Internet Engineering Task Force (IETF)
Request for Comments: 6241
Obsoletes: 4741
Category: Standards Track
ISSN: 2070-1721

R. Enns, Ed.
Juniper Networks
M. Bjorklund, Ed.
Tail-f Systems
J. Schoenwaelder, Ed.
Jacobs University
A. Bierman, Ed.
Brocade
June 2011

Протокол настройки сети (NETCONF)
Network Configuration Protocol (NETCONF)

Аннотация

Протокол настройки сети (NETCONF[1]), определённый в этом документе, обеспечивает механизмы установки, изменения и удаления конфигурационных параметров сетевых устройств. Он использует представление данных на основе расширяемого языка разметки (XML[2]) для параметров конфигурации и протокольных сообщений. Работа протокола NETCONF основана на удалённых вызовах процедур (RPC[3]). Данный документ служит заменой RFC 4741.

Статус документа

Этот документ содержит проект стандарта Internet (Internet Standards Track).

Документ является результатом работы IETF[4] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[5]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 5741.

Информация о статусе этого документа, обнаруженных ошибках и способах обратной связи доступна по ссылке http://www.rfc-editor.org/info/rfc6241.

Оглавление

[1]Network Configuration Protocol.
[2]Extensible Markup Language.
[3]Remote procedure call.
[4]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[5]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Протокол NETCONF определяет простой механизм, обеспечивающий возможность настройки сетевых устройств и получения от них параметров конфигурации. Протокол позволяет устройству представлять (показывать) полностью формальный интерфейс API[1]. Приложения могут использовать этот простой API для отправки и получения полного или частичного набора данных конфигурации устройства.

Протокол NETCONF использует парадигму удалённого вызова процедур (RPC). Клиент кодирует RPC в формат XML [W3C.REC-xml-20001006] и передаёт его серверу через защищённую, ориентированную на соединения сессию. Сервер возвращает отклик в формате XML. Содержимое запросов и откликов полностью описывается в XML DTD или схемах XML, позволяя сторонам учитывать синтаксические ограничения при обмене.

Основной особенностью NETCONF является то, что функциональность протокола управления весьма точно отражает функциональность самого устройства. Это снижает расходы на реализацию и обеспечивает своевременный доступ к новым возможностям. Кроме того, приложения могут получать доступ к семантическому и синтаксическому содержимому естественного интерфейса управления устройством.

NETCONF позволяет клиенту определить набор протокольных расширений, поддерживаемых сервером. Благодаря этому, клиент может настроить своё поведение так, чтобы в полной мере использовать представленные устройством

[1]Application programming interface.

возможности. Определения возможностей могут легко расширяться без централизованного управления. Стандартные и нестандартные возможности могут быть определены с семантической и синтаксической строгостью. Возможности рассматриваются в разделе 8.

Протокол NETCONF является одним из блоков построения автоматизированных систем настройки конфигурации. XML - это язык обмена информацией, обеспечивающий гибкий, но чётко определённый механизм представления иерархического содержимого. Протокол NETCONF может использоваться вместе с технологиями преобразования на основе XML, таких как XSLT [W3C.REC-xslt-19991116], для создания систем автоматической генерации полных или частичных наборов данных конфигурации. Система может запрашивать в одной или нескольких базах данных информацию о топологии сети, каналах, правилах, пользователях и службах. Эти данные могут преобразовываться с помощью одного или множества сценариев XSLT из нацеленных на задачи и независимых от производителя схем данных в формы для конкретного производителя, продукции, операционной системы и версии программ. Полученные в результате данные могут быть переданы устройству по протоколу NETCONF.

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с RFC 2119 [RFC2119].

1.1. Терминология

candidate configuration datastore - дополнительное хранилище конфигурации (кандидат)
: Конфигурационное хранилище, содержимое которого можно изменять без влияния на текущую конфигурацию устройства и впоследствии передать в рабочее хранилище. Такую возможность поддерживают не все устройства.

capability - дополнительная возможность
: Функциональность, дополняющая базовую спецификацию NETCONF.

client - клиент
: Инициирует (вызывает) протокольные операции на сервере. В дополнение к этому клиент может получать от сервера уведомления.

configuration data - данные конфигурации
: Набор данных, в которые возможна запись, для перевода системы из её начального состояния в текущее состояние конфигурации.

datastore - хранилище
: Концептуальное место хранения информации и доступа к ней. Хранилище может быть реализовано с использованием файлов, баз данных, флэш-памяти или их комбинации.

configuration datastore - хранилище конфигурации
: Хранилище данных, содержащее полный набор параметров конфигурации, требуемых для перевода устройства из начального состояния в желаемое рабочее состояние.

message - сообщение
: Протокольный объект, передаваемый через соединение. Сообщения используют формат XML.

notification - уведомление
: Инициированное сервером сообщение о том или ином событии на сервере.

protocol operation - протокольная операция
: Вызов указанной удалённой процедуры при использовании в протоколе NETCONF.

remote procedure call (RPC) - вызов удалённой процедуры
: Реализуется путём обмена сообщениями <rpc> и <rpc-reply>.

running configuration datastore - хранилище рабочей конфигурации
: Конфигурационное хранилище, содержащее конфигурацию, которая в настоящее время используется устройством. Рабочее хранилище существует всегда.

server - сервер
: Выполняет протокольные операции, инициированные клиентом. Кроме того, сервер может передавать клиенту уведомления.

session - сессия
: Клиент и сервер обмениваются сообщениями, используя защищённую, ориентированную на соединения сессию.

startup configuration datastore - хранилище стартовой конфигурации
: Конфигурационное хранилище, содержащее информацию, которую устройство использует при загрузке (включении). Присутствует только на устройствах с раздельными хранилищами для стартовой и рабочей конфигурации.

state data - данные состояния
: Дополнительные данные системы, которые не относятся к параметрам конфигурации (например, предназначенные только для чтения данные о состоянии и собранной статистике).

user - пользователь
: Подтвердивший свою подлинность клиент, который обычно идентифицируется именем пользователя NETCONF.

1.2. Обзор протокола

NETCONF использует простой механизм на основе RPC для организации взаимодействия между клиентом и сервером. Клиент может быть сценарием (script) или приложением и обычно является частью системы управления сетью. Сервером обычно выступает сетевое устройство. Термины «устройство» и «сервер» в этом документе имеют общее значение, равно как термины «клиент» и «приложение».

Сессия NETCONF представляет собой логическое соединение между сетевым администратором или приложением для управления сетью и сетевым устройством. Устройство **должно** поддерживать хотя бы одну сессию NETCONF и **следует** поддерживать множество сессий. Глобальные параметры конфигурации могут быть изменены в любой полномочной сессии и влияние этих изменений будет видно во всех сессиях. Сеансовые параметры оказывают влияние лишь на ту сессию, в которой они были изменены.

Протокол NETCONF можно концептуально разделить на четыре уровня, как показано на рисунке 1.

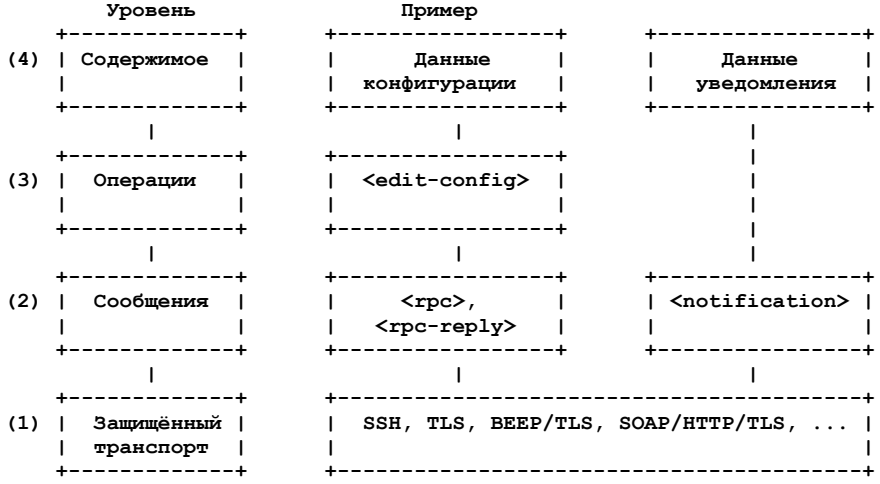

Рисунок 1. Уровни протокола NETCONF.

(1) **Защищённый транспорт** (Secure Transport) обеспечивает коммуникационный путь между клиентом и сервером. NETCONF может работать на основе любого транспортного протокола, соответствующего базовым требованиям, которые приведены в разделе 2.

(2) **Уровень сообщений** (Messages) обеспечивает простой и независимый от транспорта механизм кадрирования сообщений для представления RPC и уведомлений. Сообщения RPC описаны в разделе 4, а уведомления - в [RFC5717].

(3) **Уровень операций** (Operations) определяет набор базовых операций протокола, выполняемых методами RPC с представлением параметров в формате XML. Список базовых операций протокола приведён в разделе 7.

(4) **Уровень содержимого** (Content) выходит за рамки этого документа. Предполагается выполнение отдельной работы по стандартизации моделей данных NETCONF.

Язык моделирования данных YANG [RFC6020] был разработан для описания моделей данных и операций протокола NETCONF. Этот язык охватывает уровни (3) и (4) на рисунке 1.

1.3. Возможности

Возможности NETCONF (capability) представляют собой наборы функций, дополняющие базовую спецификацию NETCONF. Возможности указываются идентификаторами URI[1] [RFC3986].

Возможности дополняют базовые операции устройства, описывая дополнительные операции и разрешённое для этих операций содержимое. Клиент может определить возможности сервера и применять дополнительные операции, параметры и содержимое, определённые для этих возможностей.

Определение возможности может ссылаться на одну или несколько возможностей, от которых данная возможность зависит. Для поддержки данной возможности сервер **должен** поддерживать все возможности, от которых она зависит.

В разделе 8 определён обмен информацией о возможностях, который позволяет клиенту узнать возможности сервера. В этом же разделе приведён список возможностей, определённых в этом документе.

Дополнительные возможности могут определяться в других документах и это обеспечивает постоянное расширение возможностей. Органы стандартизации могут определять стандартизованные возможности, а производители - фирменные. Идентификаторы URI для возможностей **должны** чётко указывать орган именования для предотвращения путаницы.

1.4. Разделение данных конфигурации и состояния

Информация, которая может быть получена из работающей системы, делится на две категории - параметры конфигурации и данные состояния. Параметры конфигурации представляют собой набор данных с возможностью записи, которые обеспечивают переход системы из начального состояния в рабочее. Данные о состоянии системы не относятся к конфигурационным и обычно доступны только для чтения. Эти данные включают информацию о текущем состоянии устройства и собранную статистику. Если при настройке конфигурации системы используются данные состояния, возникают многочисленные проблемы:

- при сравнении данных будут доминировать не имеющие отношения к настройке величины типа статистики;
- входящая информация будет включать бессмысленные запросы типа попыток записи в поля, доступные лишь для чтения;
- наборы данных будут велики;
- архивные данные будут включать значения, доступные лишь для считывания, что усложнит обработку при восстановлении данных из архива.

Для решения упомянутых проблем протокол NETCONF различает параметры конфигурации и данные состояния, а также используемые для тех и других операции. Например, операция <get-config> возвращает только конфигурационные данные, а <get> - конфигурационные параметры и данные состояния.

Отметим, что протокол NETCONF концентрируется на информации, которая нужна для обеспечения желаемого рабочего состояния устройства. Включение других данных зависит от реализации протокола. Например, пользовательские файлы и базы данных протокол NETCONF не считает конфигурационными. Если же локальная база

[1]Uniform resource identifier - унифицированный идентификатор ресурса.

данных аутентификации пользователей сохраняется на устройстве, реализация протокола может включить её в конфигурационные данные.

2. Требования к транспортному протоколу

NETCONF использует коммуникационную парадигму на основе RPC. Клиент передаёт одно или множество сообщений с запросами RPC, которые приводят к генерации сервером соответствующих откликов RPC.

NETCONF может работать на основе любого транспортного протокола, обеспечивающего требуемую функциональность. Он не привязан к какому-то конкретному транспорту и позволяет задавать отображения для работы по разным транспортным протоколам.

Транспортный протокол **должен** обеспечивать уровню протокола NETCONF индикацию типа сессии (клиент или сервер).

Ниже более подробно описаны требования NETCONF к протоколу транспортного уровня.

2.1. Ориентированная на соединения работа

Протокол NETCONF относится к ориентированным на соединения и для работы ему требуется постоянная сессия между партнёрами. Соединение **должно** обеспечивать гарантированную доставку с сохранением порядка следования пакетов.

Кроме того, запрошенные для конкретного соединения ресурсы сервера **должен** автоматически освобождаться при разрыве соединения, что существенно упрощает восстановление при обрывах связи и повышает отказоустойчивость. Например, при организации соединения клиентом, оно сохраняется до явного завершения или до момента, когда сервер сочтёт, что соединение было прервано. При разрыве соединения в тот момент, когда клиент продолжает блокировать (использовать) его сервер может выполнить требуемые операции по восстановлению. Операция блокировки (<lock>) подробно описана в параграфе 7.5.

2.2. Проверка подлинности, целостность и конфиденциальность

Соединения NETCONF **должны** обеспечивать контроль подлинности, целостность данных, их защиту и предотвращение повторного использования (replay). Эти возможности NETCONF определяются транспортным протоколом. У партнёров NETCONF предполагается подобающий уровень защиты и конфиденциальности, независимо от данной спецификации. Например, соединения могут шифроваться с использованием защиты на транспортном уровне (TLS[1]) [RFC5246] или SSH[2] [RFC4251], в зависимости от базового протокола.

Соединения NETCONF **должны** аутентифицироваться. Транспортный протокол отвечает за проверку подлинности (аутентификацию) клиента на сервере и сервера у клиента. Партнёры NETCONF предполагают, что аутентификационные данные надлежащим образом проверены транспортным протоколом с использованием достаточно надёжных механизмов и отождествление партнёра должны образом проверено.

Одной из целей NETCONF является предоставление программного интерфейса с устройством, соответствующего естественному интерфейсу управления этим устройством. Поэтому предполагается, что базовый транспортный протокол использует доступные для устройства механизмы проверки подлинности. Например, сервер NETCONF на устройстве с поддержкой протокола RADIUS [RFC2865] может применять RADIUS для аутентификации сессий NETCONF.

Процесс аутентификации **должен** приводить к подтверждению подлинности отождествления клиента, известного серверу. Аутентифицированное отождествление клиента обычно называют именем пользователя NETCONF. Имя пользователя является строкой символов, соответствующих Char из параграфа 2.2 [W3C.REC-xml-20001006]. Алгоритм создания имён пользователей зависит от транспортного протокола и применяемого им механизма аутентификации. Транспортный протокол **должен** предоставлять имя пользователя для использования на других уровнях NETCONF.

Права доступа для данного клиента, указанного именем пользователя NETCONF, являются частью конфигурации сервера NETCONF. Эти права **должны** соблюдаться вплоть до завершения сессии NETCONF. Детали настройки прав доступа выходят за рамки этого документа.

2.3. Обязательный транспортный протокол

Реализации NETCONF **должны** поддерживать отображение транспортного протокола SSH [RFC6242].

3. Вопросы XML

XML служит форматом представления для NETCONF, позволяя задавать в форме текста сложные иерархические структуры, которые можно читать, сохранять и изменять с помощью текстовых редакторов и инструментов XML.

Все сообщения NETCONF **должны** использовать формат XML с кодировкой UTF-8 [RFC3629]. При получении сообщения <rpc> с некорректным форматом XML или отличающейся от UTF-8 кодировкой **следует** передавать в ответ сообщение об ошибке malformed-message. Если такое сообщение по какой-либо причине не может быть передано, сервер **должен** прервать сессию.

Сообщение NETCONF **может** начинаться с заголовка (декларации) XML (см. параграф 2.8 в [W3C.REC-xml-20001006]).

В этом разделе рассмотрены некоторые вопросы XML, связанные с NETCONF.

3.1. Пространство имён

Все элементы протокола NETCONF определены в пространстве имён

```
urn:ietf:params:xml:ns:netconf:base:1.0
```

[1]Transport Layer Security.
[2]Secure Shell - защищённая оболочка (среда).

