules in the compact syntax. The modules cover the NETCONF client and server vocabularies including all optional capabilities described in RFC 4741 [1].

A.1. Common patterns

This module defines several parameters patterns that are referenced in both the client and server schemas.

BEGIN

# source file: nc-base-common.rng
default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

## This pattern specifies the URI for the base NETCONF capability.
## See RFC 4741, sec. 10.1.
basic-capability-uri = "urn:ietf:params:netconf:base:1.0"

hello-element = element hello { hello-content }

## Common part of the "hello" element. See RFC 4741, sec. 8.1.
hello-content =
    element capabilities {
        element capability { basic-capability-uri }
        & element capability { extra-capability-choice }*
    }

## URNs of additional known capabilities
extra-capability-choice =
    "urn:ietf:params:netconf:capability:writable-running:1.0"
    | "urn:ietf:params:netconf:capability:candidate:1.0"
    | "urn:ietf:params:netconf:capability:confirmed-commit:1.0"
    | "urn:ietf:params:netconf:capability:rollback-on-error:1.0"
    | "urn:ietf:params:netconf:capability:validate:1.0"
    | "urn:ietf:params:netconf:capability:startup:1.0"
    | xsd:anyURI {
       pattern =
           "urn:ietf:params:netconf:capability:url:1.0" ~
           "[?]scheme=(http|ftp|file)"
      }
    | "urn:ietf:params:netconf:capability:xpath:1.0"

## Attribute "message-id" is mandatory in both "rpc" and "rpc-reply".
## See RFC 4741, sec. 4.1.
message-id-attribute =
  attribute message-id {
    xsd:string { maxLength = "4095" }
  }

## The "session-id" parameter identifies a NETCONF session.
session-id-element = element session-id { xsd:positiveInteger }

## This pattern allows arbitrary content.
arbitrary-content =
  (element * { arbitrary-content }
   | attribute * { text }
   | text)*

## This pattern allows arbitrary content wrapped in an element.
wrapped-arbitrary-content = element * { arbitrary-content }

END

A.2. Base Client Schema

This module specifies the NETCONF client vocabulary without any optional capabilities.

BEGIN

# source file: nc-base-client.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
include "nc-base-common.rnc"

## Client messages may have either "hello" or "rpc" as the document element. "hello" is reused from netconf-common without change.
## start = hello-element | rpc-element

## This element is a wrapper for all RPC requests.
## See RFC 4741, Sec. 4.1.
rpc-element =
  element rpc {
    message-id-attribute,
    attribute * { text }*,
    rpc-operation-choice
  }

## Selection of RPC operations. See RFC 4741, Sec. 7.
## Capabilities may extend this pattern.
rpc-operation-choice =
element get-config { get-config-content }
  element edit-config { edit-config-content }
  element copy-config { copy-config-content }
  element delete-config { delete-config-content }
  element lock { lock-content }
  element unlock { unlock-content }
  element get { get-content }
  element close-session { close-session-content }
  element kill-session { kill-session-content }

## See RFC 4741, Sec. 7.1.
get-config-content =
  element source { source-choice }
  & element filter { filter-content-choice }?

## This element represents the "running" datastore.
running-element = element running { empty }

## Available repositories. Capabilities may extend this pattern.
source-choice = running-element | extra-datastore-choice

## No additional datastores are present in base NETCONF.
## Capabilities may extend this pattern. See RFC 4741, Sec. 5.1.
extra-datastore-choice = empty

## Base NETCONF only supports subtree filters.
## Capabilities may extend this pattern.
filter-content-choice =
  attribute type { "subtree" }?,
  subtree-filter-content

## Content of a subtree filter depends on the data model so
## we have to allow everything here.
subtree-filter-content = wrapped-arbitrary-content

## See RFC 4741, Sec. 7.2.
edit-config-content =
  element target { target-choice }
  & element default-operation { default-operation-choice }?
  & element error-option { error-option-choice }?
  & config-data-spec

## The "operation" attribute may be used inside the "config" parameter
## of edit-config. This pattern may be combined with particular data
## models into a schema validating entire "edit-config" PDUs.
edit-config-operation-attribute =
  attribute operation { "merge" | "replace" | "create" | "delete" }
## Writable repositories (in base NETCONF there are none).
## Capabilities may extend this pattern.

```
target-choice = extra-datastore-choice
```

## Possible values for "default-operation" parameter in "edit-config".
## Default is "merge".

```
default-operation-choice = "merge" | "replace" | "none"
```

## Possible values for "error-option" parameter in "edit-config".
## Default is "stop-on-error".

```
error-option-choice = "stop-on-error" | "ignore-error"
```

## This pattern gives possible options for specifying configuration
## data. In base NETCONF, only inline tree is allowed.
## Capabilities may extend this pattern.

```
config-data-spec = element config { wrapped-arbitrary-content }
```

## See RFC 4741, Sec. 7.3.

copy-config-content =  
  element source { source-choice }  
  & element target { target-choice }

## See RFC 4741, Sec. 7.4.

delete-config-content = element target { extra-datastore-choice }

## See RFC 4741, Sec. 7.5.

lock-content = element target { target-choice }

## See RFC 4741, Sec. 7.6.

unlock-content = element target { target-choice }

## See RFC 4741, Sec. 7.7.

get-content = element filter { filter-content-choice }?