Имена возможностей NETCONF **должны** быть идентификаторами URI [RFC3986]. Возможности NETCONF рассмотрены в разделе 8.

3.2. Декларации типа документов

Декларации типа документов (см. параграф 2.8 в [W3C.REC-xml-20001006]) **недопустимо** включать в содержимое NETCONF.

4. Модель RPC

Протокол NETCONF использует коммуникационную модель на основе RPC. Партнёры NETCONF применяют элементы <rpc> и <rpc-reply> для обеспечения независимого от транспортного протокола кадрирования запросов и откликов NETCONF.

Синтаксис представления XML для уровня сообщений (Messages) в RPC формально определён в схеме XML, представленной в Приложении B.

4.1. Элемент <rpc>

Элемент <rpc> служит для представления запросов NETCONF, передаваемых от клиентов к серверам.

Элемент <rpc> имеет обязательный атрибут message-id, являющийся строкой, выбранной отправителем RPC и обычно представляющей собой символьное представление монотонно возрастающих целых чисел. Получатель RPC не декодирует и не интерпретирует эту строку, лишь сохраняя её для последующего использования в качестве атрибута message-id в своём сообщении <rpc-reply>. Отправитель **должен** обеспечить нормализацию значений message-id в соответствии с правилами нормализации атрибутов, определёнными в [W3C.REC-xml-20001006], если хочет получить строку назад в неизменном виде. Например,

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <some-method>
    <!-- параметры метода -->
  </some-method>
</rpc>
```

При наличии в элементе <rpc> дополнительных атрибутов партнёр NETCONF **должен** возвращать эти атрибуты неизменными в элементе <rpc-reply>. Это относится ко всем атрибутам xmlns.

Имя и параметры RPC указываются в содержимом элемента <rpc>. Имя RPC является элементом, расположенным непосредственно в <rpc>, а все параметры представляются уже внутри этого элемента.

В приведённом ниже примере вызывается метод <my-own-method>, имеющий два параметра - <my-first-parameter> со значением 14 и <another-parameter> со значением fred.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <my-own-method xmlns="http://example.net/me/my-own/1.0">
    <my-first-parameter>14</my-first-parameter>
    <another-parameter>fred</another-parameter>
  </my-own-method>
</rpc>
```

В следующем примере вызывается метод <rock-the-house> с параметром <zip-code>, имеющим значение 27606-0100.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rock-the-house xmlns="http://example.net/rock/1.0">
    <zip-code>27606-0100</zip-code>
  </rock-the-house>
</rpc>
```

Третий пример вызывает метод NETCONF <get> без параметров.

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

4.2. Элемент <rpc-reply>

Сообщение <rpc-reply> передаётся в ответ на запрос <rpc>.

Элемент <rpc-reply> имеет обязательный атрибут message-id, который совпадает с атрибутом message-id из соответствующего запроса <rpc>.

Сервер NETCONF **должен** также возвращать без изменения в своём элементе <rpc-reply> все дополнительные атрибуты, включённые в элемент <rpc>.

Данные отклика представляются одним или множеством дочерних элементов внутри <rpc-reply>.

Представленный ниже элемент <rpc> вызывает метод NETCONF <get> и включает дополнительный атрибут user-id. Отметим, что атрибут user-id не относится к пространству имён NETCONF. Возвращаемый элемент <rpc-reply> включает атрибут user-id, а также все запрошенное содержимое.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ex="http://example.net/content/1.0"
     ex:user-id="fred">
  <get/>
</rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ex="http://example.net/content/1.0"
     ex:user-id="fred">
  <data>
    <!-- содержимое отклика ... -->
  </data>
</rpc-reply>
```

4.3. Элемент <rpc-error>

Элемент <rpc-error> передаётся в сообщениях <rpc-reply> при возникновении ошибки в процессе обработки <rpc>.

Если при обработке запроса <rpc> на сервере возникает множество ошибок, в отклик <rpc-reply> **может** включаться более одного элемента <rpc-error>. Однако от серверов в таких случаях не требуется обнаружение и представление более одного элемента <rpc-error>. От сервера не требуется проверки ошибок в каком-либо определённом порядке. Сервер лишь **должен** возвратить элемент <rpc-error> при возникновении любой ошибки в процессе обработки запроса.

Серверу **недопустимо** возвращать информацию об ошибках, относящуюся к прикладному уровню или модели данных, в элементе <rpc-error>, если клиент не имеет соответствующих прав доступа.

Элемент <rpc-error> включает перечисленную ниже информацию.

error-type

Определяет концептуальный уровень, на котором возникла ошибка и включает одно из перечисляемых значений:

- transport (уровень защищённого транспорта);
- rpc (уровень сообщений);
- protocol (уровень операций);
- application (уровень содержимого).

error-tag

Строка с кратким описанием ошибки. Разрешённые значения приведены в Приложении A.

error-severity

Строка, показывающая уровень важности ошибки в точки зрения устройства:

- error (ошибка);
- warning (предупреждение).

Отметим, что ни одно из значений <error-tag>, определённых в этом документе, не использует уровня warning, который является резервом на будущее.

error-app-tag

Строка с относящейся к модели данных или реализации информацией об ошибках (если они имеются). Этот элемент не применяется, если с конкретной ошибкой не связано подходящего кода error-tag. Если имеются теги для ошибок, связанных с моделью данных и реализацией сразу, сервер **должен** использовать значение, относящееся к модели данных.

error-path

Содержит абсолютное выражение XPath [W3C.REC-xpath-19991116], указывающее элемент пути к узлу, связанный с ошибкой, указанной в конкретном элементе <rpc-error>. Этот элемент не применяется при отсутствии элемента данных или узла хранения, связанного с конкретной ошибкой.

Выражение XPath интерпретируется в описанном ниже контексте:

- набор деклараций пространств имён в рамках элемента <rpc-error>;
- набор привязок переменных пуст;
- библиотекой является основная (core) библиотека функций.

Контекст узла зависит от связанного с ошибкой узла:

- если с ошибкой может быть связан элемент данных (payload), контекстом узла будет узел документа с запросом rpc (т. е. элемент <rpc>);
- в остальных случаях контекстом узла является общий корень всех моделей данных, т. е. узел, который имеет в качестве потомков узлы верхнего уровня всех моделей данных.

error-message

Содержит строку с описанием ошибки для представления человеку. Этот элемент не включается, если для соответствующей ошибки нет подходящего описания. В элемент **следует** включать атрибут xml:lang в соответствии с определением [W3C.REC-xml-20001006] и обсуждением в [RFC3470].

error-info

Содержит описание ошибки для конкретного протокола или модели данных. Этот элемент не включается, если для соответствующей ошибки нет подходящего описания. В списке Приложения A определены все обязательные значения error-info для каждой ошибки. После обязательного содержимого, диктуемого протоколом, определение модели данных **может** требовать включения некой информации уровня приложений в контейнер error-info. Реализация **может** включать дополнительные элементы для представления связанной с реализацией или расширенной информацией для отладки.

В Приложении A приведены стандартные ошибки NETCONF.

Например, ошибка возвращается, если элемент <rpc> получен без атрибута message-id. Отметим, что это единственный случай, когда узел NETCONF может опускать атрибут message-id в элементе <rpc-reply>.

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

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>rpc</error-type>
```

```
        <error-tag>missing-attribute</error-tag>
        <error-severity>error</error-severity>
        <error-info>
          <bad-attribute>message-id</bad-attribute>
          <bad-element>rpc</bad-element>
        </error-info>
      </rpc-error>
    </rpc-reply>
```

Приведённый ниже отклик <rpc-reply> иллюстрирует возврат множества элементов <rpc-error>. Отметим, что используемая в примерах модель данных применяет элемент <name> для обозначения экземпляров <interface>.

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /t:top/t:interface[t:name="Ethernet0/0"]/t:mtu
    </error-path>
    <error-message xml:lang="en">
      MTU value 25000 is not within range 256..9192
    </error-message>
  </rpc-error>
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /t:top/t:interface[t:name="Ethernet1/0"]/t:address/t:name
    </error-path>
    <error-message xml:lang="en">
      Invalid IP address for interface Ethernet1/0
    </error-message>
  </rpc-error>
</rpc-reply>
```

4.4. Элемент <ok>

Элемент <ok> передаётся в сообщениях <rpc-reply> при отсутствии ошибок в процессе выполнения запроса <rpc> и данных, возвращаемых из операции. Пример показан ниже.

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

4.5. Конвейер обработки

Запросы NETCONF <rpc> **должны** обрабатываться управляемым устройством последовательно. Дополнительные запросы <rpc> **могут** быть переданы до завершения обработки предшествующих. Управляемое устройство **должно** передавать отклики в соответствии с порядком получения запросов.

5. Модель конфигурации

Протокол NETCONF обеспечивает начальный набор операций и множество возможностей, которые могут служить для расширения базы. Партнёры NETCONF обмениваются информацией о возможностях при организации сессии, как описано в параграфе 8.1.

5.1. Хранилища конфигурации

NETCONF определяет наличие одного или нескольких хранилищ конфигурации и разрешает выполнять операции с ними. Хранилище конфигурации определяется как полный набор конфигурационных данных, которые нужны для перевода устройства из начального состояние в желаемое рабочее состояние. Конфигурационные хранилища не включают данных состояния и выполняемых команд.

Хранилище рабочей конфигурации содержит полный набор конфигурационных параметров, используемых устройством в данный момент. В устройстве всегда присутствует одно хранилище этого типа. Операции протокола NETCONF, относящиеся к этому хранилищу, используют элемент <running>.

В базовой модели присутствует только рабочее (<running>) хранилище конфигурации. Дополнительные хранилища **могут** определяться возможностями. Такие конфигурационные хранилища доступны лишь на устройствах, анонсирующих возможности.

Возможности из параграфов 8.3 и 8.7 определяют конфигурационные хранилища <candidate> и <startup>, соответственно.

5.2. Моделирование данных

Моделирование данных и вопросы содержимого выходят за рамки протокола NETCONF. Принимается допущение о том, что модель данных устройства известна приложению и обе стороны осведомлены о таких аспектах, как макет (схема), упаковка, закрепление, поиск, замена и управление данными, а также о любых ограничениях, накладываемых моделью данных.

NETCONF передаёт конфигурационные данные внутри элемента <config> для модели данных устройства. Протокол считает содержимое этого элемента непрозрачным. Устройство использует возможности для анонсирования набора реализованных моделей данных. Определение возможности детализирует работу и ограничения, вносимые моделью данных.

Устройства и менеджеры могут поддерживать множество моделей данных, включая стандартные и фирменные.

6. Фильтрация ветвей

6.1. Обзор

Фильтрация ветвей (subtree) XML представляет собой механизм, который позволяет приложению выбрать отдельные ветви XML для включения в <rpc-reply> операции <get> или <get-config>. Предоставляется небольшой набор фильтров для включения, совпадения содержимого и выбора, который позволяет создать полезные, но тоже весьма ограниченные механизмы выбора. Серверу не нужно использовать специфическую для модели данных семантику при обработке, что обеспечивает простую и централизованную стратегию реализации.

Концептуально фильтр subtree может быть пустым или содержать ветви элементов, которые представляют критерии выбора. На каждом уровне ветви набор «братских» (sibling) узлов логически обрабатывается сервером для определения включать ли ветви и элементы пути к корню в выходной результат фильтра.

Каждый узел, указанный в фильтре subtree, представляет собой включительный (inclusive) фильтр. Только связанные узлы в базовой модели или моделях данных внутри указанного хранилища на сервере будут выбраны фильтром. Узел выбирается, если он соответствует критерию выбора и иерархии элементов, указанных в данных фильтра, за исключением того, что абсолютное имя фильтра настроено на начало с уровня ниже <filter>.

Отклики содержат лишь ветви, выбранные фильтром. Любой критерий выбора, который присутствовал в запросе, включается и в отклик. Отметим, что некоторые элементы, выраженные в фильтре листьями, будут раскрыты (т. е. будут включены ветви) в выходной информации фильтра. Конкретные экземпляры данных не дублируются в отклике, если этот запрос содержит множество выражений фильтров ветвей, выбирающих одни и те же данные.

6.2. Компоненты фильтра subtree

Фильтр subtree состоит из элементов XML и их атрибутов XML. В фильтрах может применяться 5 типов компонент:

- выбор пространства имён;
- выражение для сопоставления атрибутов;
- узлы включения (containment);
- узлы выбора;
- узлы сопоставления содержимого.

6.2.1. Выбор пространства имён

Пространство имён считается соответствующим (в смысле фильтрации), если связанное с конкретным узлом пространство имён XML в элементе <filter> совпадает с пространством имён базовой (underlying) модели данных. Отметим, что выбор пространства имён не может применяться сам по себе. В фильтре **должен** быть задан хотя бы один элемент, если в результат фильтрации будут включаться любые элементы.

Для фильтра subtree определён механизм шаблонов (wildcard) пространства имён XML. Если элемент внутри <filter> не определяется пространством имён (например, xmlns=""), сервер **должен** оценить (evaluate) все поддерживаемые пространства имён XML, при обработке данного узла фильтра subtree. Этот механизм шаблонов не применим к атрибутам XML.

Отметим, что значения префиксов для подходящих (qualified) пространств имён не принимаются во внимание при сравнении элементов фильтра с элементами базовой модели данных.

Например,

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config"/>
</filter>
```

Элемент <top> является узлом выбора и только этот элемент пространств имён http://example.com/schema/1.2/config и любых дочерних узлов (из базовой модели данных) будет включён в выход фильтра.

6.2.2. Выражение для сопоставления атрибутов

Атрибуты в фильтре subtree являются частью выражения для сопоставления атрибутов. В любом типе узла фильтрации **может** присутствовать любое число (подходящих или не подходящих) атрибутов XML. В дополнение к критериям выбора, обычно применяемым к этому узлу, выбранные данные **должны** иметь соответствующие значения для каждого атрибута, заданного в узле. Если элемент не определён для включения указанного атрибута, он не будет выбран в результате фильтрации.

Например,

```
<filter type="subtree">
  <t:top xmlns:t="http://example.com/schema/1.2/config">
    <t:interfaces>
      <t:interface t:ifName="eth0"/>
    </t:interfaces>
  </t:top>
</filter>
```

В этом примере элементы <top> и <interfaces> являются узлами включения, элемент <interface> является узлом выбора, ifName является выражением для сопоставления атрибутов. Только узлы interface в пространстве имён http://example.com/schema/1.2/config, имеющие атрибут ifName со значением eth0 и находящиеся в узлах interfaces внутри узлов top, будут включены в выход фильтра.

6.2.3. Узлы включения

Узлы, которые содержат дочерние элементы в фильтре subtree, называются узлами включения (containment node). Каждый дочерний элемент может быть узлом любого типа, включая другой узел включения. Для каждого узла включения, заданного в фильтре subtree, все экземпляры модели данных, которые точно соответствуют пространствам имён, иерархии элементов и всем выражениям для сопоставления атрибутов, включаются в выход фильтра.

Например,

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
    <users/>
  </top>
</filter>
```

В этом примере элемент <top> является узлом включения.

6.2.4. Узлы выбора

Пустой узел leaf (лист) в фильтре называется узлом выбора (selection node) и представляет явный выбор фильтра в базовой модели данных. Присутствие каких-либо узлов выбора в наборе братских узлов будет приводить к выбору фильтром указанной ветви или ветвей и подавлению автоматического выбора всего набора братских узлов в базовой модели данных. Для целей фильтрации пустой узел leaf может быть заявлен либо с пустым тегом (например, <foo/>), либо с явными тегами начала и завершения (например, <foo> </foo>). Любые пробельные символы в этой форме игнорируются.

Например,

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
    <users/>
  </top>
</filter>
```

В этом примере <top> является узлом включения, а <users> - узлом выбора. Только узлы users в пространстве имён http://example.com/schema/1.2/config, которые находятся внутри элемента <top>, являющегося корнем хранилища конфигурации, будут включены в результат фильтрации.

6.2.5. Узлы сопоставления содержимого

Лист (leaf) с простым содержимым называется узлом сопоставления содержимого (content match node). Этот узел используется для выбора некоторых или всех его братских узлов в результат фильтра и представляет собой фильтр точного соответствия (совпадения) содержимого элемента leaf. Ниже перечислены ограничения, применяемые к узлам сопоставления содержимого.

- Узлу сопоставления **недопустимо** включать вложенные элементы.
- Множество узлов сопоставления (т. е., братских узлов) логически связывается операцией AND.
- Фильтрация смешанного содержимого не поддерживается.
- Фильтрация содержимого списков не поддерживается.
- Фильтрация содержимого, включающего только пробельные символы (whitespace), не поддерживается.
- Узел сопоставления **должен** включать непробельные символы. Пустой элемент (например, <foo></foo>) будет считаться узлом выбора (например, <foo/>).
- Пробельные символы в начале и в конце игнорируются, но эти же символы внутри блока текстовых символов учитываются и не меняются.

Если содержимое всех указанных братских узлов в фильтре subtree даёт результат true, узлы для выхода фильтра выбираются в соответствии с приведёнными ниже условиями.

- Каждый узел сопоставления содержимого из набора братских узлов включается в выход фильтра.
- Если в братском наборе присутствуют узлы включения, они обрабатываются дальше и включаются в выходной результат фильтра при выполнении вложенных условий.
- Если в братском наборе присутствуют узлы выбора, эти узлы включаются в выходной результат фильтра.
- Если какие-либо братские узлы узла выбора являются компонентами идентификатора экземпляра для концептуальной структуры данных (например, лист ключа списка), они **могут** включаться в выход фильтра.
- В остальных случаях (т. е. при отсутствии узлов включения и выбора в братском наборе) все узлы, определённые на этом уровне в базовой модели данных (и их ветви, если они есть), возвращаются в выходном результате фильтра.

Если любой из братских узлов сопоставления даёт результат false, дальнейшая обработка этого братского набора не выполняется и ни один из узлов набора (включая узлы сопоставления содержимого) не выбирается фильтром.

Например,

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
```

```
        <users>
          <user>
            <name>fred</name>
          </user>
        </users>
      </top>
    </filter>
```

В этом примере узлы <users> и <user> являются узлами включения, а <name> - узлом сопоставления содержимого. Поскольку не указаны братские узлы для <name> (и, следовательно, нет узлов включения и выбора), все «братья» узла <name> возвращаются на выходе фильтра. Только узлы user в пространстве имён http://example.com/schema/1.2/config, которые соответствуют иерархии элементов и для которых элемент <name> имеет значение fred, будут включены в выходной результат фильтра.

6.3. Обработка фильтра subtree

Выход фильтра (набор выбранных узлов) изначально пуст.

Каждый фильтр subtree может содержать один или множество фрагментов модели данных, представляющих часть модели, которая будет выбрана (со всеми дочерними узлами) на выходе фильтра.

Каждый фрагмент данных субдерева сравнивается сервером с поддерживаемыми им моделями данных. Если фрагмент данных фильтра (от корня до самого внутреннего элемента, указанного в фильтре) совпадает с соответствующей частью поддерживаемой модели данных, тогда узел и все его потомки будут включены в результат фильтрации.

Сервер обрабатывает все узлы с общим предком (братский набор) совместно, начиная от корня в направлении листьев. Корневые элементы фильтра считаются входящими в один братский набор (для них предполагается общее пространство имён), хотя они и не имеют общего предка.

Для каждого братского набора сервер определяет узлы для включения (или возможного включения) в результат и братские субдеревья для исключения из результата фильтрации. Сначала сервер определяет типы присутствующих в братском наборе узлов и обрабатывает эти узлы в соответствии с правилами для их типа. Если какие-то узлы из братского набора выбраны, процесс рекурсивно применяется к братским наборам каждого выбранного узла. Алгоритм продолжается, пока не будут обработаны братские наборы во всех субдеревьях, заданных фильтром.

6.4. Примеры фильтрации subtree

6.4.1. Нет фильтра

При отказе от фильтрации операция <get> возвращает модель данных целиком.

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

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <!-- ... возвращается весь набор данных ... -->
  </data>
</rpc-reply>
```

6.4.2. Пустой фильтр

Пустой фильтр не выберет ничего, поскольку нет совпадений содержимого или узлов выбора. Это не будет ошибкой. Атрибут type элемента, использованный в этих примерах, рассмотрен в параграфе 7.1.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
    </filter>
  </get>
</rpc>
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
  </data>
</rpc-reply>
```

6.4.3. Выбор всей ветви <users>

Фильтр в этом примере включает один узел выбора (<users>), поэтому он выберет лишь субдерево пользователей. Этот пример представляет заполненную модель данных <users> в большинстве из приведённых ниже примеров фильтров. В реальном мире <company-info> вряд ли будет возвращаться со списком пользователей конкретного хоста или сети.

Примечание. Примеры фильтрации и конфигурации в этом документе даны в пространстве имён http://example.com/schema/1.2/config. Корневым элементом этого пространства является <top>. Элемент <top> и его потомки представляют в примере только модель конфигурационных данных.

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

```
      </source>
      <filter type="subtree">
        <top xmlns="http://example.com/schema/1.2/config">
          <users/>
        </top>
      </filter>
    </get-config>
  </rpc>

  <rpc-reply message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
            <type>superuser</type>
            <full-name>Charlie Root</full-name>
            <company-info>
              <dept>1</dept>
              <id>1</id>
            </company-info>
          </user>
          <user>
            <name>fred</name>
            <type>admin</type>
            <full-name>Fred Flintstone</full-name>
            <company-info>
              <dept>2</dept>
              <id>2</id>
            </company-info>
          </user>
          <user>
            <name>barney</name>
            <type>admin</type>
            <full-name>Barney Rubble</full-name>
            <company-info>
              <dept>2</dept>
              <id>3</id>
            </company-info>
          </user>
        </users>
      </top>
    </data>
  </rpc-reply>
```

Следующий пример дал бы такой же результат для контейнера <users> с одним дочерним элементом (<user>).

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
      <source>
        <running/>
      </source>
      <filter type="subtree">
        <top xmlns="http://example.com/schema/1.2/config">
          <users>
            <user/>
          </users>
        </top>
      </filter>
    </get-config>
  </rpc>
```

6.4.4. Выбор всех элементов <name> в ветви <users>

Этот фильтр задаёт два узла включения (<users>, <user>) и один узел выбора (<name>). Все экземпляры элемента <name> в одном братском наборе выбираются в результате фильтрации. Клиенту может потребоваться знать, что <name> используется в этой конкретной структуре данных в качестве идентификатора экземпляра, но серверу не эти метаданные не нужны для обработки запроса.

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
      <source>
        <running/>
      </source>
      <filter type="subtree">
        <top xmlns="http://example.com/schema/1.2/config">
          <users>
            <user>
              <name/>
            </user>
          </users>
        </top>
      </filter>
    </get-config>
  </rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
        </user>
        <user>
          <name>fred</name>
        </user>
        <user>
          <name>barney</name>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

6.4.5. Запись для конкретного <user>

Этот фильтр задаёт два узла включения (<users>, <user>) и один узел сопоставления содержимого (<name>). Все экземпляры братского набора, содержащие <name> со значением fred, будут включены в результат.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>fred</name>
          </user>
        </users>
      </top>
    </filter>
  </get-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>fred</name>
          <type>admin</type>
          <full-name>Fred Flintstone</full-name>
          <company-info>
            <dept>2</dept>
            <id>2</id>
          </company-info>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

6.4.6. Отдельные элементы конкретной записи <user>

Этот фильтр задаёт два узла включения (<users>, <user>), один узел сопоставления содержимого (<name>) и два узла выбора (<type>, <full-name>). Все экземпляры элементов <type> и <full-name> из братского набора, содержащие элемент <name> со значением fred, будут включены в результат фильтра. Элемент <company-info> не будет включён, поскольку братский набор содержит узлы выбора.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>fred</name>
            <type/>
            <full-name/>
          </user>
        </users>
      </top>
    </filter>
  </get-config>
</rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>fred</name>
          <type>admin</type>
          <full-name>Fred Flintstone</full-name>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

6.4.7. Множество ветвей

Этот фильтр содержит три ветви (name=root, fred, barney).

Фильтр ветви root содержит два узла включения (<users>, <user>), один узел сопоставления содержимого (<name>) и один узел выбора (<company-info>). Критерии фильтрации выполняются и выходной результат содержит ветвь company-info дерева root.

Фильтр ветви fred содержит три узла включения (<users>, <user>, <company-info>), один узел сопоставления содержимого (<name>) и один узел выбора (<id>). Критерии фильтрации выполняются и результат будет включает <id> в ветви company-info для пользователя fred.

Фильтр ветви barney содержит три узла включения (<users>, <user>, <company-info>), два узла сопоставления содержимого (<name>, <type>) и один узел выбора (<dept>). Критерии фильтра не выполняются, поскольку пользователь barney не относится к типу superuser и вся ветвь пользователя barney (включая родительский элемент <user>) исключается из финального результата.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
            <company-info/>
          </user>
          <user>
            <name>fred</name>
            <company-info>
              <id/>
            </company-info>
          </user>
          <user>
            <name>barney</name>
            <type>superuser</type>
            <company-info>
              <dept/>
            </company-info>
          </user>
        </users>
      </top>
    </filter>
  </get-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <user>
          <name>fred</name>
          <company-info>
            <id>2</id>
          </company-info>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

6.4.8. Элементы с именованием атрибутов

В этом примере фильтр содержит один узел включения (<interfaces>), одно выражение для сопоставления атрибутов ("ifName") и один узел выбора (<interface>). Все экземпляры ветви <interface>, имеющие атрибут ifName со значением eth0, будут включены в результат. Элементы данных фильтра и атрибуты пригодны, поскольку непригодные атрибуты ifName не будут частью пространства имён schema/1.2.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <t:top xmlns:t="http://example.com/schema/1.2/stats">
        <t:interfaces>
          <t:interface t:ifName="eth0"/>
        </t:interfaces>
      </t:top>
    </filter>
  </get>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <t:top xmlns:t="http://example.com/schema/1.2/stats">
      <t:interfaces>
        <t:interface t:ifName="eth0">
          <t:ifInOctets>45621</t:ifInOctets>
          <t:ifOutOctets>774344</t:ifOutOctets>
        </t:interface>
      </t:interfaces>
    </t:top>
  </data>
</rpc-reply>
```

Если ifName будет дочерним узлом, а не атрибутом, приведённый ниже запрос даст похожий результат.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/stats">
        <interfaces>
          <interface>
            <ifName>eth0</ifName>
          </interface>
        </interfaces>
      </top>
    </filter>
  </get>
</rpc>
```

7. Операции протокола

Протокол NETCONF обеспечивает небольшой набор операций нижнего уровня для управления конфигурацией устройства и получения данных состояния. Базовый протокол обеспечивает операции для извлечения, настройки, копирования и удаления хранилищ конфигурации. Дополнительные операции могут обеспечиваться с помощью анонсируемых устройством возможностей.

Базовый протокол поддерживает следующие операции:

- get;
- get-config;
- edit-config;
- copy-config;
- delete-config;
- lock;
- unlock;
- close-session;
- kill-session.

Протокольная операция может сталкиваться с отказами по разным причинам, включая отсутствие поддержки операции (operation not supported). Инициатору **не следует** надеяться, что любая операция всегда будет успешной. Возвращаемые любым откликом RPC значения **следует** проверять на предмет наличия ошибок.

Синтаксис и представление XML для протокольных операций формально определены в модуле YANG (Приложение C). В следующих параграфах описана семантика каждой протокольной операции.

7.1. <get-config>

Описание

Извлекает содержимое конфигурационного хранилища целиком или частично.

Параметры

source

Имя конфигурационного хранилища, из которого запрашиваются данные (например, <running/>).

filter

Этот параметр задаёт извлекаемые части конфигурационного хранилища. При отсутствии параметра извлекаются все данные.

Элемент <filter> **может** включать атрибут type, показывающий тип фильтрации в элементе <filter>. Используемый по умолчанию в NETCONF механизм фильтрации называется фильтром ветвей (subtree) и описан в разделе 6. Значение subtree явно указывает этот тип фильтрации.

Если партнёр NETCONF поддерживает возможность :xpath (параграф 8.9), **можно** использовать xpath для индикации того, что атрибут select в элементе <filter> содержит выражение XPath.

Позитивный отклик

Если устройство может выполнить запрос, сервер передаёт <rpc-reply>, содержащий элемент <data> с результатами запроса.

Негативный отклик

Если по какой-либо причине запрос не может быть выполнен полностью в отклик <rpc-reply> включается элемент <rpc-error>.

Пример

Для извлечения всей ветви <users> можно использовать приведённый ниже запрос.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </filter>
  </get-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <type>superuser</type>
          <full-name>Charlie Root</full-name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <!-- здесь могут быть дополнительные элементы <user> -->
      </users>
    </top>
  </data>
</rpc-reply>
```

В разделе 6 приведены дополнительные примеры фильтрации ветвей.

7.2. <edit-config>

Описание

Операция <edit-config> загружает указанную конфигурацию целиком или частично в заданное хранилище данных конфигурации. Эта операция позволяет выразить новую конфигурацию разными способами, такими как запись в локальный или удалённый файл, а также в устройство (inline). Если целевое хранилище отсутствует, оно будет создано.

Если партнёр NETCONF поддерживает возможность :url (параграф 8.8), вместо параметра <config> может указываться элемент <url>.

Устройство анализирует исходную и целевую конфигурацию и вносит запрошенные изменения. Целевая конфигурация не обязательно заменяется, как в случае <copy-config>. Вместо замены она просто изменяется в соответствии с исходной конфигурацией и запрошенными действиями.

Если операция <edit-config> включает множество субопераций, применяемых к одному и тому же концептуальному узлу базовой модели данных, результат операции становится неопределённым (т. е. выходит за рамки протокола NETCONF).

Атрибуты

operation

Элементы ветви <config> **могут** включать атрибут operation, который относится к пространству имён NETCONF, определённому в параграфе 3.1. Атрибут указывает точку в конфигурации, к которой операция применяется, и **может** встречаться во множестве элементов ветви <config>.

Если атрибут operation не задан, конфигурация сливается (merge) с конфигурационным хранилищем.

Атрибут operation может принимать одно из приведённых ниже значений.

merge

Конфигурационные данные, указанные элементом, который содержит этот атрибут, объединяются с конфигурацией на соответствующем уровне в хранилище, заданном параметром <target>. Этот вариант применяется по умолчанию.

replace
Конфигурационные данные, указанные элементом, который содержит этот атрибут, заменяют соответствующую конфигурацию в хранилище, заданном параметром <target>. Если таких данных нет в хранилище, они просто создаются. В отличие от операции <copy-config>, заменяющей целевую конфигурацию целиком, здесь заменяется только часть, указанная параметром <config>.

create
Конфигурационные данные, указанные элементом, который содержит этот атрибут, добавляются в конфигурацию тогда и только тогда, когда их нет в конфигурационном хранилище. Если такие данные имеются, возвращается элемент <rpc-error> с тегом ошибки (<error-tag>) data-exists.

delete
Конфигурационные данные, указанные элементом, который содержит этот атрибут, удаляются из хранилища тогда и только тогда, когда они там имеются. Если таких данных нет, возвращается элемент <rpc-error> с тегом ошибки с тегом ошибки data-missing.

remove
Конфигурационные данные, указанные элементом, который содержит этот атрибут, удаляются из хранилища, если они там имеются. Если таких данных нет, сервер просто игнорирует операцию.