## See RFC 4741, Sec. 7.8.

close-session-content = empty

## See RFC 4741, Sec. 7.9.

kill-session-content = session-id-element

END

### A.3. Base Server Schema

This module specifies the NETCONF server vocabulary without any optional capabilities.

BEGIN

Lhotka

Expires September 3, 2009

[Page 14]
# source file: nc-base-server.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

include "nc-base-common.rnc"

## Server messages may have either "hello" or "rpc-reply" as the document element.
start = hello-element | rpc-reply-element

## Server’s "hello" message must include "session-id".
## See RFC 4741, Sec. 8.1.
hello-content &amp;= session-id-element

## This element is a wrapper for all RPC replies.
## See RFC 4741, Sec. 4.2.
rpc-reply-element =
  element rpc-reply {
    attribute * { text }*,
    ((# Positive response
      message-id-attribute,
      element rpc-reply { rpc-reply-choice })
    | (# error reply to a rpc without message-id is special
      missing-message-id-error
    | (# Negative response
      message-id-attribute,
      element rpc-error { rpc-error-choice }+)))
  }

## Specific error reply to an "rpc" without message-id.
## Only in this case can "message-id" be missing in an "rpc-reply".
missing-message-id-error =
  element rpc-error {
    element error-tag { "missing-attribute" }
    & element error-type { "rpc" }
    & element error-severity { "error" }
    & element error-info {
      element bad-attribute { "message-id" }
      & element bad-element { "rpc" }
    }
    & optional-error-content
  }

## Possible RPC replies. See RFC 4741, Sec. 4.2.
## Capabilities may extend this pattern.
rpc-reply-choice = ok-element | data-element
## See RFC 4741, Sec. 4.4.

### ok-element

```
ok-element = element ok { empty }
```

## Content depends on data model. Here we have to allow everything.

### data-element

```
data-element = element data { wrapped-arbitrary-content }
```

## These errors are defined in RFC 4741, Appendix A.

### rpc-error-choice

```
rpc-error-choice =
  in-use-error-content |
  too-big-error-content |
  missing-attribute-error-content |
  bad-attribute-error-content |
  unknown-attribute-error-content |
  missing-element-error-content |
  bad-element-error-content |
  unknown-element-error-content |
  unknown-namespace-error-content |
  access-denied-error-content |
  lock-denied-error-content |
  resource-denied-error-content |
  rollback-failed-error-content |
  data-exists-error-content |
  data-missing-error-content |
  operation-not-supported-error-content |
  operation-failed-error-content |
  partial-operation-error-content |
```

```
error-types-pa = "protocol" | "application"
```

```
error-types-rpa = error-types-pa | "rpc"
```

```
error-types-all = error-types-rpa | "transport"
```

### optional-error-content

```
optional-error-content =
  element error-app-tag { xsd:string }?
  & element error-path { xsd:string }?
  & element error-message {
    attribute xml:lang { xsd:token }?,
    xsd:string
  }?
```

```
element-error-info = element bad-element { xsd:QName }
```

```
attribute-error-info =
  element bad-attribute { xsd:QName }
  & element-error-info
```

## Resource in use

### in-use-error-content

```
in-use-error-content =
  element error-tag { "in-use" }
  & element error-type { error-types-pa }
  & element error-severity { "error" }
  & optional-error-content
```
## Response would be too large for the implementation

```
too-big-error-content =
  element error-tag { "too-big" }
  & element error-type { error-types-all }
  & element error-severity { "error" }
  & optional-error-content
```

## A required attribute is missing

```
missing-attribute-error-content =
  element error-tag { "missing-attribute" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info { attribute-error-info }
  & optional-error-content
```

## Incorrect attribute value

```
bad-attribute-error-content =
  element error-tag { "bad-attribute" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info { attribute-error-info }
  & optional-error-content
```

## An unexpected attribute is present

```
unknown-attribute-error-content =
  element error-tag { "unknown-attribute" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info { attribute-error-info }
  & optional-error-content
```

## A required element is missing

```
missing-element-error-content =
  element error-tag { "missing-element" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info { element-error-info }
  & optional-error-content
```

## Incorrect element value

```
bad-element-error-content =
  element error-tag { "bad-element" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info { element-error-info }
  & optional-error-content
```

## An unexpected element is present
unknown-element-error-content =
  element error-tag { "unknown-element" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info { element-error-info }
  & optional-error-content