Параметры

target
Имя конфигурационного хранилища, которое будет изменено (например, <running/> или <candidate/>).

default-operation
Задаёт принятую по умолчанию операцию (см. описание атрибута operation выше) для данного запроса <edit-config>. По умолчанию для параметра <default-operation> используется значение merge.
Параметр <default-operation> является необязательным и может принимать одно из приведённых значений.

merge
Конфигурационные данные, заданные параметром <config>, объединяются с конфигурацией на соответствующем уровне в целевом хранилище. Этот вариант применяется по умолчанию.

replace
Конфигурационные данные, заданные параметром <config>, полностью заменяют конфигурацию в целевом хранилище. Это полезно для загрузки сохранённых ранее данных конфигурации.

none
Заданная параметром <config> конфигурация не оказывает влияния на целевое хранилище, пока новые конфигурационные данные не включают атрибут operation для запроса другой операции. Если конфигурация в параметре <config> содержит данные, для которых в целевом хранилище нет соответствующего уровня, возвращается <rpc-error> с тегом ошибки data-missing. Использование none позволяет операциям типа delete избегать создания родительской иерархии для элемента, который будет удалён.

test-option
Элемент <test-option> **может** быть задан лишь в тех случаях, когда устройство анонсирует возможность :validate:1.1 (параграф 8.6).
Элемент <test-option> может принимать одно из приведённых ниже значений.

test-then-set
Перед установкой конфигурации выполняется проверка её пригодности. Если проверка завершилась ошибкой, операция <edit-config> не применяется. Этот вариант проверки применяется по умолчанию.

set
Конфигурация устанавливается без предварительной проверки её пригодности.

test-only
Выполняется только проверка пригодности без попытки установить конфигурацию.

error-option
Элемент <error-option> использует одно из приведённых ниже значений.

stop-on-error
Операция <edit-config> прерывается при первой ошибке. Этот вариант применяется по умолчанию.

continue-on-error
При возникновении ошибки обработка конфигурации продолжается с записью информации об ошибке и возвратом негативного отклика.

rollback-on-error
При возникновении ошибки, которая требует генерации элемента <rpc-error>, сервер прекращает обработку операции <edit-config> и восстанавливает состояние, которое было до начала операции. Опция требует поддержки на сервере возможности :rollback-on-error, описанной в параграфе 8.5.

config
Иерархия данных конфигурации, определённых одной из моделей данных устройства. Содержимое **должно** помещаться в подходящее пространство имён, чтобы устройство могло определить нужную модель данных, а также **должны** соблюдаться ограничения этой модели данных в соответствии с определением возможности (см. раздел 8).

Позитивный отклик
Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик
Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример
Примеры операции <edit-config> в этом параграфе используют простую модель данных, в которой может присутствовать множество экземпляров <interface>, различающихся элементами <name>.
Установка MTU 1500 на интерфейсе Ethernet0/0 в рабочей конфигурации может иметь вид

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

```
          <top xmlns="http://example.com/schema/1.2/config">
            <interface>
              <name>Ethernet0/0</name>
              <mtu>1500</mtu>
            </interface>
          </top>
        </config>
      </edit-config>
    </rpc>
    <rpc-reply message-id="101"
         xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
    </rpc-reply>
```

Добавим интерфейс Ethernet0/0 в рабочую конфигурацию, заменив имеющийся интерфейс с таким именем.

```
    <rpc message-id="101"
         xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <edit-config>
        <target>
          <running/>
        </target>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
          <top xmlns="http://example.com/schema/1.2/config">
            <interface xc:operation="replace">
              <name>Ethernet0/0</name>
              <mtu>1500</mtu>
              <address>
                <name>192.0.2.4</name>
                <prefix-length>24</prefix-length>
              </address>
            </interface>
          </top>
        </config>
      </edit-config>
    </rpc>

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

Удалим интерфейс Ethernet0/0 из рабочей конфигурации.

```
    <rpc message-id="101"
         xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <edit-config>
        <target>
          <running/>
        </target>
        <default-operation>none</default-operation>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
          <top xmlns="http://example.com/schema/1.2/config">
            <interface xc:operation="delete">
              <name>Ethernet0/0</name>
            </interface>
          </top>
        </config>
      </edit-config>
    </rpc>

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

Удалим интерфейс 192.0.2.4 из области OSPF (другие интерфейсы этой области не изменяются).

```
    <rpc message-id="101"
         xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <edit-config>
        <target>
          <running/>
        </target>
        <default-operation>none</default-operation>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
          <top xmlns="http://example.com/schema/1.2/config">
            <protocols>
              <ospf>
                <area>
                  <name>0.0.0.0</name>
                  <interfaces>
                    <interface xc:operation="delete">
                      <name>192.0.2.4</name>
                    </interface>
                  </interfaces>
                </area>
              </ospf>
            </protocols>
```

```
        </top>
      </config>
    </edit-config>
  </rpc>

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

7.3. <copy-config>

Описание

Создаёт или заменяет хранилище данных целиком содержимым другого полного хранилища конфигурационных данных. Если целевое хранилище существует, оно переписывается, в противном случае создаётся новое, когда это разрешено.

Если партнёр NETCONF поддерживает свойство :url (параграф 8.8), элемент <url> может присутствовать в качестве параметра <source> или <target>. Даже в тех случаях, когда устройство анонсирует поддержку возможности :writable-running, **оно** может отказаться от поддержки хранилища <running/> в качестве цели (<target>) операции <copy-config>. Устройство **может** отказаться от поддержки операций копирования одной удалённой конфигурации в другую удалённую конфигурацию, когда оба параметра <source> и <target> используют элемент <url>. Если <source> и <target> указывают один идентификатор URL или хранилище конфигурации, **должна** возвращаться ошибка с тегом, содержащим invalid-value.

Параметры

target

Имя конфигурационного хранилища, в которое операция <copy-config> будет записывать данные.

source

Имя конфигурационного хранилища, служащего источником данных для операции <copy-config>, или элемент <config>, содержащий полную конфигурацию для копирования.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <copy-config>
      <target>
        <running/>
      </target>
      <source>
        <url>https://user:password@example.com/cfg/new.txt</url>
      </source>
    </copy-config>
  </rpc>
  <rpc-reply message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
  </rpc-reply>
```

7.4. <delete-config>

Описание

Удаляет конфигурационное хранилище. Хранилище рабочей конфигурации <running> не может быть удалено.

Если партнёр NETCONF поддерживает возможность :url (параграф 8.8), в качестве параметра <target> может присутствовать элемент <url>.

Параметры

target

Имя удаляемого хранилища конфигурации.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <delete-config>
      <target>
        <startup/>
      </target>
    </delete-config>
  </rpc>

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

7.5. <lock>

Описание

Операция <lock> позволяет клиенту заблокировать конфигурационное хранилище на устройстве. Такие блокировки рассчитаны на короткое время и позволяют клиенту внести изменения, не опасаясь конфликтов с другими

клиентами NETCONF или иных протоколов (например, SNMP и сценарии командного интерфейса CLI), а также с действиями операторов.

Попытка блокировки конфигурационного хранилища **должна** приводить к отказу, если существующая сессия или другой объект уже блокируют какую-либо часть этого хранилища.

При активной блокировке сервер **должен** предотвращать любые изменения заблокированных ресурсов, кроме исходящих из данной сессии. Запросы SNMP и CLI на изменение ресурса **должны** завершаться отказом с возвратом подходящей ошибки.

Блокировка продолжается с момента её получения до освобождения блокировки или завершения сессии NETCONF. Сессия может быть явно закрыта клиентом или неявно прекращена севером в результате отказа базового транспорта, тайм-аута бездействия, недопустимого поведения на стороне клиента. Критерии разрыва сессии зависят от реализации и базового транспорта.

Операция <lock> принимает обязательный параметр <target>, который указывает имя блокируемого конфигурационного хранилища. При активной блокировке использование заблокированного хранилища в качестве цели операции <copy-config> будет запрещено и для любой другой сессии NETCONF. Кроме того, система будет предотвращать изменение заблокированных ресурсов другими операциями управления (например, SNMP или CLI). Для снятия блокировки, активизированной в другой сессии NETCONF, можно воспользоваться командой <kill-session>. Снятие блокировок, заданных другими средствами, выходит за рамки этого документа.

Блокировку **недопустимо** предоставлять при наличии любого из приведённых ниже условий.

- Блокировка уже активизирована любой сессией NETCONF или другим объектом.
- Целевая конфигурация имеет тип <candidate>, уже изменена и эти изменения не были представлены или отменены (rolled back).
- Целевая конфигурация имеет тип <running> и другая сессия NETCONF уже представила изменения (параграф 8.4).

Сервер **должен** отвечать на запрос блокировки элементом <ok> или откликом об ошибке <rpc-error>.

Блокировка будет быть снята системой при разрыве установившей блокировку сессии по любой причине.

Параметры

target

Имя конфигурационного хранилища для блокировки.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Если блокировка уже установлена, элемент <error-tag> будет иметь значение lock-denied, <error-info> будет включать <session-id> для держателя блокировки. Если блокировка задана объектом, не относящимся к NETCONF, <session-id> будет иметь значение 0. Отметим, что даже частичная блокировка целевого хранилища другим объектом будет препятствовать (глобальной) блокировке NETCONF для этого хранилища.

Пример

Приведённый ниже пример показывает успешную блокировку.

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

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

Следующий пример показывает отказ блокировки по причине того, что хранилище уже заблокировано.

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

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error> <!-- отказ блокировки -->
    <error-type>protocol</error-type>
    <error-tag>lock-denied</error-tag>
    <error-severity>error</error-severity>
    <error-message>
      Lock failed, lock is already held
    </error-message>
    <error-info>
      <session-id>454</session-id>
      <!-- блокировка задана сессией NETCONF с номером 454 -->
    </error-info>
  </rpc-error>
</rpc-reply>
```

7.6. <unlock>

Описание

Операция <unlock> служит для снятия блокировки конфигурации, заданной ранее операцией <lock>.

Операция <unlock> не снимет блокировку при наличии любого из указанных условий:

- указанная блокировка в данных момент не активна;
- сессия, где задана операция <unlock>, отличается от установившей блокировку сессии.

Сервер **должен** отвечать на запрос снятия блокировки элементом <ok> или откликом об ошибке <rpc-error>.

Параметры

target

Имя конфигурационного хранилища для снятия блокировки.

Клиент NETCONF не может снимать блокировку конфигурационного хранилища, которую он не устанавливал.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

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

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

7.7. <get>

Описание

Извлекает рабочую конфигурацию и данные о состоянии устройства.

Параметры

filter

Этот параметр задаёт часть конфигурации системы и данных состояния для извлечения информации. При отсутствии параметра возвращается вся конфигурация и данные состояния.

Элемент <filter> **может** включать атрибут type, показывающий тип фильтрации. Используемый по умолчанию в NETCONF механизм фильтрации называется фильтром ветвей (subtree) и описан в разделе 6. Значение subtree явно указывает этот тип фильтрации.

Если партнёр NETCONF поддерживает возможность :xpath (параграф 8.9), **можно** использовать xpath для индикации того, что атрибут select в элементе <filter> содержит выражение XPath.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>. Раздел <data> содержит запрошенные данные.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/stats">
        <interfaces>
          <interface>
            <ifName>eth0</ifName>
          </interface>
        </interfaces>
      </top>
    </filter>
  </get>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/stats">
      <interfaces>
        <interface>
          <ifName>eth0</ifName>
          <ifInOctets>45621</ifInOctets>
          <ifOutOctets>774344</ifOutOctets>
        </interface>
      </interfaces>
    </top>
  </data>
</rpc-reply>
```

7.8. <close-session>

Описание

Запрашивает аккуратное прерывание сессии NETCONF.

Сервер NETCONF при получении запроса <close-session> будет аккуратно завершать сессию, освобождая все блокировки и ресурсы, связанные с сессией, а также закрывая все связанные с ней соединения. Любые запросы NETCONF, полученные после <close-session>, будут игнорироваться.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

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

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

7.9. <kill-session>

Описание

Принудительное завершение сессии NETCONF.

При получении узлом NETCONF запроса <kill-session> он будет прерывать все обрабатываемые операции, освобождать все связанные с сессией блокировки и ресурсы, а также закрывать связанные с ней соединения.

Если сервер NETCONF получает запрос <kill-session> в процессе выполнения подтверждаемого представления конфигурации (параграф 8.4), он **должен** восстановить конфигурацию, которая была до начала представления.

В остальных случаях операция <kill-session> не обеспечивает возврата к прежней конфигурации или отмены других изменений, внесённых задавшим блокировку объектом.

Параметры

session-id

Идентификатор прерываемой сессии NETCONF. Если это значение не совпадает с идентификатором текущей сессии, возвращается ошибка invalid-value.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>4</session-id>
  </kill-session>
</rpc>

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

8. Возможности

В этом разделе определяется набор возможностей, которые **могут** применять клиент и сервер. Каждый партнёр анонсирует свои возможности, указывая их в процессе начального согласования. От каждого партнёра требуется понимание лишь тех возможностей, которые он может применять, и он **должен** игнорировать полученные от другого партнёра возможности, которые ему не требуются или непонятны.

Дополнительные возможности могут определяться с использованием шаблона, описанного в Приложении D. Будущие определения возможностей могут публиковаться органами стандартизации или в виде фирменных расширений.

Возможности NETCONF указываются идентификаторами URI. Базовые возможности определяются с использованием идентификаторов URN в соответствии с методом, описанным в RFC 3553 [RFC3553]. Определённые в этом документе возможности используют формат

```
urn:ietf:params:netconf:capability:{name}:1.x
```

где {name} указывает имя возможности. В обсуждениях и почтовых сообщениях возможности часто указываются в сокращённом виде :{name} или :{name}:{version}, если существует несколько версий возможности. Например, возможность foo будет иметь формальное имя urn:ietf:params:netconf:capability:foo:1.0, и скрашенное :foo. Сокращённую форму **недопустимо** применять в протоколе.

8.1. Обмен данными о возможностях

Возможности анонсируются в сообщениях, переданных каждым партнёром в процессе организации сессии. Когда сеанс NETCONF организован, каждый партнёр (и клиент, и сервер) **должен** передать элемент <hello> содержащий список поддерживаемых возможностей. Каждый из партнёров **должен** анонсировать по меньшей мере базовую возможность NETCONF urn:ietf:params:netconf:base:1.1. Партнёры **могут** включать возможности из предыдущих версий NETCONF для индикации поддержки разных версий протокола.

Партнёры **должны** убедиться в наличии поддерживаемой обоими версии протокола. При сравнении URI версии протокола используется только компонента base, если в конце строки возможностей представлены дополнительные параметры. Если общей версии не найдено, партнёрам NETCONF **недопустимо** продолжать сессию. Если в URI указано несколько версий протокола, **должна** выбираться старшая (наиболее свежая) версия, поддерживаемая обоими партнёрами.

Сервер, передающий элемент <hello>, **должен** включать в него элемент <session-id> с идентификатором данной сессии NETCONF. Клиенту при передаче элемента <hello> **недопустимо** включать в него элемент <session-id>.

Сервер, получивший сообщение <hello> с элементом <session-id>, **должен** прервать сессию NETCONF. Аналогично, клиент, не получивший элемент <session-id> в сообщении <hello> от сервера, **должен** прервать сессию NETCONF (без отправки <close-session>).

В приведённом ниже примере сервер анонсирует базовые возможности NETCONF, одну дополнительную возможность из базовой спецификации NETCONF и одну дополнительную возможность данного сервера.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.1
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:startup:1.0
    </capability>
    <capability>
      http://example.net/router/2.3/myfeature
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

Каждый из партнёров передаёт сообщение <hello> сразу после организации соединения. Партнёрам **недопустимо** ждать приветствия от другой стороны перед отправкой своего.

8.2. Возможность записи в рабочую конфигурацию

8.2.1. Описание

Возможность :writable-running говорит о том, что устройство поддерживает непосредственную запись в хранилище конфигурации <running>. Иными словами, устройство поддерживает операции <edit-config> и <copy-config> с целевым хранилищем <running>.

8.2.2. Зависимости

Нет.

8.2.3. Идентификатор возможности

Возможность :writable-running указывается строкой

```
urn:ietf:params:netconf:capability:writable-running:1.0
```

8.2.4. Новые операции

Нет.

8.2.5. Изменения существующих операций

8.2.5.1. <edit-config>

Возможность :writable-running меняет операцию <edit-config>, позволяя указывать <running> в качестве <target>.

8.2.5.2. <copy-config>

Возможность :writable-running меняет операцию <copy-config>, позволяя указывать <running> в качестве <target>.

8.3. Поддержка конфигурации-кандидата

8.3.1. Описание

Возможность :candidate указывает поддержку устройством дополнительного хранилища конфигурации, служащего для хранения конфигурационных данных, которыми можно манипулировать без влияния на текущую конфигурацию устройства. Хранилище-кандидат содержит полный набор конфигурационных данных и позволяет манипулировать ими. Операции добавления, удаления или изменения конфигурационных данных позволяют создать желаемую конфигурацию. Операция <commit> **может** быть вызвана в любой момент для подачи этих данных в рабочую конфигурацию устройства.

Операция <commit> представляет текущее содержимое конфигурации-кандидата в рабочую конфигурацию устройства. Хотя это представление выглядит простым копированием, оно выделено в специальную операцию по множеству причин. Сохранение концепции верхнего уровня как операции первого класса позволяет разработчикам более ясно представлять, что запрашивает клиент и что должен выполнить сервер. Это делает цели более понятными, специальные случаи менее сложными, а взаимодействие операций более прямым. Например, возможность :confirmed-commit:1.1 (параграф 8.4) не имеет смысла в качестве операции подтверждаемого копирования (copy confirmed).

Конфигурация-кандидат допускает совместное использование в нескольких сеансах. Пока у клиента нет конкретной информации о том, что конфигурация-кандидат не используется в других сессиях, он **должен** предполагать возможность её одновременного изменения в разных сессиях. Поэтому клиенту разумно блокировать конфигурацию-кандидат перед её изменением.

Клиент может отказаться от не представленных изменений конфигурации-кандидата с помощью операции <discard-changes>. Эта операция вернёт содержимое конфигурации-кандидата в состояние рабочей конфигурации.

8.3.2. Зависимости

Нет.

8.3.3. Идентификатор возможности

Возможность :candidate указывается строкой

```
urn:ietf:params:netconf:capability:candidate:1.0
```

8.3.4. Новые операции

8.3.4.1. <commit>

Описание

Когда содержимое хранилища-кандидата полностью подготовлено, данные из него могут быть представлены (commit) устройству для их проверки и подтверждения пригодности.

Для представления конфигурации-кандидата в качестве новой текущей конфигурации устройства служит операция <commit>.

Эта операция предлагает устройству реализовать данные конфигурации-кандидата. Если устройство не способно реализовать все изменения, предложенные конфигурацией-кандидатом, оно **должно** сохранить прежнюю конфигурацию. Если устройство может применить предложенную конфигурацию полностью, рабочая конфигурация **должна** быть заменена содержимым конфигурации-кандидата.

Если рабочая или предлагаемая конфигурация в данный момент заблокирована другой сессией, операция <commit> **должна** привести к отказу и отправке <error-tag> со значением in-use.

Если система не поддерживает возможность :candidate, операция <commit> будет недоступна.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

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

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

8.3.4.2. <discard-changes>

Если клиент решил отказаться от представления конфигурации-кандидата, он может использовать операцию <discard-changes> для копирования в конфигурацию-кандидат содержимого рабочей конфигурации.

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

Эта операция отбрасывает все не представленные изменения конфигурации-кандидата и возвращает ту в состояние рабочей конфигурации.

8.3.5. Изменение существующий операций

8.3.5.1. <get-config>, <edit-config>, <copy-config>, <validate>

Конфигурация-кандидат может служить в качестве источника или цели в любой из операций <get-config>, <edit-config>, <copy-config> и <validate> при её задании параметром <source> или <target>, соответственно. Для указания этой конфигурации служит элемент <candidate>.

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

8.3.5.2. <lock> и <unlock>

Конфигурация-кандидат может быть заблокирована операцией <lock> при указании <candidate> в качестве параметра <target>.

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

Для снятия блокировки конфигурации-кандидата служит операция <unlock> с элементом <candidate> в качестве параметра <target>.

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

```
    </unlock>
  </rpc>
```

При отказе клиента с не сохранёнными изменениями конфигурации-кандидата восстановление может оказаться сложным. Для упрощения восстановления все не сохранённые изменения отбрасываются при снятии блокировки в результате явного вызова операции <unlock> или неявно при отказе в сессии.

8.4. Подтверждаемое представление

8.4.1. Описание

Возможность :confirmed-commit:1.1 показывает поддержку сервером операции <cancel-commit> и параметров <confirmed>, <confirm-timeout>, <persist>, <persist-id> для операции <commit> (см. параграф 8.3).

Подтверждаемая операция <commit> **должна** быть отменена, если подтверждённая подача (commit) не состоялась в течение заданного интервала (по умолчанию 600 секунд = 10 минут). Подтверждённая подача - это операция <commit> без параметра <confirmed>. Интервал ожидания можно изменить с помощью параметра <confirm-timeout>. Если до истечения срока была введена следующая подтверждаемая операция <commit>, таймер сбрасывается в новое значение (по умолчанию 600 секунд). И подтверждённая подача, и следующая подтверждаемая операция <commit> **могут** вносить в конфигурацию дополнительные изменения.

Если в подтверждаемой операции подачи не задан элемент <persist>, любая последующая подтверждаемая и подтверждённая подача **должны** выполняться в той же сессии, где была введена подтверждаемая подача. Если элемент <persist> задан в подтверждаемой операции <commit>, следующая подача и подтверждённая подача могут выполняться из любой сессии и **должны** включать элемент <persist-id> значение которого совпадает со значением элемента <persist>.

Если сервер анонсирует также способность :startup, требуется операция <copy-config> (копирование из рабочего хранилища в стартовое) для сохранения изменений при перезагрузке устройства.

Если сессия, вызвавшая подтверждаемую подачу, была по какой-либо причине прервана до наступления тайм-аута подтверждения, сервер **должен** восстановить конфигурацию, которая была до подачи, если подтверждаемая подача не включала также элемент <persist>.

Если устройство по какой-либо причине перезагружается до завершения времени ожидания, сервер **должен** восстановить конфигурацию, которая была до подтверждаемого представления.

Если подтверждённое представление не введено, устройство будет возвращаться к конфигурации, которая была до ввода подтверждаемого представления. Для отказа от подтверждаемого представление и восстановления конфигурации, которая была до того, клиент может явно восстановить конфигурацию с помощью операции <cancel-commit>.

Для совместно используемых конфигураций эта возможность может приводить к непреднамеренному удалению чужих изменений (например, в других сессиях NETCONF), если не используется функция блокировки конфигурации (т. е. блокировка не организована до начала работы операции <edit-config>). Поэтому при использовании этой функции с общими конфигурационными хранилищами **следует** также применять блокировку конфигурации.

Версия 1.0 этой возможности была определена в [RFC4741]. Этот документ определяет версию 1.1, которая расширяет версию 1.0 путём добавления новой операции <cancel-commit> и двух новых необязательных параметров <persist> и <persist-id>. Для совместимости со старыми клиентами серверы, поддерживающие эту спецификацию, **могут** анонсировать версию 1.0 в дополнение к версии 1.1.

8.4.2. Зависимости

Возможность :confirmed-commit:1.1 имеет смысл только при наличии поддержки возможности :candidate.

8.4.3. Идентификатор возможности

Возможность :confirmed-commit:1.1 указывается строкой

```
urn:ietf:params:netconf:capability:confirmed-commit:1.1
```

8.4.4. Новые операции

8.4.4.1. <cancel-commit>

Описание

Отменяет подтверждаемую подачу конфигурации. Если параметр <persist-id> не задан, операция <cancel-commit> **должна** быть введена в той же сессии, что и подтверждаемое представление.

Параметры

persist-id

Отменяет устойчивую к разрыву сессии подтверждаемую подачу конфигурации. Значение **должно** совпадать со значением параметра <persist> в операции <commit>. Если значения различаются, операция завершается отказом с ошибкой invalid-value.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Пример

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

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

<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cancel-commit/>
</rpc>

<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

8.4.5. Изменение существующих операций

8.4.5.1. <commit>

Возможность :confirmed-commit:1.1 добавляет 4 новых параметра для операции <commit>.

Параметры

confirmed

Выполнить подтверждаемую операцию <commit>.

confirm-timeout

Время ожидания для подтверждаемого представления (в секундах). Если тайм-аут на задан, время ожидания составит 600 секунд.

persist

Делает подтверждаемое представление устойчивым к разрыву сессии и устанавливает маркер для выполняющегося подтверждаемого представления.

persist-id

Используется в следующем подтверждаемом представлении или в подтверждённом представлении из любой сессии с маркером предыдущей операции <commit>.

Пример

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <confirm-timeout>120</confirm-timeout>
  </commit>
</rpc>

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

Пример

```
<!-- начало устойчивой к разрыву сессии операции confirmed-commit -->
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <persist>IQ,d4668</persist>
  </commit>
</rpc>

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

<!-- подтверждение устойчивой к разрыву сессии операции confirmed-commit,
     confirmed-commit, возможно из другой сессии -->
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <persist-id>IQ,d4668</persist-id>
  </commit>
</rpc>

<rpc-reply message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

8.5. Возможность отката после ошибки

8.5.1. Описание

Эта возможность говорит о поддержке сервером значения rollback-on-error для параметра <error-option> операции <edit-config>.

Для совместно используемых конфигураций эта возможность может приводить к непреднамеренному удалению чужих изменений (например, в других сессиях NETCONF), если не используется функция блокировки конфигурации (т. е. блокировка не организована до начала работы операции <edit-config>). Поэтому при использовании этой функции с общими конфигурационными хранилищами **следует** также применять блокировку конфигурации.

8.5.2. Зависимости

Нет.

8.5.3. Идентификатор возможности

Возможность :rollback-on-error указывается строкой

```
urn:ietf:params:netconf:capability:rollback-on-error:1.0
```

8.5.4. Новые операции

Нет.

8.5.5. Изменения существующих операций

8.5.5.1. <edit-config>

Возможность :rollback-on-error позволяет указывать значение rollback-on-error для параметра <error-option> операции <edit-config>.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>100000</mtu>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>

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

8.6. Проверка пригодности

8.6.1. Описание

Проверка пригодности представляет собой полную проверку конфигурации на предмет наличия синтаксических и семантических ошибок до применения этой конфигурации в устройстве.

Анонсирование этой возможности означает поддержку протокольной операции <validate> и проверки по крайней мере синтаксических ошибок. Кроме того, эта возможность поддерживает параметр <test-option> для операции <edit-config>, который задаёт проверку по крайней мере синтаксических ошибок.

Версия 1.0 этой возможности была определена в [RFC4741]. Описанная здесь версия 1.1 расширяет версию 1.0 путём добавления значения test-only для параметра <test-option> операции <edit-config>. Для совместимости со старыми клиентами серверы, поддерживающие эту спецификацию, **могут** анонсировать версию 1.0 в дополнение к версии 1.1.

8.6.2. Зависимости

Нет.

8.6.3. Идентификатор возможности

Возможность :validate:1.1 указывается строкой

```
urn:ietf:params:netconf:capability:validate:1.1
```

8.6.4. Новые операции

8.6.4.1. <validate>

Описание

Эта протокольная операция проверяет пригодность содержимого указанной конфигурации.

Параметры

source

Имя конфигурационного хранилища для проверки (например, <candidate>) или элемент <config>, содержащий полную конфигурацию для проверки.

Позитивный отклик

Если устройство смогло выполнить запрос, возвращается отклик <rpc-reply> с элементом <ok>.

Негативный отклик

Если запрос не выполнен полностью, возвращается отклик <rpc-reply> с элементом <rpc-error>.

Операция <validate> может завершаться отказом по разным причинам - синтаксических ошибок, пропущенных параметров, ссылок на не определённые данные конфигурации и других нарушений правил, заданных базовой моделью.

Пример

```
<rpc message-id="101"
```

```
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <validate>
      <source>
        <candidate/>
      </source>
    </validate>
  </rpc>

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

8.6.5. Изменения существующих операций

8.6.5.1. <edit-config>

Возможность :validate:1.1 меняет операцию <edit-config>, позволяя принимать параметр <test-option>.

8.7. Отдельное стартовое хранилище

8.7.1. Описание

Устройство поддерживает разные хранилища для рабочей и стартовой конфигурации. Стартовая конфигурация используется при загрузке устройства. Операции, воздействующие на рабочее хранилище, не будут автоматически копироваться в стартовую конфигурацию. Явная операция <copy-config> из хранилища <running> в <startup> служит для обновления стартовой конфигурации в соответствии с текущим содержимым рабочей конфигурации. Операции протокола NETCONF указывают стартовое хранилище с помощью элемента <startup>.

8.7.2. Зависимости

Нет.

8.7.3. Идентификатор возможности

Возможность :startup указывается строкой

```
urn:ietf:params:netconf:capability:startup:1.0
```

8.7.4. Новые операции

Нет.

8.7.5. Изменения существующих операций

8.7.5.1. General

Возможность :startup добавляет конфигурационное хранилище <startup/> к аргументам некоторых операций NETCONF. Сервер **должен** поддерживать это значение для указанных в таблице параметров.

Операция	*Параметры*	*Примечания*
<get-config>	<source>	
<copy-config>	<source> <target>	
<lock>	<target>	
<unlock>	<target>	
<validate>	<source>	Если анонсирована возможность :validate:1.1
<delete-config>	<target>	Сбрасывает устройство в заводские установки

Для записи стартовой конфигурации используется команда <copy-config>, копирующая содержимое хранилища <running> в хранилище <startup>.

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

8.8. Поддержка URL

8.8.1. Описание

Узел NETCONF может поддерживать элемент <url> в качестве значения параметров <source> и <target>. Эта возможность дополнительно идентифицируется аргументами URL, указывающими поддерживаемые схемы URL.

8.8.2. Зависимости

Нет.

8.8.3. Идентификатор возможности

Возможность :url указывается строкой

```
urn:ietf:params:netconf:capability:url:1.0?scheme={name,...}
```

Идентификатор URI возможность :url **должен** включать аргумент scheme, содержащий список разделённых запятыми имён схем, поддерживаемых узлом NETCONF. Например,

```
urn:ietf:params:netconf:capability:url:1.0?scheme=http,ftp,file
```

8.8.4. Новые операции

Нет.

8.8.5. Изменения существующих операций

8.8.5.1. <edit-config>

Возможность :url меняет операцию <edit-config>, позволяя той принимать элемент <url> в качестве параметра <config>.

Файл, на который указывает url в качестве содержимого конфигурационных данных для редактирования, кодируется в формате XML внутри элемента <config> в пространстве имён urn:ietf:params:xml:ns:netconf:base:1.0.

8.8.5.2. <copy-config>

Возможность :url меняет операцию <copy-config>, позволяя той принимать элемент <url> в качестве параметров <source> и <target>.

Файл, на который указывает url в качестве полного конфигурационного хранилища, кодируется в формате XML внутри элемента <config> в пространстве имён urn:ietf:params:xml:ns:netconf:base:1.0.

8.8.5.3. <delete-config>

Возможность :url меняет операцию <delete-config>, позволяя ей принимать <url> в качестве параметра <target>.

8.8.5.4. <validate>

Возможность :url меняет операцию <validate>, позволяя ей принимать элемент <url> в качестве параметра <source>.

8.9. Выражения XPath

8.9.1. Описание

Возможность XPath показывает, что узел NETCONF поддерживает использование выражений XPath в элементах <filter>. Описание XPath приведено в [W3C.REC-xpath-19991116].

Модель данных в выражениях XPath совпадает с используемой в XPath 1.0 [W3C.REC-xpath-19991116] с тем же расширением для потомков корневого узла, которое применяется в XSLT 1.0 ([W3C.REC-xslt-19991116], параграф 3.1). В частности, это означает, что корневой узел **может** иметь в качестве потомков любое число элементов.

Выражения XPath оцениваются в следующем контексте:

- набор объявлений пространств имён составляют те, которые относятся к зоне действия элемента <filter>;
- набор привязок переменных определяется моделью данных и будет пуст, если привязки не заданы;
- библиотекой функций является основная библиотека, дополненная функциями модели данных;
- узлом контекста является корневой узел.

Выражение XPath **должно** возвращать набор узлов. Если этого не происходит, операция завершается отказом с ошибкой invalid-value.

Сообщение с откликом содержит ветви (subtree), выбранные фильтром. Для каждой такой ветви путь от корня модели данных вниз к ветви, включая все элементы или атрибуты, необходимые для однозначной идентификации ветви, включает в отклик. Конкретные экземпляры данных не дублируются в отклике.