## An unknown namespace is used
unknown-namespace-error-content =
  element error-tag { "unknown-element" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & element error-info {
    element-error-info
    & element bad-namespace { xsd:Name }
  }
  & optional-error-content

## Access denied due to authorization failure
access-denied-error-content =
  element error-tag { "access-denied" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & optional-error-content

## Access to the requested lock is denied
lock-denied-error-content =
  element error-tag { "lock-denied" }
  & element error-type { "protocol" }
  & element error-severity { "error" }
  & element error-info {
    element session-id { xsd:nonNegativeInteger }
  }
  & optional-error-content

## Insufficient resources
resource-denied-error-content =
  element error-tag { "resource-denied" }
  & element error-type { error-types-all }
  & element error-severity { "error" }
  & optional-error-content

## Rollback not completed.
## Requires either :rollback-on-error or :candidate capability.
rollback-failed-error-content =
  element error-tag { "rollback-failed" }
  & element error-type { error-types-pa }
  & element error-severity { "error" }
## Relevant data already exists

data-exists-error-content =
  element error-tag { "data-exists" }
  & element error-type { "application" }
  & element error-severity { "error" }
  & optional-error-content

## Relevant data missing

data-missing-error-content =
  element error-tag { "data-missing" }
  & element error-type { "application" }
  & element error-severity { "error" }
  & optional-error-content

## Operation not supported by the implementation

operation-not-supported-error-content =
  element error-tag { "operation-not-supported" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & optional-error-content

## Requested operation failed for some reason

operation-failed-error-content =
  element error-tag { "operation-failed" }
  & element error-type { error-types-rpa }
  & element error-severity { "error" }
  & optional-error-content

## Part of the operation failed or was not attempted

partial-operation-error-content =
  element error-tag { "partial-operation" }
  & element error-type { "application" }
  & element error-severity { "error" }
  & element error-info {
      element ok-element { xsd:QName }*
        & element err-element { xsd:QName }*
        & element noop-element { xsd:QName }*
    }
  & optional-error-content

This module specifies the extensions to the client schema provided by the :writable-running capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.2.
BEGIN

# source file: nc-writable-running-client.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

## This adds "running" as an option for the "target" parameter of NETCONF operations.
target-choice |= running-element

END

A.5. Capability :candidate

This module specifies the extensions to the client schema provided by the :candidate capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.3.
BEGIN

# source file: nc-candidate-client.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

## This adds "candidate" as an option for both "source" and "target" parameter of NETCONF operations.

extra-datastore-choice |= element candidate { empty }

## Two new RPC operations are added.
rpc-operation-choice |=
   element commit { commit-content }
   | element discard-changes { discard-changes-content }

## See RFC 4741, Sec. 8.3.4.1.
commit-content = empty

## See RFC 4741, Sec. 8.3.4.2.
discard-changes-content = empty

END
A.6. Capability :confirmed-commit

This module specifies the extensions to the client schema provided by the :confirmed-commit capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.4.
BEGIN

# source file: nc-confirmed-commit-client.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

## The "confirm-timeout" parameter is given in seconds.
## The default is 600. See RFC 4741, Sec. 8.4.5.1.
commit-content &=
    (element confirmed { empty }
    & element confirm-timeout { xsd:positiveInteger }?)?

END

A.7. Capability :rollback-on-error

This module specifies the extensions to the client schema provided by the :rollback-on-error capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.5.
BEGIN

# source file: nc-rollback-on-error-client.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

error-option-choice |= "rollback-on-error"

END

A.8. Capability :validate

This module specifies the extensions to the client schema provided by the :validate capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.6.
internet-draft             netconf data types             march 2009

begin

# source file: nc-validate-client.rng
default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

rpc-operation-choice |= element validate { validate-content }

## see rfc 4741, sec. 8.6.4.1.
validate-content = element source { source-choice | config-data-spec }

end

A.9. Capability :startup

This module specifies the extensions to the client schema provided by the :startup capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.7.

begin

# source file: nc-startup-client.rng
default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

## this adds "startup" as an option for both "source" and "target"
## parameter of netconf operations.
extra-datastore-choice |= element startup { empty }

end

A.10. Capability :url

This module specifies the extensions to the client schema provided by the :url capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.8.

begin

# source file: nc-url-client.rng
default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

source-choice |= element url { xsd:anyURI }
target-choice |= element url { xsd:anyURI }

end
A.11. Capability :xpath

This module specifies the extensions to the client schema provided by the :xpath capability. The server schema is unchanged. See RFC 4741 [1], Sec. 8.9.

BEGIN

# source file: nc-xpath-client.rng

default namespace = "urn:ietf:params:xml:ns:netconf:base:1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

filter-content-choice |=
    attribute type { "xpath" },
    attribute select { xsd:string }

END
Author’s Address

Ladislav Lhotka
CESNET
Zikova 4
Praha 6 160 00
CZ

Email: lhotka@cesnet.cz