8.9.2. Зависимости

Нет.

8.9.3. Идентификатор возможности

Возможность :xpath указывается строкой

```
urn:ietf:params:netconf:capability:xpath:1.0
```

8.9.4. Новые операции

Нет.

8.9.5. Изменения существующих операций

8.9.5.1. <get-config> и <get>

Возможность :xpath меняет операции <get> и <get-config>, позволяя им принимать значение xpath в атрибуте type элемента <filter>. Если атрибут type имеет значение xpath, в элементе <filter> **должен** присутствовать атрибут select, который будет считаться выражением XPath и применяться для фильтрации возвращаемых данных. В таком случае сам элемент <filter> **должен** быть пустым.

Результатом XPath для выбора **должен** быть набор узлов (node-set). Каждый узел в node-set **должен** соответствовать узлу в базовой модели данных. Для корректной идентификации каждого узла служат приведённые ниже правила.

- Все узлы-предки в результате **должны** кодироваться первыми, чтобы элемент <data>, возвращаемый в отклике, содержал лишь полностью заданные ветви в соответствии с базовой моделью данных.
- Если любые братские или родительские узлы в результате нужны для идентификации конкретного экземпляра в концептуальной структуре данных, эти узлы также **должны** быть представлены в отклике.

Пример

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <!-- найти пользователя с именем named fred -->
    <filter xmlns:t="http://example.com/schema/1.2/config"
            type="xpath"
            select="/t:top/t:users/t:user[t:name='fred']"/>
   </get-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>fred</name>
          <company-info>
            <id>2</id>
          </company-info>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

9. Вопросы безопасности

В этом разделе рассматриваются вопросы безопасности базового уровня сообщений NETCONF и базовых операций NETCONF. Вопросы безопасности транспорта NETCONF рассмотрены в соответствующих транспортных протоколах, а безопасность содержимого, с которым работает NETCONF, - в документах, определяющих модели данных.

Этот документ не задаёт схемы проверки и предоставления полномочий (authorization), поскольку такая схема будет скорей всего связана с моделью данных или метаданных. Разработчикам **следует** обеспечивать полнофункциональную схему контроля полномочий для NETCONF.

Проверка полномочий отдельного пользователя сервером NETCONF может (но не обязана) отображаться (1:1) на другие интерфейсы. Во-первых, модели данных могут быть несовместимы. Во-вторых, может оказаться желательной проверка полномочий на основе механизмов, доступных на уровне защищённого транспорта (например, SSH, BEEP[1] и т. п.).

Кроме того, операции с данными конфигурации могут приводить к нежелательным последствиям, если эти операции не защищены также глобальной блокировкой файлов или объектов, с которыми операция имеет дело. Например, если рабочую конфигурацию не заблокировать, может быть представлен неполный список контроля доступа из конфигурации-кандидата, ведущий к снижению уровня безопасности или даже к недоступности устройства.

Конфигурационные данные по своей природе конфиденциальны. Их передача в открытом виде без контроля целостности оставляет устройство открытым для классических атак с перехватом и вставкой фальсифицированных данных. Конфигурационные данные часто включают пароли, имена пользователей, описания услуг и топологические сведения, которые являются конфиденциальными. По этой причине данный протокол **следует** реализовать с адекватным учётом всех возможных атак.

По этим причинам протокол **должен** обеспечивать минимальную поддержку конфиденциальности и проверки подлинности. Предполагается, что базовый транспорт (SSH, BEEP и т. п.) будет обеспечивать требуемые услуги конфиденциальности и проверки подлинности. Предполагается также, что отождествление каждой стороны сессии NETCONF будет доступно другой стороне для контроля доступа при выполнении запросов. Можно также легко представить дополнительные методы для контроля полномочий. Протокол NETCONF, сам по себе, не предоставляет никаких средств повторной или первичной проверки подлинности. Все такие действия происходят на нижележащем уровне.

В разных средах могут предоставляться разные права до и после проверки подлинности. Поэтому модель проверки подлинности в документе не задана. Когда операцию пытаются выполнить без должных полномочий, достаточно простого отказа в доступе (access denied). Отметим, что обмен информацией о правах доступа, может происходить в форме обмена конфигурационными данными, что дополнительно усиливает требования по защите соединений.

Следует подчеркнуть важность понимания того, что разные операции по своей природе имеют разный уровень «деликатности». Например, операция <copy-config> для стартовой или рабочей конфигурации явно не является обычной операцией инициализации, как <edit-config>. Такие глобальные операции **должны** запрещать изменение данных лицам, не имеющим на то полномочий. Например, если пользователь A не имеет права настраивать адрес IP

[1]Blocks Extensible Exchange Protocol - расширяемый протокол обмена блоками.

на интерфейсе, а пользователь B настраивает такой адрес в конфигурации <candidate>, пользователю A **недопустимо** разрешать представление конфигурации <candidate>.

Точно так же, если кто-то скажет: «запишите конфигурацию, хранящуюся по ссылке URL,» - этого не следует делать без должной проверки полномочий.

Операция <lock> показывает, что протокол NETCONF предназначен для использования системами, которым администратор имеет некое доверие. Как указано в этом документе, можно заблокировать части конфигурации, что сделает их недоступными для других. В конечном итоге может быть заблокирована конфигурация целиком. Для смягчения этой проблемы есть два подхода. Можно «убить» другую сессию NETCONF программными средствами в рамках протокола NETCONF, если идентификатор этой сессии известен. Другим вариантом является снятие блокировки с использованием пользовательского интерфейса самого устройства. Эти два механизма могут действовать один против другого, что может быть устранено путём удаления нарушителя с сервера AAA[1]. Однако такое решение удобно не во всех случаях (например, при использовании парных ключей SSH).

10. Взаимодействие с IANA

10.1. Пространство имён NETCONF XML

Этот документ регистрирует URI для пространства имён NETCONF XML в реестре IETF XML [RFC3688].

Агентство IANA обновило приведённые ниже URI со ссылками на этот документ.

```
URI: urn:ietf:params:xml:ns:netconf:base:1.0
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

10.2. Схема NETCONF XML

Этот документ регистрирует URI для схемы NETCONF XML в реестре IETF XML [RFC3688].

Агентство IANA обновило приведённые ниже URI со ссылками на этот документ.

```
URI: urn:ietf:params:xml:schema:netconf
Registrant Contact: The IESG.
XML: Appendix B of this document.
```

10.3. Модуль для YANG NETCONF

Этот документ регистрирует модуль YANG в реестре YANG Module Names [RFC6020].

```
name:         ietf-netconf
namespace:    urn:ietf:params:xml:ns:netconf:base:1.0
prefix:       nc
reference:    RFC 6241
```

10.4. URN для возможностей NETCONF

Агенство IANA создало и поддерживает реестр Network Configuration Protocol (NETCONF) Capability URNs для идентификаторов возможностей NETCONF. Добавление в этот реестр выполняется по процедуре IETF Standards Action[2].

Агентство IANA обновило в реестре перечисленные в таблице возможности со ссылками на этот документ.

Индекс	*Идентификатор возможности*
:writable-running	urn:ietf:params:netconf:capability:writable-running:1.0
:candidate	urn:ietf:params:netconf:capability:candidate:1.0
:rollback-on-error	urn:ietf:params:netconf:capability:rollback-on-error:1.0
:startup	urn:ietf:params:netconf:capability:startup:1.0
:url	urn:ietf:params:netconf:capability:url:1.0
:xpath	urn:ietf:params:netconf:capability:xpath:1.0

В реестр добавлены также перечисленные в следующей таблице возможности.

Индекс	*Идентификатор возможности*
:base:1.1	urn:ietf:params:netconf:base:1.1
:confirmed-commit:1.1	urn:ietf:params:netconf:capability:confirmed-commit:1.1
:validate:1.1	urn:ietf:params:netconf:capability:validate:1.1

11. Авторы

Кроме редакторов в написании этого документа принимали участие

Ken Crozier, Cisco Systems;
Ted Goddard, IceSoft;
Eliot Lear, Cisco Systems;
Phil Shafer, Juniper Networks;
Steve Waldbusser;
Margaret Wasserman, Painless Security, LLC.

12. Благодарности

Авторы признательны члена рабочей группы NETCONF. В частности, авторы благодарны Wes Hardaker за настойчивость и терпение при разъяснении вопросов безопасности. Спасибо также Randy Presuhn, Sharon Chisholm, Glenn Waters, David Perkins, Weijing Chen, Simon Leinen, Keith Allen, Dave Harrington, Ladislav Lhotka, Tom Petch и Kent Watsen за их ценные советы.

[1]Authentication, Authorization, and Accounting - проверка подлинности, контроль полномочий и учёт.
[2]В RFC 7803 это требование было смягчено. *Прим. перев.*

13. Литература

13.1. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, June 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5717] Lengyel, B. and M. Bjorklund, "Partial Lock Remote Procedure Call (RPC) for NETCONF", RFC 5717, December 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020[1], October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6242] Wasserman, M., "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[W3C.REC-xml-20001006] Sperberg-McQueen, C., Bray, T., Paoli, J., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium REC-xml-20001006, October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>[2].

[W3C.REC-xpath-19991116] DeRose, S. and J. Clark, "XML Path Language (Xpath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

13.2. Дополнительная литература

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3470] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, January 2003.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[W3C.REC-xslt-19991116] Clark, J., "XSL Transformations (XSLT) Version 1.0", World Wide Web Consortium Recommendation REC-xslt-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xslt-19991116>.

Приложение A. Список ошибок NETCONF

Это приложение является нормативным.

Для каждого error-tag указаны действительные значения error-type и error-severity вместе с обязательными данными error-info, если они есть.

```
error-tag:      in-use
error-type:     protocol, application
error-severity: error
error-info:     none
Description:    The request requires a resource that already is in use.

error-tag:      invalid-value
error-type:     protocol, application
error-severity: error
error-info:     none
Description:    The request specifies an unacceptable value for one or more parameters.

error-tag:      too-big
error-type:     transport, rpc, protocol, application
error-severity: error
error-info:     none
Description:    The request or response (that would be generated) is
                too large for the implementation to handle.

error-tag:      missing-attribute
error-type:     rpc, protocol, application
error-severity: error
```

[1]В RFC 7950 определена спецификация версии 1.1 протокола YANG, однако новый документ не отменяет RFC 6020. *Прим. перев.*
[2]Этот документ был обновлён. На момент публикации перевода новый вариант был доступен по ссылке. *Прим. перев.*

```
   error-info:     <bad-attribute> : name of the missing attribute
                   <bad-element> : name of the element that is supposed
                     to contain the missing attribute
   Description:    An expected attribute is missing.

   error-tag:      bad-attribute
   error-type:     rpc, protocol, application
   error-severity: error
   error-info:     <bad-attribute> : name of the attribute w/ bad value
                   <bad-element> : name of the element that contains
                     the attribute with the bad value
   Description:    An attribute value is not correct; e.g., wrong type,
                   out of range, pattern mismatch.

   error-tag:      unknown-attribute
   error-type:     rpc, protocol, application
   error-severity: error
   error-info:     <bad-attribute> : name of the unexpected attribute
                   <bad-element> : name of the element that contains
                     the unexpected attribute
   Description:    An unexpected attribute is present.

   error-tag:      missing-element
   error-type:     protocol, application
   error-severity: error
   error-info:     <bad-element> : name of the missing element
   Description:    An expected element is missing.

   error-tag:      bad-element
   error-type:     protocol, application
   error-severity: error
   error-info:     <bad-element> : name of the element w/ bad value
   Description:    An element value is not correct; e.g., wrong type,
                   out of range, pattern mismatch.

   error-tag:      unknown-element
   error-type:     protocol, application
   error-severity: error
   error-info:     <bad-element> : name of the unexpected element
   Description:    An unexpected element is present.

   error-tag:      unknown-namespace
   error-type:     protocol, application
   error-severity: error
   error-info:     <bad-element> : name of the element that contains
                     the unexpected namespace
                   <bad-namespace> : name of the unexpected namespace
   Description:    An unexpected namespace is present.

   error-tag:      access-denied
   error-type:     protocol, application
   error-severity: error
   error-info:     none
   Description:    Access to the requested protocol operation or
                   data model is denied because authorization failed.
   error-tag:      lock-denied
   error-type:     protocol
   error-severity: error
   error-info:     <session-id> : session ID of session holding the
                     requested lock, or zero to indicate a non-NETCONF
                     entity holds the lock
   Description:    Access to the requested lock is denied because the
                   lock is currently held by another entity.

   error-tag:      resource-denied
   error-type:     transport, rpc, protocol, application
   error-severity: error
   error-info:     none
   Description:    Request could not be completed because of
                   insufficient resources.

   error-tag:      rollback-failed
   error-type:     protocol, application
   error-severity: error
   error-info:     none
   Description:    Request to roll back some configuration change (via
                   rollback-on-error or <discard-changes> operations)
                   was not completed for some reason.

   error-tag:      data-exists
   error-type:     application
   error-severity: error
   error-info:     none
   Description:    Request could not be completed because the relevant
                   data model content already exists.  For example,
                   a "create" operation was attempted on data that
```

```
                already exists.

error-tag:      data-missing
error-type:     application
error-severity: error
error-info:     none
Description:    Request could not be completed because the relevant
                data model content does not exist.  For example,
                a "delete" operation was attempted on
                data that does not exist.

error-tag:      operation-not-supported
error-type:     protocol, application
error-severity: error
error-info:     none
Description:    Request could not be completed because the requested
                operation is not supported by this implementation.
error-tag:      operation-failed
error-type:     rpc, protocol, application
error-severity: error
error-info:     none
Description:    Request could not be completed because the requested
                operation failed for some reason not covered by
                any other error condition.

error-tag:      partial-operation
error-type:     application
error-severity: error
error-info:     <ok-element> : identifies an element in the data
                  model for which the requested operation has been
                  completed for that node and all its child nodes.
                  This element can appear zero or more times in the
                  <error-info> container.

                <err-element> : identifies an element in the data
                  model for which the requested operation has failed
                  for that node and all its child nodes.
                  This element can appear zero or more times in the
                  <error-info> container.

                <noop-element> : identifies an element in the data
                  model for which the requested operation was not
                  attempted for that node and all its child nodes.
                  This element can appear zero or more times in the
                  <error-info> container.

Description:    This error-tag is obsolete, and SHOULD NOT be sent
                by servers conforming to this document.

                Some part of the requested operation failed or was
                not attempted for some reason.  Full cleanup has
                not been performed (e.g., rollback not supported)
                by the server.  The error-info container is used
                to identify which portions of the application
                data model content for which the requested operation
                has succeeded (<ok-element>), failed (<bad-element>),
                or not been attempted (<noop-element>).
error-tag:      malformed-message
error-type:     rpc
error-severity: error
error-info:     none
Description:    A message could not be handled because it failed to
                be parsed correctly.  For example, the message is not
                well-formed XML or it uses an invalid character set.

                This error-tag is new in :base:1.1 and MUST NOT be
                sent to old clients.
```

Приложение B. Схема XML для уровня сообщений NETCONF

Это приложение является нормативным.

```
<CODE BEGINS> file "netconf.xsd"

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           targetNamespace="urn:ietf:params:xml:ns:netconf:base:1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           xml:lang="en"
           version="1.1">

  <xs:annotation>
    <xs:documentation>
      This schema defines the syntax for the NETCONF Messages layer
      messages 'hello', 'rpc', and 'rpc-reply'.
```

```
      </xs:documentation>
    </xs:annotation>

    <!--
       импорт стандартных определений XML
      -->
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
               schemaLocation="http://www.w3.org/2001/xml.xsd">
      <xs:annotation>
        <xs:documentation>
          This import accesses the xml: attribute groups for the
          xml:lang as declared on the error-message element.
        </xs:documentation>
      </xs:annotation>
    </xs:import>
    <!--
       Атрибут message-id
      -->
    <xs:simpleType name="messageIdType">
      <xs:restriction base="xs:string">
        <xs:maxLength value="4095"/>
      </xs:restriction>
    </xs:simpleType>
    <!--
       Типы, используемые для session-id
      -->
    <xs:simpleType name="SessionId">
      <xs:restriction base="xs:unsignedInt">
        <xs:minInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="SessionIdOrZero">
      <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <!--
       Элемент <rpc>
      -->
    <xs:complexType name="rpcType">
      <xs:sequence>
        <xs:element ref="rpcOperation"/>
      </xs:sequence>
      <xs:attribute name="message-id" type="messageIdType"
                    use="required"/>
      <!--
         С элементом <rpc> могут представляться произвольные атрибуты.
        -->
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:element name="rpc" type="rpcType"/>
    <!--
       Типы данные и элементы, используемые для создания rpc-error
      -->
    <xs:simpleType name="ErrorType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="transport"/>
        <xs:enumeration value="rpc"/>
        <xs:enumeration value="protocol"/>
        <xs:enumeration value="application"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ErrorTag">
      <xs:restriction base="xs:string">
        <xs:enumeration value="in-use"/>
        <xs:enumeration value="invalid-value"/>
        <xs:enumeration value="too-big"/>
        <xs:enumeration value="missing-attribute"/>
        <xs:enumeration value="bad-attribute"/>
        <xs:enumeration value="unknown-attribute"/>
        <xs:enumeration value="missing-element"/>
        <xs:enumeration value="bad-element"/>
        <xs:enumeration value="unknown-element"/>
        <xs:enumeration value="unknown-namespace"/>
        <xs:enumeration value="access-denied"/>
        <xs:enumeration value="lock-denied"/>
        <xs:enumeration value="resource-denied"/>
        <xs:enumeration value="rollback-failed"/>
        <xs:enumeration value="data-exists"/>
        <xs:enumeration value="data-missing"/>
        <xs:enumeration value="operation-not-supported"/>
        <xs:enumeration value="operation-failed"/>
        <xs:enumeration value="partial-operation"/>
        <xs:enumeration value="malformed-message"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ErrorSeverity">
      <xs:restriction base="xs:string">
```

```
        <xs:enumeration value="error"/>
        <xs:enumeration value="warning"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="errorInfoType">
      <xs:sequence>
        <xs:choice>
          <xs:element name="session-id" type="SessionIdOrZero"/>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:sequence>
              <xs:element name="bad-attribute" type="xs:QName"
                          minOccurs="0" maxOccurs="1"/>
              <xs:element name="bad-element" type="xs:QName"
                          minOccurs="0" maxOccurs="1"/>
              <xs:element name="ok-element" type="xs:QName"
                          minOccurs="0" maxOccurs="1"/>
              <xs:element name="err-element" type="xs:QName"
                          minOccurs="0" maxOccurs="1"/>
              <xs:element name="noop-element" type="xs:QName"
                          minOccurs="0" maxOccurs="1"/>
              <xs:element name="bad-namespace" type="xs:string"
                          minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
          </xs:sequence>
        </xs:choice>
        <!-- Элементы из любого другого пространства имён также могут
             следовать за элементами NETCONF -->
        <xs:any namespace="##other" processContents="lax"
                minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="rpcErrorType">
      <xs:sequence>
        <xs:element name="error-type" type="ErrorType"/>
        <xs:element name="error-tag" type="ErrorTag"/>
        <xs:element name="error-severity" type="ErrorSeverity"/>
        <xs:element name="error-app-tag" type="xs:string"
                    minOccurs="0"/>
        <xs:element name="error-path" type="xs:string" minOccurs="0"/>
        <xs:element name="error-message" minOccurs="0">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute ref="xml:lang" use="optional"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="error-info" type="errorInfoType"
                    minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
    <!--
       Атрибуты операции, используемые в <edit-config>
      -->
    <xs:simpleType name="editOperationType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="merge"/>
        <xs:enumeration value="replace"/>
        <xs:enumeration value="create"/>
        <xs:enumeration value="delete"/>
        <xs:enumeration value="remove"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:attribute name="operation" type="editOperationType"/>
    <!--
       Элемент <rpc-reply>
      -->
    <xs:complexType name="rpcReplyType">
      <xs:choice>
        <xs:element name="ok"/>
        <xs:sequence>
          <xs:element ref="rpc-error"
                      minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="rpcResponse"
                      minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:choice>
      <xs:attribute name="message-id" type="messageIdType"
                    use="optional"/>
      <!--
         Любые атрибуты, представленные с элементом <rpc>, должны возвращаться
         с <rpc-reply>.
        -->
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
```

```
    <xs:element name="rpc-reply" type="rpcReplyType"/>
    <!--
       Элемент <rpc-error>
         -->
    <xs:element name="rpc-error" type="rpcErrorType"/>
    <!--
       rpcOperationType используется в качестве базового типа
       для всех операций NETCONF
      -->
    <xs:complexType name="rpcOperationType"/>
    <xs:element name="rpcOperation" type="rpcOperationType"
                abstract="true"/>
    <!--
       rpcResponseType используется в качестве базового типа
       для всех откликов NETCONF
      -->
    <xs:complexType name="rpcResponseType"/>
    <xs:element name="rpcResponse" type="rpcResponseType"
                abstract="true"/>
    <!--
       Элемент <hello>
      -->
    <xs:element name="hello">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="capabilities">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="capability" type="xs:anyURI"
                            maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="session-id" type="SessionId"
                      minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>

  <CODE ENDS>
```

Приложение C. Модуль YANG для операций протокола NETCONF

Это приложение является нормативным.

Модуль YANG ietf-netconf импортирует определения типов (typedef) из [RFC6021].

```
  <CODE BEGINS> file "ietf-netconf@2011-06-01.yang"

  module ietf-netconf {

    // Пространство имён для определений NETCONF XML не изменилось
    // по сравнению RFC 4741, который этот документ заменяет
    namespace "urn:ietf:params:xml:ns:netconf:base:1.0";

    prefix nc;

    import ietf-inet-types {
      prefix inet;
    }

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

    contact
      "WG Web:   <http://tools.ietf.org/wg/netconf/>
       WG List:  <netconf@ietf.org>

       WG Chair: Bert Wijnen
                 <bertietf@bwijnen.net>

       WG Chair: Mehmet Ersue
                 <mehmet.ersue@nsn.com>

       Editor:   Martin Bjorklund
                 <mbj@tail-f.com>

       Editor:   Juergen Schoenwaelder
                 <j.schoenwaelder@jacobs-university.de>

       Editor:   Andy Bierman
                 <andy.bierman@brocade.com>";

    description
      "NETCONF Protocol Data Types and Protocol Operations.
```

```
      Copyright (c) 2011 IETF Trust and the persons identified as
      the document authors.  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 Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC 6241; see
      the RFC itself for full legal notices.";
    revision 2011-06-01 {
      description
        "Initial revision";
      reference
        "RFC 6241: Network Configuration Protocol";
    }

    extension get-filter-element-attributes {
      description
        "If this extension is present within an 'anyxml'
         statement named 'filter', which must be conceptually
         defined within the RPC input section for the <get>
         and <get-config> protocol operations, then the
         following unqualified XML attribute is supported
         within the <filter> element, within a <get> or
         <get-config> protocol operation:

           type : optional attribute with allowed
                  value strings 'subtree' and 'xpath'.
                  If missing, the default value is 'subtree'.

         If the 'xpath' feature is supported, then the
         following unqualified XML attribute is
         also supported:

           select: optional attribute containing a
                   string representing an XPath expression.
                   The 'type' attribute must be equal to 'xpath'
                   if this attribute is present.";
    }

    // Возможности NETCONF, определённые как функции
    feature writable-running {
      description
        "NETCONF :writable-running capability;
         If the server advertises the :writable-running
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.2";
    }

    feature candidate {
      description
        "NETCONF :candidate capability;
         If the server advertises the :candidate
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.3";
    }

    feature confirmed-commit {
      if-feature candidate;
      description
        "NETCONF :confirmed-commit:1.1 capability;
         If the server advertises the :confirmed-commit:1.1
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";

      reference "RFC 6241, Section 8.4";
    }

    feature rollback-on-error {
      description
        "NETCONF :rollback-on-error capability;
         If the server advertises the :rollback-on-error
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.5";
    }
```

```
    feature validate {
      description
        "NETCONF :validate:1.1 capability;
         If the server advertises the :validate:1.1
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.6";
    }

    feature startup {
      description
        "NETCONF :startup capability;
         If the server advertises the :startup
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.7";
    }

    feature url {
      description
        "NETCONF :url capability;
         If the server advertises the :url
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.8";
    }

    feature xpath {
      description
        "NETCONF :xpath capability;
         If the server advertises the :xpath
         capability for a session, then this feature must
         also be enabled for that session.  Otherwise,
         this feature must not be enabled.";
      reference "RFC 6241, Section 8.9";
    }

    // Простые типы NETCONF

    typedef session-id-type {
      type uint32 {
        range "1..max";
      }
      description
        "NETCONF Session Id";
    }

    typedef session-id-or-zero-type {
      type uint32;
      description
        "NETCONF Session Id or Zero to indicate none";
    }
    typedef error-tag-type {
      type enumeration {
          enum in-use {
            description
              "The request requires a resource that
               already is in use.";
          }
          enum invalid-value {
            description
              "The request specifies an unacceptable value for one
               or more parameters.";
          }
          enum too-big {
            description
              "The request or response (that would be generated) is
               too large for the implementation to handle.";
          }
          enum missing-attribute {
            description
              "An expected attribute is missing.";
          }
          enum bad-attribute {
            description
              "An attribute value is not correct; e.g., wrong type,
               out of range, pattern mismatch.";
          }
          enum unknown-attribute {
            description
              "An unexpected attribute is present.";
          }
          enum missing-element {
```

```
          description
            "An expected element is missing.";
        }
        enum bad-element {
          description
            "An element value is not correct; e.g., wrong type,
             out of range, pattern mismatch.";
        }
        enum unknown-element {
          description
            "An unexpected element is present.";
        }
        enum unknown-namespace {
          description
            "An unexpected namespace is present.";
        }
        enum access-denied {
          description
            "Access to the requested protocol operation or
             data model is denied because authorization failed.";
        }
        enum lock-denied {
          description
            "Access to the requested lock is denied because the
             lock is currently held by another entity.";
        }
        enum resource-denied {
          description
            "Request could not be completed because of
             insufficient resources.";
        }
        enum rollback-failed {
          description
            "Request to roll back some configuration change (via
             rollback-on-error or <discard-changes> operations)
             was not completed for some reason.";

        }
        enum data-exists {
          description
            "Request could not be completed because the relevant
             data model content already exists.  For example,
             a 'create' operation was attempted on data that
             already exists.";
        }
        enum data-missing {
          description
            "Request could not be completed because the relevant
             data model content does not exist.  For example,
             a 'delete' operation was attempted on
             data that does not exist.";
        }
        enum operation-not-supported {
          description
            "Request could not be completed because the requested
             operation is not supported by this implementation.";
        }
        enum operation-failed {
          description
            "Request could not be completed because the requested
             operation failed for some reason not covered by
             any other error condition.";
        }
        enum partial-operation {
          description
            "This error-tag is obsolete, and SHOULD NOT be sent
             by servers conforming to this document.";
        }
        enum malformed-message {
          description
            "A message could not be handled because it failed to
             be parsed correctly.  For example, the message is not
             well-formed XML or it uses an invalid character set.";
        }
      }
      description "NETCONF Error Tag";
      reference "RFC 6241, Appendix A";
    }

    typedef error-severity-type {
      type enumeration {
        enum error {
          description "Error severity";
        }
        enum warning {
          description "Warning severity";
```

```
      }
    }
    description "NETCONF Error Severity";
    reference "RFC 6241, Section 4.3";
  }

  typedef edit-operation-type {
    type enumeration {
      enum merge {
        description
          "The configuration data identified by the
           element containing this attribute is merged
           with the configuration at the corresponding
           level in the configuration datastore identified
           by the target parameter.";
      }
      enum replace {
        description
          "The configuration data identified by the element
           containing this attribute replaces any related
           configuration in the configuration datastore
           identified by the target parameter.  If no such
           configuration data exists in the configuration
           datastore, it is created.  Unlike a
           <copy-config> operation, which replaces the
           entire target configuration, only the configuration
           actually present in the config parameter is affected.";
      }
      enum create {
        description
          "The configuration data identified by the element
           containing this attribute is added to the
           configuration if and only if the configuration
           data does not already exist in the configuration
           datastore.  If the configuration data exists, an
           <rpc-error> element is returned with an
           <error-tag> value of 'data-exists'.";
      }
      enum delete {
        description
          "The configuration data identified by the element
           containing this attribute is deleted from the
           configuration if and only if the configuration
           data currently exists in the configuration
           datastore.  If the configuration data does not
           exist, an <rpc-error> element is returned with
           an <error-tag> value of 'data-missing'.";
      }
      enum remove {
        description
          "The configuration data identified by the element
           containing this attribute is deleted from the
           configuration if the configuration
           data currently exists in the configuration
           datastore.  If the configuration data does not
           exist, the 'remove' operation is silently ignored
           by the server.";
      }
    }
    default "merge";
    description "NETCONF 'operation' attribute values";
    reference "RFC 6241, Section 7.2";
  }

  // Стандартные протокольные операции NETCONF

  rpc get-config {
    description
      "Retrieve all or part of a specified configuration.";

    reference "RFC 6241, Section 7.1";

    input {
      container source {
        description
          "Particular configuration to retrieve.";

        choice config-source {
          mandatory true;
          description
            "The configuration to retrieve.";
          leaf candidate {
            if-feature candidate;
            type empty;
            description
              "The candidate configuration is the config source.";
```

```
          }
          leaf running {
            type empty;
            description
              "The running configuration is the config source.";
          }
          leaf startup {
            if-feature startup;
            type empty;
            description
              "The startup configuration is the config source.
               This is optional-to-implement on the server because
               not all servers will support filtering for this
               datastore.";
          }
        }
      }

      anyxml filter {
        description
          "Subtree or XPath filter to use.";
        nc:get-filter-element-attributes;
      }
    }

    output {
      anyxml data {
        description
          "Copy of the source datastore subset that matched
           the filter criteria (if any).  An empty data container
           indicates that the request did not produce any results.";
      }
    }
  }

  rpc edit-config {
    description
      "The <edit-config> operation loads all or part of a specified
       configuration to the specified target configuration.";

    reference "RFC 6241, Section 7.2";

    input {
      container target {
        description
          "Particular configuration to edit.";

        choice config-target {
          mandatory true;
          description
            "The configuration target.";

          leaf candidate {
            if-feature candidate;
            type empty;
            description
              "The candidate configuration is the config target.";
          }
          leaf running {
            if-feature writable-running;
            type empty;
            description
              "The running configuration is the config source.";
          }
        }
      }

      leaf default-operation {
        type enumeration {
          enum merge {
            description
              "The default operation is merge.";
          }
          enum replace {
            description
              "The default operation is replace.";
          }
          enum none {
            description
              "There is no default operation.";
          }
        }
        default "merge";
        description
          "The default operation to use.";
      }
```

```
        leaf test-option {
          if-feature validate;
          type enumeration {
            enum test-then-set {
              description
                "The server will test and then set if no errors.";
            }
            enum set {
              description
                "The server will set without a test first.";
            }
            enum test-only {
              description
                "The server will only test and not set, even
                 if there are no errors.";
            }
          }
          default "test-then-set";
          description
            "The test option to use.";
        }

        leaf error-option {
          type enumeration {
            enum stop-on-error {
              description
                "The server will stop on errors.";
            }
            enum continue-on-error {
              description
                "The server may continue on errors.";
            }
            enum rollback-on-error {
              description
                "The server will roll back on errors.
                 This value can only be used if the 'rollback-on-error'
                 feature is supported.";
            }
          }
          default "stop-on-error";
          description
            "The error option to use.";
        }

        choice edit-content {
          mandatory true;
          description
            "The content for the edit operation.";

          anyxml config {
            description
              "Inline Config content.";
          }
          leaf url {
            if-feature url;
            type inet:uri;
            description
              "URL-based config content.";
          }
        }
      }
    }

    rpc copy-config {
      description
        "Create or replace an entire configuration datastore with the
         contents of another complete configuration datastore.";

      reference "RFC 6241, Section 7.3";

      input {
        container target {
          description
            "Particular configuration to copy to.";

          choice config-target {
            mandatory true;
            description
              "The configuration target of the copy operation.";

            leaf candidate {
              if-feature candidate;
              type empty;
              description
                "The candidate configuration is the config target.";
```

```
          }
          leaf running {
            if-feature writable-running;
            type empty;
            description
              "The running configuration is the config target.
               This is optional-to-implement on the server.";
          }
          leaf startup {
            if-feature startup;
            type empty;
            description
              "The startup configuration is the config target.";
          }
          leaf url {
            if-feature url;
            type inet:uri;
            description
              "The URL-based configuration is the config target.";
          }
        }
      }

      container source {
        description
          "Particular configuration to copy from.";

        choice config-source {
          mandatory true;
          description
            "The configuration source for the copy operation.";

          leaf candidate {
            if-feature candidate;
            type empty;
            description
              "The candidate configuration is the config source.";
          }
          leaf running {
            type empty;
            description
              "The running configuration is the config source.";
          }
          leaf startup {
            if-feature startup;
            type empty;
            description
              "The startup configuration is the config source.";
          }
          leaf url {
            if-feature url;
            type inet:uri;
            description
              "The URL-based configuration is the config source.";
          }
          anyxml config {
            description
              "Inline Config content: <config> element.  Represents
               an entire configuration datastore, not
               a subset of the running datastore.";
          }
        }
      }
    }
  }

  rpc delete-config {
    description
      "Delete a configuration datastore.";

    reference "RFC 6241, Section 7.4";

    input {
      container target {
        description
          "Particular configuration to delete.";

        choice config-target {
          mandatory true;
          description
            "The configuration target to delete.";

          leaf startup {
            if-feature startup;
            type empty;
            description
```

```
              "The startup configuration is the config target.";
          }
          leaf url {
            if-feature url;
            type inet:uri;
            description
              "The URL-based configuration is the config target.";
          }
        }
      }
    }
  }

  rpc lock {
    description
      "The lock operation allows the client to lock the configuration
       system of a device.";
    reference "RFC 6241, Section 7.5";

    input {
      container target {
        description
          "Particular configuration to lock.";

        choice config-target {
          mandatory true;
          description
            "The configuration target to lock.";

          leaf candidate {
            if-feature candidate;
            type empty;
            description
              "The candidate configuration is the config target.";
          }
          leaf running {
            type empty;
            description
              "The running configuration is the config target.";
          }
          leaf startup {
            if-feature startup;
            type empty;
            description
              "The startup configuration is the config target.";
          }
        }
      }
    }
  }

  rpc unlock {
    description
      "The unlock operation is used to release a configuration lock,
       previously obtained with the 'lock' operation.";

    reference "RFC 6241, Section 7.6";

    input {
      container target {
        description
          "Particular configuration to unlock.";

        choice config-target {
          mandatory true;
          description
            "The configuration target to unlock.";

          leaf candidate {
            if-feature candidate;
            type empty;
            description
              "The candidate configuration is the config target.";
          }
          leaf running {
            type empty;
            description
              "The running configuration is the config target.";
          }
          leaf startup {
            if-feature startup;
            type empty;
            description
              "The startup configuration is the config target.";
          }
        }
```

```
      }
    }
  }

  rpc get {
    description
      "Retrieve running configuration and device state information.";

    reference "RFC 6241, Section 7.7";

    input {
      anyxml filter {
        description
          "This parameter specifies the portion of the system
           configuration and state data to retrieve.";
        nc:get-filter-element-attributes;
      }
    }

    output {
      anyxml data {
        description
          "Copy of the running datastore subset and/or state
           data that matched the filter criteria (if any).
           An empty data container indicates that the request did not
           produce any results.";
      }
    }
  }

  rpc close-session {
    description
      "Request graceful termination of a NETCONF session.";

    reference "RFC 6241, Section 7.8";
  }

  rpc kill-session {
    description
      "Force the termination of a NETCONF session.";

    reference "RFC 6241, Section 7.9";

    input {
      leaf session-id {
        type session-id-type;
        mandatory true;
        description
          "Particular session to kill.";
      }
    }
  }

  rpc commit {
    if-feature candidate;

    description
      "Commit the candidate configuration as the device's new
       current configuration.";

    reference "RFC 6241, Section 8.3.4.1";

    input {
      leaf confirmed {
        if-feature confirmed-commit;
        type empty;
        description
          "Requests a confirmed commit.";
        reference "RFC 6241, Section 8.3.4.1";
      }

      leaf confirm-timeout {
        if-feature confirmed-commit;
        type uint32 {
          range "1..max";
        }
        units "seconds";
        default "600";   // 10 минут
        description
          "The timeout interval for a confirmed commit.";
        reference "RFC 6241, Section 8.3.4.1";
      }

      leaf persist {
        if-feature confirmed-commit;
        type string;
```

```
        description
          "This parameter is used to make a confirmed commit
           persistent.  A persistent confirmed commit is not aborted
           if the NETCONF session terminates.  The only way to abort
           a persistent confirmed commit is to let the timer expire,
           or to use the <cancel-commit> operation.

           The value of this parameter is a token that must be given
           in the 'persist-id' parameter of <commit> or
           <cancel-commit> operations in order to confirm or cancel
           the persistent confirmed commit.

           The token should be a random string.";
        reference "RFC 6241, Section 8.3.4.1";
      }

      leaf persist-id {
        if-feature confirmed-commit;
        type string;
        description
          "This parameter is given in order to commit a persistent
           confirmed commit.  The value must be equal to the value
           given in the 'persist' parameter to the <commit> operation.
           If it does not match, the operation fails with an
          'invalid-value' error.";
        reference "RFC 6241, Section 8.3.4.1";
      }
    }
  }

  rpc discard-changes {
    if-feature candidate;

    description
      "Revert the candidate configuration to the current
       running configuration.";
    reference "RFC 6241, Section 8.3.4.2";
  }

  rpc cancel-commit {
    if-feature confirmed-commit;
    description
      "This operation is used to cancel an ongoing confirmed commit.
       If the confirmed commit is persistent, the parameter
       'persist-id' must be given, and it must match the value of the
       'persist' parameter.";
    reference "RFC 6241, Section 8.4.4.1";

    input {
      leaf persist-id {
        type string;
        description
          "This parameter is given in order to cancel a persistent
           confirmed commit.  The value must be equal to the value
           given in the 'persist' parameter to the <commit> operation.
           If it does not match, the operation fails with an
          'invalid-value' error.";
      }
    }
  }

  rpc validate {
    if-feature validate;

    description
      "Validates the contents of the specified configuration.";

    reference "RFC 6241, Section 8.6.4.1";

    input {
      container source {
        description
          "Particular configuration to validate.";

        choice config-source {
          mandatory true;
          description
            "The configuration source to validate.";

          leaf candidate {
            if-feature candidate;
            type empty;
            description
              "The candidate configuration is the config source.";
          }
          leaf running {
```

```
              type empty;
              description
                "The running configuration is the config source.";
            }
            leaf startup {
              if-feature startup;
              type empty;
              description
                "The startup configuration is the config source.";
            }
            leaf url {
              if-feature url;
              type inet:uri;
              description
                "The URL-based configuration is the config source.";
            }
            anyxml config {
              description
                "Inline Config content: <config> element.  Represents
                 an entire configuration datastore, not
                 a subset of the running datastore.";
            }
          }
        }
      }
    }
  }

  <CODE ENDS>
```

Приложение D. Шаблон возможности

В этот не нормативном приложении определён шаблон, который может использоваться для определения возможностей протокола. Моделям данных YANG обычно не требуется определять возможности протокола, поскольку использование YANG автоматически ведёт к анонсированию модели данных и всех необязательных частей этой модели, которые в YANG называются функциями. Шаблон возможностей предназначен для использования в тех случаях, когда механизмов YANG недостаточно (например, при обработке параметризованных функций) или применяется другой язык моделирования данных.

D.1. capability-name (шаблон)

D.1.1. Обзор

D.1.2. Зависимости

D.1.3. Идентификатор свойства

Имя {name} свойства указывается строкой вида

```
{capability uri}
```

D.1.4. Новые операции

D.1.4.1. <op-name>

D.1.5. Изменение операций

D.1.5.1. <op-name>

Если имеющиеся операции это свойство не меняет, это раздел можно опустить.

D.1.6. Взаимодействия с другими свойствами

Если это свойство не взаимодействует с другими, раздел можно опустить.

Приложение E. Настройка множества устройств в NETCONF

Это приложение не является нормативным.

E.1. Операции с отдельным устройством

Рассмотрим работу, связанную с обновлением конфигурации отдельно взятого устройства. Принимая изменения конфигурации, приложение должно верить, что изменение выполнено корректно и не будет препятствовать работе устройства. Приложению (и его пользователю) нужна уверенность в том, что изменение не повредит сети.

Защищённая смена конфигурации каждого отдельного устройства включает множество этапов:

- блокировка конфигурации;
- создание контрольной точки для рабочей конфигурации;
- загрузка и проверка пригодности новой конфигурации;
- замена рабочей конфигурации;
- тестирование новой конфигурации;
- сохранение изменений (если они устраивают);

- снятие блокировки.

Далее все эти этапы рассмотрены более подробно.

E.1.1. Блокировка конфигурации

Блокировка нужна для предотвращения одновременного обновления конфигурации из нескольких источников. При воздействии на устройство нескольких источников ему будет сложно проверить изменения и восстановить конфигурацию в случае неудачи. Краткосрочная блокировка является самой простой защитой от попыток внесения изменений с нескольких сторон.

Блокировку можно организовать с помощью операции <lock>.

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

Если поддерживается возможность :candidate, конфигурацию-кандидат следует заблокировать.

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

E.1.2. Контрольные точки рабочей конфигурации

Рабочую конфигурацию можно сохранить в локальном файле в качестве контрольной точки перед загрузкой новой конфигурации. Если обновление будет неудачным, конфигурацию можно будет восстановить из этого файла.

Контрольную точку можно создать с помощью операции <copy-config>.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <copy-config>
    <target>
      <url>file://checkpoint.conf</url>
    </target>
    <source>
      <running/>
    </source>
  </copy-config>
</rpc>
```

Для восстановления конфигурации служит та же операция со сменой мест параметров <source> и <target>.

E.1.3. Загрузка и проверка пригодности конфигурации

Если поддерживается возможность :candidate, конфигурация может быть загружена в устройство без воздействия на работающую систему.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <!-- здесь указываются изменения конфигурации -->
    </config>
  </edit-config>
</rpc>
```

Если устройство поддерживает возможность :validate:1.1, оно будет по умолчанию проверять пригодность входящей конфигурации при загрузке в хранилище-кандидат. Для отключения такой проверки служит параметр <test-option> со значением set. Полную проверку можно запросить с помощью операции <validate>.

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

E.1.4. Изменение рабочей конфигурации

Когда входящая конфигурация была загружена в устройство и проверена, её можно активизировать в системе.

Если устройство поддерживает возможность :candidate, операция <commit> копирует информацию из хранилища-кандидата в рабочую конфигурацию. Параметр <confirmed> позволяет автоматически восстановить исходную конфигурацию, если связь с устройством будет потеряна.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <confirm-timeout>120</confirm-timeout>
  </commit>
</rpc>
```

Если конфигурация-кандидат не поддерживается устройством, входящая конфигурация загружается непосредственно в рабочее хранилище.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <!-- здесь указываются изменения конфигурации -->
    </config>
  </edit-config>
</rpc>
```

E.1.5. Тестирование новой конфигурации

Когда входящая конфигурация передана в рабочую, приложению нужно убедиться в отсутствии негативного влияния на работу устройства.

Для обретения такой уверенности приложение может воспользоваться тестами рабочего состояния устройства. Природа таких тестов зависит от внесённых изменений и выходит за рамки документа. Тесты могут включать проверку доступности из системы, где работает приложение (ping), изменений в доступности остальной сети (сравнение таблиц маршрутизации) или инспектирование отдельных изменений (например, обнаружение сессии с добавленным партнёром BGP).

E.1.6. Сохранение изменений

Когда изменения в конфигурацию внесены и приложение имеет достаточную уверенность в корректности работы устройства, внесённые изменения можно зафиксировать (сохранить).

Если устройство поддерживает возможность :startup, текущая конфигурация может быть сохранена путём указания стартового хранилища в качестве цели операции <copy-config>.

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

Если устройство поддерживает возможность :candidate и было запрошено подтверждаемое представление, должно произойти подтверждённое представление конфигурации до завершения срок ожидания.

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

E.1.7. Снятие блокировки конфигурации

Когда обновление конфигурации завершено, блокировка должна быть снята, чтобы другие приложения могли получить доступ к конфигурации.

Для снятия блокировки служит операция <unlock>.

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

Если поддерживается возможность :candidate, следует разблокировать конфигурацию-кандидат.

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

E.2. Операции со множеством устройств

Когда для изменения конфигурации требуется обновление множества устройство, следует соблюдать осторожность при выборе семантики транзакций. Протокол NETCONF содержит достаточно примитивов для организации ориентированных на транзакции операций. Обеспечение полной семантики транзакций на многочисленных устройствах чрезмерно дорого, но размер и число окон для случаев отказа можно снизить.

Имеется два класса операций с множеством устройств. Первый класс разрешает отказы на отдельных устройствах без возврата конфигурации остальных устройств в исходное состояние. Операция может быть повторена позднее или об отказе может быть просто сообщено пользователю. Примером такого класса операций может служить добавление сервера NTP. Для операций этого класса предотвращение отказов и восстановление сфокусировано на отдельных устройствах. Это предполагает восстановление устройства, отчёт об отказе и возможно повторную попытку.

Второй класс более интересен и требует завершения операций на всех устройствах или полного возврата к прежней конфигурации. Т. е. сеть должна целиком перейти в новое состояние или восстановить прежнее. Например, переход на VPN может потребовать обновления множества устройств. Другим примером может служить определение нового класса обслуживания. Если в сети будет обновлена лишь часть устройств, это приведёт к проблемам, когда начнётся применение определения нового класса.

Для обеспечения семантики транзакций выполняются те же операции, которые были описаны выше для отдельного устройства, но они происходят параллельно на всех устройствах. Блокировку конфигурации следует активизировать на всех целевых устройствах и сохранять до тех пор, пока все устройства не будут полностью обновлены с сохранением их новой конфигурации. Изменения конфигурации следует загружать и проверять на всех устройствах. Контрольные точки также следует делать для каждого устройства. После этого рабочая конфигурация меняется, тестируется и сохраняется. Если на любом из этапов возникает отказ, конфигурация может быть восстановлена на любом устройстве, где она была изменена. После полной реализации (или полного отказа) блокировку можно снять на всех устройствах.

Приложение F. Отличия от RFC 4741

В этом приложении отмечены основные отличия данного документа от RFC 4741.

- Добавлен тег ошибки malformed-message.
- Добавлено перечисляемое значение remove для атрибута operation.
- Отменен тег ошибки partial-operation.
- Добавлены параметры <persist> и <persist-id> для операции <commit>.
- Обновлён идентификатор URI базового протокола и уточнён обмен сообщениями <hello> для выбора и идентификации версии базового протокола, используемого в конкретной сессии.
- Добавлен модуль YANG для моделирования операций и удалён уровень операций из XSD.
- Уточнено поведение блокировки для хранилища-кандидата.
- Уточнены требования к откликам сервера на ошибки для значения delete в атрибуте operation.
- Добавлен механизм шаблонов пространства имён для фильтрации ветвей (subtree).
- Добавлено значение test-only для параметра <test-option> операции <edit-config>.
- Добавлена операция <cancel-commit>.
- Введено имя пользователя NETCONF и требования для транспортных протоколов, разъясняющие вывод имени пользователя.

Адреса авторов

Rob Enns (редактор)
Juniper Networks
EMail: rob.enns@gmail.com

Martin Bjorklund (редактор)
Tail-f Systems
EMail: mbj@tail-f.com

Juergen Schoenwaelder (редактор)
Jacobs University
EMail: j.schoenwaelder@jacobs-university.de

Andy Bierman (редактор)
Brocade
EMail: andy.bierman@brocade.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru