Internet Engineering Task Force (IETF)
Request for Comments: 8040
Category: Standards Track
ISSN: 2070-1721

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
K. Watsen
Juniper Networks

January 2017

RESTCONF Protocol

Протокол RESTCONF

Аннотация

Этот документ описывает основанный на HTTP протокол, который обеспечивает программируемый интерфейс для доступа к данным, определенным в YANG и использующим концепцию хранилищ данных NETCONF[1].

Статус документа

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[2] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[3]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 7841.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc8040.

Оглавление

[1]Network Configuration Protocol - протокол настройки конфигурации сети.
[2]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[3]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Имеется необходимость в стандартных механизмах, обеспечивающих Web-приложениям возможность доступа к данным конфигурации и состояния, определяемым моделью данных операциям RPC[1] и уведомлениям о событиях в сетевых устройствах, с модульной структурой и возможностями расширения.

В этом документе определён основанный на HTTP [RFC7230] протокол RESTCONF, для работы с конфигурационными данными YANG версии 1 [RFC6020] или 1.1 [RFC7950] с использованием концепций протокола NETCONF[2] [RFC6241].

NETCONF определяет хранилища конфигурации и набор операций для создания (Create), считывания (Read), обновления (Update) и удаления (Delete), сокращённо обозначаемых CRUD[3], которые могут применяться для доступа к этим хранилищам. NETCONF также определяет протокол для вызова этих операций. Язык YANG определяет синтаксис и семантику содержимого хранилищ, конфигурации, данных состояния, операций RPC и уведомлений.

RESTCONF использует методы HTTP для выполнения операций CRUD на концептуальном хранилище данных YANG, которые совместимы с серверами, реализующими хранилища NETCONF.

Если сервер RESTCONF совмещён с сервером NETCONF, возникают протокольные взаимодействия с NETCONF, описанные в параграфе 1.4. Сервер RESTCONF **может** обеспечивать доступ к конкретным хранилищам, используя ресурсы операций, как описано в параграфе 3.6. Протокол RESTCONF не задаёт каких-либо обязательных ресурсов операций. Семантика каждой операции определяет возможность и способ доступа к хранилищам.

Данные конфигурации и состояния раскрываются как ресурсы, доступные с помощью метода GET. Ресурсы, представляющие данные конфигурации, можно изменить с помощью методов DELETE, PATCH, POST, PUT. Данные кодируются с использованием XML [W3C.REC-xml-20081126] или JSON [RFC7159].

Связанные с моделью данных операции RPC, определённые в операторах YANG rpc или action, могут быть вызваны с помощью метода POST. Доступны уведомления о связанных с моделью данных событиях, определённых в операторах YANG notification.

1.1. Терминология

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

1.1.1. NETCONF

Ниже перечислены термины, определённые в [RFC6241].

- candidate configuration datastore - хранилище конфигурации-кандидата;
- configuration data - данные конфигурации;
- datastore - хранилище данных;
- configuration datastore - хранилище конфигурации;
- running configuration datastore - хранилище рабочей конфигурации;
- startup configuration datastore - хранилище стартовой конфигурации;
- state data - данные состояния;
- user - пользователь.

1.1.2. HTTP

Ниже перечислены термины, определённые в [RFC3986].

[1]Remote Procedure Call - вызов удалённой процедуры.
[2]Network Configuration Protocol - протокол настройки конфигурации сети.
[3]Create, Read, Update, Delete.

- fragment - фрагмент;
- path - путь;
- query - запрос.

Ниже перечислены термины, определённые в [RFC7230].

- header field - поле заголовка;
- message-body - тело сообщения;
- request-line - строка запроса;
- request URI - идентификатор запроса;
- status-line - строка состояния.

Ниже перечислены термины, определённые в [RFC7231].

- method - метод;
- request - запрос;
- resource - ресурс.

Ниже приведён термин, определённый в [RFC7232].

- entity-tag - тег объекта (сущности, элемента).

1.1.3. YANG

Ниже перечислены термины, определённые в [RFC7950].

- action - действие;
- container - контейнер;
- data node - узел данных;
- key leaf - ключевой лист;
- leaf - лист (дерева);
- leaf-list - лист-список;
- list - список;
- mandatory node - обязательный узел;
- ordered-by user - упорядочение пользователем;
- presence container - контейнер присутствия;
- RPC operation - операция RPC;
- top-level data node - узел данных верхнего уровня.

1.1.4. Уведомления NETCONF

Ниже приведён термин, определённый в [RFC5277].

- notification replay - повторное использование уведомления.

1.1.5. Определения терминов

Ниже приведены определения используемых в этом документе терминов.

API resource - ресурс API
Ресурс, моделирующий корневой ресурс RESTCONF и субресурсы для доступа к определённому в YANG содержимому. Определяется с помощью шаблона данных YANG yang-api в модуле ietf-restconf.

client - клиент
Клиент RESTCONF.

data resource - ресурс данных
Ресурс, моделирующий узел данных YANG. Определяется операторами определений YANG.

datastore resource - ресурс хранилища данных
Ресурс, который моделирует программируемый интерфейс использования хранилища данных в стиле NETCONF. По умолчанию методы RESTCONF используют унифицированное представление реализации хранилища данных на сервере. Оно определяется как субресурс внутри ресурса API.

edit operation - операция редактирования
Операция RESTCONF над ресурсом данных с использованием метода POST, PUT, PATCH или DELETE. Это отличается от операций редактирования NETCONF (т. е. одного из значений атрибута nc:operation - create, replace, merge, delete или remove).

event stream resource - ресурс потока событий
Ресурс, представляющий поток событий SSE[1]. Содержимое представляет собой текст, использующий тип носителя text/event-stream, как определено в спецификации SSE [W3C.REC-eventsource-20150203]. Содержимое потока событий описано в параграфе 3.8.

[1]Server-Sent Events - передаваемые сервером события.

media type - тип носителя (среды)
HTTP использует типы носителей Internet (media type) [RFC2046] в полях заголовка Content-Type и Accept для указания и согласования типа.

NETCONF client - клиент NETCONF
Клиент, реализующий протокол NETCONF. В [RFC6241] называется просто клиентом.

NETCONF server - сервер NETCONF
Сервер, реализующий протокол NETCONF. В [RFC6241] называется просто сервером.

operation - операция
Концептуальная операция RESTCONF для сообщения с использованием метода HTTP, URI запроса, полей заголовка и тела сообщения (message-body).

operation resource - ресурс операции
Ресурс, моделирующий определяемую моделью данных операцию, которая в свою очередь определяется с помощью оператора YANG rpc или action. Вызывается с помощью метода POST.

patch - исправления, заплатка
Метод PATCH для целевого хранилища или ресурса данных. Тип носителя для содержимого message-body будет указывать используемый тип patch.

plain patch - простое исправление
Конкретный тип носителя (media type) для использования в методе PATCH (параграф 4.6.1). Этот тип может применяться для простой операции слияния (merge). Тип указывается в запросе полем Content-Type со значением application/yang-data+xml или application/yang-data+json.

query parameter - параметр запроса
Параметр (и его значение при наличии), закодированный в компоненте query идентификатора URI для запроса.

resource type - тип ресурса
Один из классов ресурсов RESTCONF, определённых в этом документе, - api, datastore, data, operation, schema или event stream.

RESTCONF capability - возможность (свойство) RESTCONF
Необязательная функция протокола RESTCONF, которая анонсируется конкретным сервером в случае её поддержки. Возможности указываются зарегистрированными IANA идентификаторами NETCONF Capability URI и анонсируются записью в листе-списке capability, определённом в параграфе 9.3.

RESTCONF client - клиент RESTCONF
Клиент, реализующий протокол RESTCONF.

RESTCONF server - сервер RESTCONF
Сервер, реализующий протокол RESTCONF.

retrieval request - запрос на извлечение
Запрос с использованием метода GET или HEAD.

schema resource - ресурс схемы
Ресурс, используемый клиентом для получения схемы YANG с помощью метода GET. Представляется типом носителя application/yang.

server - сервер
Сервер RESTCONF.

"stream" list - список “потоков”
Набор экземпляров ресурсов данных, который описывает ресурсы потоков событий, доступные на сервер. Эта информация определяется в модуле ietf-restconf-monitoring как список stream и может быть получена с использованием целевого ресурса {+restconf}/data/ietf-restconf-monitoring:restconf-state/streams/stream. Список stream содержит информацию о каждом потоке, такую как URL для извлечения данных потока событий.

stream resource - ресурс потока
Ресурс потока событий.

target resource - целевой ресурс
Ресурс, связанный с отдельным сообщением, указываемый компонентом path в URI запроса.

yang-data extension - расширение yang-data
Внешний оператор YANG, соответствующий оператору расширения yang-data, описанному в разделе 8. Расширение yang-data служит для определения структур данных YANG, которые предполагается применять в качестве шаблонов YANG. Эти структуры данных не предназначены для реализации в конфигурационной хранилище или операционном состоянии на сервере, поэтому в них не могут применяться обычные операторы определения данных YANG.

YANG data template - шаблон данных YANG
Схема для компонентов сообщения протокола моделирования как концептуальных структур данных, использующих YANG. Это позволяет определять сообщения независимо от их кодирования. Каждый шаблон данных YANG определяется с помощью расширения yang-data, описанного в разделе 8. Для YANG могут быть определены представления экземпляров, соответствующих определённому шаблону данных YANG. Представление XML определено в YANG версии 1.1 [RFC7950] и поддерживается с типом носителя application/yang-data+xml. Представление JSON определено в документе «JSON Encoding of Data Modeled with YANG» [RFC7951] и поддерживается с типом носителя application/yang-data+json.

1.1.6. Шаблон и примеры URI

В этом документе применяется синтаксис шаблона URI [RFC6570] {+restconf} для ссылок на корневой ресурс RESTCONF за пределами примера. Подробности приведены в параграфе 3.1.

Для простоты все примеры в документе используют /restconf в качестве «найденного» пути к корню RESTCONF API. Многие из приведённых в документе примеров основаны на модуле YANG example-jukebox, определённом в Приложении A.1.

Многие строки заголовков протокола и текста в message-body приведённых в документе примеров разбиты на несколько строк для наглядности. Строки, заканчивающиеся символом \ продолжаются следующей строкой документа. Т. е. строки объединяются с удалением символа \, за которым следует перевод строки и пробельных символов в начале следующей строки.

1.1.7. Диаграммы деревьев

В документе применяется упрощённое графическое представление моделей данных. Назначение символов описано ниже.

- Ключи списков указываются в квадратных скобках [].
- Сокращения перед именами узлов задают режим доступа - rw для данных конфигурации (чтение и запись), ro для данных состояния (только чтение).
- Символ ? после узла данных указывает необязательный узел, ! - контейнер присутствия, * - список или лист-список.
- В круглых скобках указываются узлы выбора (choice) и вариантов (case), последние также маркируются двоеточием (:).
- Троеточие (...) указывает пропущенное содержимое субдерева.

1.2. Подмножество функций NETCONF

В RESTCONF не требуется поддержка всей функциональности протокола NETCONF, но нужна совместимость с NETCONF. Протокол RESTCONF обеспечивает это за счёт реализации подмножества возможностей взаимодействия, обеспечиваемых NETCONF. Например, не поддерживаются некоторые хранилища и явные блокировки.

RESTCONF использует методы HTTP для реализации эквивалентов операций NETCONF, что позволяет выполнять базовые операции CRUD на иерархии концептуальных ресурсов.

Методы HTTP POST, PUT, PATCH и DELETE служат для редактирования ресурсов данных, представленных моделями YANG. Эти базовые операции редактирования позволяют менять рабочую конфигурацию из клиентов RESTCONF.

Протокол RESTCONF не предназначен для замены NETCONF и просто обеспечивает интерфейс HTTP, следующий принципам REST[1] [REST-Dissertation] и совместимый с моделью хранилища данных NETCONF.

1.3. Управляемый моделью данных интерфейс API

RESTCONF объединяет простоту HTTP с предсказуемостью и возможностями автоматизации управляемых схемами API. Зная модули YANG, используемый сервером, клиент может вывести URL всех ресурсов управления, а также подходящую структуру всех запросов и откликов RESTCONF. Такая стратегия избавляет от необходимости включать в отклики сервера ссылки HATEOAS[2], впервые описанные в докторской диссертации Roy Fielding [REST-Dissertation], поскольку клиент может определить нужные ему ссылки из модулей YANG.

RESTCONF использует библиотеку YANG [RFC7895] для предоставления клиентам возможности найти информацию л соответствии модуля YANG для на сервере, если клиент хочет применять её.

Сервер может поддерживать извлечение используемых им модулей YANG из своей библиотеки. Подробности этого описаны в параграфе 3.7.

Идентификаторы URI для зависимых от модели данных операций RPC и содержимого хранилищ предсказуемы на основе определений в модуле YANG.

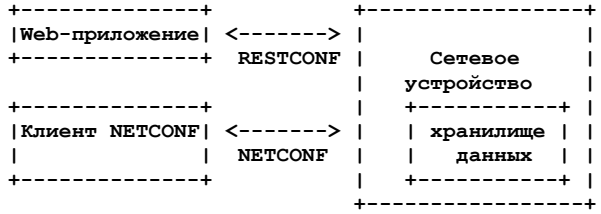
RESTCONF работает с концептуальным хранилищем данных, определенным на языке моделирования YANG. Сервер перечисляет все поддерживаемые модули YANG с использованием модуля ietf-yang-library, определённого в [RFC7895]. Сервер **должен** реализовать модуль ietf-yang-library, который **должен** указывать все используемые сервером модули YANG в списке modules-state/module. Содержимое концептуального хранилища данных, зависимые от модели операции RPC и уведомления о событиях тоже указываются этим набором модулей YANG.

Классификация данных как относящихся или не относящихся к конфигурационным выводится из оператора YANG config. Поведение, связанное с упорядочиванием данных, определяется оператором YANG ordered-by. Не относящиеся к конфигурации данные называют также данными состояния.

Модель редактирования хранилища данных в RESTCONF проста и прямолинейна, подобно поведению возможности :writable-running в NETCONF. Каждое редактирование RESTCONF ресурса данных в ресурсе хранилища активируется после успешного завершения редактирования.

1.4. Сосуществование с NETCONF

RESTCONF можно реализовать на устройстве, поддерживающем протокол NETCONF.

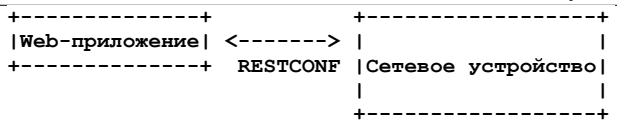


На рисунке представлены системные компоненты RESTCONF размещаемые на одном сервере с NETCONF.

Ниже показаны системные компоненты при реализации RESTCONF на устройстве, не являющемся сервером NETCONF.

Существуют взаимодействия между серверами NETCONF и RESTCONF, связанные с операциями редактирования. Возможно использование блокировок на сервере RESTCONF, хотя протокол RESTCONF и не может управлять блокировками. В таком случае протокол RESTCONF не получит доступа для записи ресурсов данных в хранилище.

[1]Representational State Transfer - перенос репрезентативного состояния.
[2]Hypermedia as the Engine of Application State - гиперсреда как машина состояний приложения.



Если сервер NETCONF поддерживает возможность :writable-running, все операции по редактированию конфигурационных узлов {+restconf}/data выполняются в хранилище рабочей конфигурации. Шаблон URI {+restconf} определён в параграфе 1.1.6.

В противном случае, если устройство поддерживает хранилище :candidate, все операции редактирования конфигурационных узлов {+restconf}/data будут применяться к содержимому хранилища-кандидата. Конфигурация-кандидат **должна** автоматически фиксироваться сразу после успешного редактирования. Все операции редактирования хранилища-кандидата также должны фиксироваться. Если любым клиентом NETCONF выполняется процедура подтверждаемой фиксации (confirmed commit), любая новая фиксация будет служить подтверждающей (confirming commit). Если сервер NETCONF ждёт параметр persist-id для завершения процедуры подтверждаемой фиксации, операция редактирования RESTCONF **должна** приводить к отказу с возвратом кода «409 Conflict». В таком случае возвращается тег ошибки in-use.

Если сервер NETCONF поддерживает :startup, сервер RESTCONF **должен** автоматически обновить энергонезависимое хранилище стартовой конфигурации после изменения рабочего (running) хранилища данных операцией редактирования RESTCONF.

Если на хранилище, изменяемом операцией редактирования RESTCONF, имеется активная блокировка от клиента NETCONF, операция редактирования RESTCONF **должна** приводить к отказу с кодом «409 Conflict». В таком случае возвращается тег ошибки in-use.

1.5. Расширяемость RESTCONF

В RESTCONF имеется два встроенных механизма расширяемости:

- версия протокола;
- дополнительные возможности.

Этот документ определяет версию 1 протокола RESTCONF. При определении в будущем новой версии протокола спецификация будет указывать способ идентификации новой версии RESTCONF. Предполагается, что будет использован другой корневой ресурс RESTCONF, который будет иметь иное размещение (параграф 3.1).

Сервер будет анонсировать все поддерживаемые им версии протокола в своих данных host-meta.

В приведённом ниже примере сервер поддерживает RESTCONF версии 1 и фиктивной версии 2. Клиент может передать

```
GET /.well-known/host-meta HTTP/1.1
Host: example.com
Accept: application/xrd+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Content-Type: application/xrd+xml
Content-Length: nnn

<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
    <Link rel='restconf' href='/restconf'/>
    <Link rel='restconf2' href='/restconf2'/>
</XRD>
```

RESTCONF также поддерживает определяемый сервером список дополнительных возможностей, которые перечисляются сервером с использованием модуля ietf-restconf-monitoring, определённого в параграфе 9.3. Этот документ определяет несколько параметров запроса в параграфе 4.8. Каждый необязательный параметр имеет соответствующий идентификатор URI для возможности, определённый в параграфе 9.1.1, который анонсируется сервером в случае поддержки возможности.

Лист-список capability может указывать любые типы серверных расширений. В настоящее время этот механизм расширения служит для указания поддерживаемых необязательных параметров запроса, но не ограничивается решением этой задачи. Например, идентификатор URI defaults, описанный в параграфе 9.1.2, задаёт обязательное значение URI, указывающего принятое сервером по умолчанию поведение при обработке.

С возможностью может указываться новый тип субресурса, если это не обязательно для реализации. Обязательные возможности протокола и новые типы ресурсов потребуют нового выпуска протокола RESTCONF.

2. Транспортный протокол

2.1. Целостность и конфиденциальность

HTTP [RFC7230] является протоколом прикладного уровня, который может работать на основе любого транспорта с гарантированной доставкой. Протокол RESTCONF работает на основе HTTP, но конфиденциальная природа передаваемой информации требует для RESTCONF транспортного протокола с защитой целостности и конфиденциальности данных. Сервер RESTCONF **должен** поддерживать протокол TLS[1] [RFC5246] и придерживаться документа [RFC7525]. Протокол RESTCONF **недопустимо** использовать с HTTP без поддержки протокола TLS.

RESTCONF не требует определённой версии HTTP, однако **рекомендуется** поддерживать версию не ниже HTTP/1.1 [RFC7230] во всех реализациях.

[1]Transport Layer Security - защита на транспортном уровне.

2.2. HTTPS с сертификатами X.509v3

Учитывая повсеместное сегодня применение HTTP на основе TLS [RFC7230], реализации RESTCONF **должны** поддерживать схему URI https, для которой агентство IANA назначило используемый по умолчанию порт 443.

Серверы RESTCONF **должны** предоставлять сертификат на основе X.509v3 при организации соединения TLS с клиентом RESTCONF. Использование сертификатов X.509v3 согласуется с работой NETCONF на базе TLS [RFC7589].

2.3. Проверка сертификата

Клиент RESTCONF **должен** (1) использовать путь проверки сертификата X.509 [RFC5280] для контроля целостности сертификата TLS на сервере RESTCONF или (2) сравнивать полученный от сервера сертификат TLS с сертификатом, доставленным с помощью доверенного механизма (например, указанный на устройстве сертификат). Если путь проверки сертификата X.509 даёт отказ и представленный сертификат X.509 не соответствует полученному с помощью доверенного механизма, соединение **должно** быть прервано, как описано в параграфе 7.2.1 [RFC5246].

2.4. Отождествление аутентифицированного сервера

Клиент RESTCONF **должен** проверить отождествление сервера в соответствии с параграфом 3.1 [RFC2818].

2.5. Отождествление аутентифицированного клиента

Сервер RESTCONF **должен** аутентифицировать доступ клиента к любому защищённому ресурсу. Если клиент RESTCONF не прошёл проверку подлинности, серверу **следует** передать отклик HTTP с кодом «401 Unauthorized», как указано в параграфе 3.1 [RFC7235]. В таких случаях применяется тег ошибки access-denied.

Для проверки подлинности клиента серверу RESTCONF **следует** требовать аутентификацию на основе сертификата клиента TLS (параграф 7.4.6 [RFC5246]). Если аутентификация на основе сертификата не подходит (например, при невозможности организовать PKI для клиентов), **можно** воспользоваться аутентификацией HTTP. В этом случае **должна** применяться одна из схем аутентификации HTTP, определённых в реестре «Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry» (параграф 5.1 [RFC7235]).

Сервер **может** также поддерживать аутентификацию по сертификатам клиентов и аутентификацию HTTP, решение вопроса использования такой комбинации остаётся за реализацией.

Отождествление клиента RESTCONF, полученное от механизма аутентификации, далее будет называться именем пользователя RESTCONF и к нему применяется модель контроля доступа NACM[1] [RFC6536]. Когда предоставляется сертификат клиента, имя пользователя RESTCONF **должно** выводиться с использованием механизма, определённого в разделе 7 [RFC7589]. Для остальных случаев, когда применяется аутентификация HTTP, имя пользователя RESTCONF **должно** предоставляться используемой схемой аутентификации HTTP.

3. Ресурсы

Протокол RESTCONF работает с иерархией ресурсов, начиная с ресурса API верхнего уровня (параграф 3.1). Каждый из ресурсов представляет управляемый компонент устройства.

Ресурс можно рассматривать как набор данных и методов, разрешённых для этих данных, который может содержать вложенные дочерние ресурсы. Типы дочерних ресурсов и разрешённые для них методы зависят от модели данных.

Ресурс имеет представление, связанное с типом носителя, указанным полем заголовка Content-Type в отклике HTTP. Ресурс имеет одно или множество представления, каждое из которых связано с разным типом носителя. Когда представление ресурса передаётся в сообщении HTTP, связанный тип носителя указывается в заголовке Content-Type. Ресурс может (не обязательно) включать вложенные ресурсы. Ресурсы могут быть созданы или удалены независимо от родительского ресурса, пока тот сохраняется.

Доступ к ресурсам RESTCONF выполняется через набор идентификаторов URI, определённых в этом документе. Набор поддерживаемых сервером модулей YANG будет определять зависимые от модели данных операции RPC, узлы верхнего уровня и уведомления о событиях, поддерживаемые сервером.

Протокол RESTCONF не включает механизма обнаружения ресурсов данных. Вместо этого применяются анонсы сервера с определениями модулей YANG для создания операций RPC или идентификаторов ресурсов.

3.1. Обнаружение корневого ресурса

В соответствии с рекомендациями [RFC7320] протокол RESTCONF разрешает развёрнутым системам указывать местоположение RESTCONF API. При первом подключении к серверу RESTCONF клиент RESTCONF **должен** определить корень RESTCONF API. **Должна** быть в точности одна привязка restconf, возвращённая устройством.

Клиент находит это, получая ресурс /.well-known/host-meta ([RFC6415]) и используя элемент <Link>, содержащий атрибут restconf.

Например, для получения /restconf клиент может передать

```
GET /.well-known/host-meta HTTP/1.1
Host: example.com
Accept: application/xrd+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Content-Type: application/xrd+xml
Content-Length: nnn

<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
    <Link rel='restconf' href='/restconf'/>
```

[1] NETCONF Access Control Model.

```
</XRD>
```

После нахождения корня RESTCONF API клиент **должен** применять это значение в качестве начальной части пути в запросе URI при всех последующих запросах ресурсов RESTCONF.

В этом примере клиент будет применять путь /restconf в качестве корневого ресурса RESTCONF.

Для получения /top/restconf клиент может передать

```
GET /.well-known/host-meta HTTP/1.1
Host: example.com
Accept: application/xrd+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Content-Type: application/xrd+xml
Content-Length: nnn

<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
    <Link rel='restconf' href='/top/restconf'/>
</XRD>
```

В этом примере клиент будет применять путь /top/restconf как корневой ресурс RESTCONF.

Клиент может теперь определить ресурсы операций, поддерживаемые сервером. В приведённом ниже примере сервер поддерживает операцию play. Клиент может передать

```
GET /top/restconf/operations HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Last-Modified: Thu, 26 Jan 2017 16:00:14 GMT
Content-Type: application/yang-data+json

{ "operations" : { "example-jukebox:play" : [null] } }
```

Если дескриптор расширяемого ресурса XRD[1] содержит более одной привязки, к данной спецификации будет относиться лишь привязка, названная restconf.

Отметим, что любая конкретная конечная точка (host:port) может поддерживать лишь один сервер RESTCONF, что обусловлено механизмом обнаружения корневого ресурса. Это ограничивает число серверов RESTCONF, которые могут одновременно работать на хосте, поскольку каждый сервер должен использовать свой порт.

3.2. Типы носителей RESTCONF

Протокол RESTCONF определяет два зависимых от приложения типа носителя для указания представлений данных, которые соответствуют схеме для определённой конструкции YANG.

В этом документе определены два типа носителя для представления данных YANG в XML и JSON. Другие документы могут определяет иные типы носителей для представления данных YANG. Тип application/yang-data+xml определён в параграфе 11.3.1, application/yang-data+json - в параграфе 11.3.2.

3.3. Ресурс API

Ресурс API содержит корневой ресурс RESTCONF для хранилища данных RESTCONF и ресурсов операций. Он является ресурсом верхнего уровня, расположенным в {+restconf} и имеет тип носителя «application/yang-data+xml» или «application/yang-data+json».

Ниже представлена диаграмма дерева YANG для ресурса API.

```
+---- {+restconf}
      +---- data
      | ...
      +---- operations?
      | ...
      +--ro yang-library-version    string
```

Шаблон данных YANG yang-api определён с использованием расширения yang-data из модуля ietf-restconf, представленного в разделе 8. Он задаёт структуру и синтаксис концептуальных дочерних ресурсов в ресурсе API.

Ресурс API можно извлечь с помощью метода GET.

Имя корневого ресурса {+restconf} используется в откликах, представляющих корень модуля ietf-restconf, **должно** указывать модуль YANG ietf-restconf. Например, запрос GET для корневого ресурса /restconf в формате JSON будет возвращать представления ресурса API с именем ietf-restconf:restconf.

Потомки этого ресурса перечислены в таблице.

Ресурс RESTCONF API.

Дочерний ресурс	***Описание***
data	Содержит все ресурсы данных
operations	Определяемые моделью данных операции
yang-library-version	Дата модуля ietf-yang-library

[1]Extensible Resource Descriptor.

3.3.1. Ресурс {+restconf}/data

Этот обязательный ресурс представляет комбинацию ресурсов данных конфигурации и состояния, к которым клиент может получить доступ. Клиент не может создавать или удалять эти ресурсы. Ресурс хранилища данных определён в параграфе 3.4.

В приведённом ниже примере запрос клиента с использованием параметра content (параграф 4.8.1) будет возвращать лишь не относящиеся к конфигурации узла данных, имеющиеся в ресурсе library.

```
GET /restconf/data/example-jukebox:jukebox/library\
    ?content=nonconfig HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+xml

<library xmlns="https://example.com/ns/example-jukebox">
  <artist-count>42</artist-count>
  <album-count>59</album-count>
  <song-count>374</song-count>
</library>
```

3.3.2. Ресурс {+restconf}/operations

Этот необязательный ресурс является контейнером, обеспечивающим доступ к определяемым моделью данных операциям RPC, поддерживаемым сервером. Сервер **может** опускать этот ресурс, если он не анонсирует связанных с моделью операций RPC.

Все связанные с моделью операции RPC, определённые в модулях YANG, которые сервер анонсирует **должны** быть доступны как дочерние узлы этого ресурса.

Точка доступа для каждой операции RPC представляется как пустой лист. Если ресурс операции найден, сервер возвращает представление пустого листа.

Ресурсы операций определены в параграфе 3.6.

3.3.3. Лист {+restconf}/yang-library-version

Этот обязательный лист указывает дату выпуска модуля YANG ietf-yang-library, реализованного этим сервером. В приведённом ниже примере используется дата выпуска модуля из [RFC7895].

```
GET /restconf/yang-library-version HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+xml

<yang-library-version
  xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">\
  2016-06-21</yang-library-version>
```

3.4. Ресурс хранилища данных

Субдерево {+restconf}/data представляет ресурс хранилища данных, которое содержит набор узлов данных конфигурации и состояния.

Этот тип ресурса представляет собой абстракцию реализации базового хранилища данных в системе. Клиент применяет его для редактирования и извлечения ресурсов данных как концептуальный корень для всех данных конфигурации и состояния, имеющихся в устройстве.

Управление транзакциями редактирования и сохранения конфигурации обслуживается сервером и клиент не контролирует этого. Запись в ресурс хранилища данных возможна напрямую с использованием методов POST и PATCH. Каждое редактирование RESTCONF сохраняется в энергонезависимой памяти ресурса хранилища, если сервер поддерживает такую память (параграф 1.4).

Если ресурс хранилища данных, представленный субдеревом {+restconf}/data, найден, хранилище и его содержимое возвращается сервером. Хранилище данных представляется узлом data в пространстве имён модуля ietf-restconf.

3.4.1. Предотвращение конфликтов при редактировании

В RESTCONF поддерживается два механизма детектирования и предотвращения конфликтов при одновременном редактировании - временная метка (timestamp) и тег объекта (entity-tag).При любом изменении ресурсов конфигурационных данных обновляются значения timestamp и entity-tag для ресурса хранилища. В дополнение к этому сервер RESTCONF **должен** возвращать ошибку, если хранилище заблокировано из другого источника (например, сервером NETCONF).

3.4.1.1. Временная метка

Поддерживается время последнего изменения и возвращается поле заголовка Last-Modified (параграф 2.2 [RFC7232]) в ответах на запросы извлечения. Поле заголовка If-Unmodified-Since (параграф 3.4 of [RFC7232]) может использоваться в запросах на операции редактирования для того, чтобы принудить сервер отвергать запрос на редактирование, если ресурс был изменён после указанного временной меткой момента.

Серверу **следует** поддерживать временную метку момента последнего изменения ресурса хранилища данных, определённого в параграфе 3.4. На эту метку оказывает влияние лишь изменение дочерних ресурсов конфигурационных данных и **недопустимо** устанавливать метку в ответ на изменение не связанных с конфигурацией данных. Временные метки для ресурсов данных рассматриваются в параграфе 3.5.

Если сервер RESTCONF совмещён с сервером NETCONF, временная метка последнего обновления **должна** относиться к рабочему (running) хранилищу. Отметим, что метка может меняться и в результате действий других протоколов, но этот вопрос выходит за рамки документа.

3.4.1.2. Тег объекта

Сервер **должен** поддерживать уникальный тег объекта (entity-tag) для ресурса хранилища и **должен** возвращать этот тег в поле заголовка ETag (параграф 2.3 [RFC7232]) отклика на запрос извлечения. Клиент **может** использовать заголовок If-Match в запросах на операции редактирования, чтобы вынудить сервер отвергать запрос для ресурса, у которого тег объекта не соответствует указанному значению.

Сервер **должен** поддерживать тег объекта для ресурса верхнего уровня {+restconf}/data. На этот тег влияют лишь ресурсы данных конфигурации и его **недопустимо** обновлять при изменении не относящихся к конфигурации данных. Теги объекта для ресурсов данных рассмотрены в параграфе 3.5. Отметим, что для каждого представления (например, XML и JSON) требуется свой тег объекта.

Если сервер RESTCONF совмещён с сервером NETCONF, тег объекта **должен** относиться к рабочему (running) хранилищу. Отметим, что тег может меняться и в результате действий других протоколов, но этот вопрос выходит за рамки документа.

3.4.1.3. Процедура обновления

Изменение ресурса конфигурационных данных влияет на временную метку и тег объекта для этого ресурса, родительского ресурса и ресурса хранилища данных.

Например, редактирование с отключением интерфейса может быть выполнено путём установки значения false для листа /interfaces/interface/enabled. Узел данных enabled и его предки (экземпляр списка interface и контейнер interfaces) также считаются изменёнными. Хранилище считается обновлённым при изменении любого узла конфигурационных данных верхнего уровня (например, interfaces).

3.5. Ресурс данных

Ресурс данных представляет узел данных YANG, являющийся потомком ресурса хранилища данных. Каждый определённый в YANG узел данных может быть однозначно указан строкой запроса в методе HTTP. Узлами данных являются контейнеры, листья, записи листьев-списков и списков, узлы anydata и anyxml.

Представление, поддерживаемое для каждого ресурса данных, является определенным в YANG субдеревом данного узла. Методы HTTP для ресурса данных относятся к целевому узлу и всем его потомкам, если он имеются.

Ресурс данных можно извлечь с помощью метода GET, используя идентификатор URI {+restconf}/data. Это субдерево применяется для извлечения и редактирования ресурсов данных.

3.5.1. Временная метка

Для ресурсов конфигурационных данных сервер **может** поддерживать временную метку последнего изменения ресурса и возвращать поле заголовка Last-Modified при извлечении ресурса методом GET или HEAD.

Полученное значение заголовка Last-Modified может применяться клиентом RESTCONF в последующих запросах в полях заголовка If-Modified-Since и If-Unmodified-Since последующих запросов.

При поддержке временной метки для ресурса в ней **должно** устанавливаться текущее время изменения самого ресурса или любого содержащегося в нем конфигурационного ресурса. Если такие метки не поддерживаются, взамен **должна** применяться временная метка хранилища данных. Если сервер RESTCONF совмещён с сервером NETCONF, временная метка **должна** относиться к экземпляру в рабочем хранилище данных.

На эту метку оказывает влияние лишь изменение конфигурационных данных и **недопустимо** менять метку в ответ на изменение не относящихся к конфигурации данных.

3.5.2. Тег объекта

Для каждого ресурса конфигурационных данных серверу **следует** поддерживать тег объекта и возвращать поле заголовка Etag при извлечении ресурса как цели с помощью метода GET или HEAD. При поддержке таких тегов они **должны** обновляться при изменении ресурса или любого из включённых в него конфигурационных ресурсов. Если такие теги не поддерживаются, взамен **должен** применяться тег объекта для хранилища конфигурации.

Полученное значение заголовка ETag может применяться клиентом RESTCONF в последующих запросах в полях заголовка If-Match и If-None-Match последующих запросов.

На этот тег оказывает влияние лишь изменение конфигурационных данных и **недопустимо** менять тег в ответ на изменение не относящихся к конфигурации данных. Если сервер RESTCONF совмещён с сервером NETCONF, тег объекта **должен** относиться к экземпляру в рабочем хранилище данных.

3.5.3. Кодирование идентификаторов ресурса данных в URI запроса

В YANG узлы данных могут указываться определёнными в [XPath] абсолютными выражениями XPath, которые начинаются с корня документа для целевого ресурса. В RESTCONF применяются закодированные в URI выражения путей.

Предсказуемое местоположение ресурса данных важно, поскольку оно кодируется в модуле модели данных YANG со статическим именованием и абсолютным расположением путей для всех узлов данных.

Идентификаторы ресурсов данных в RESTCONF кодируются слева направо, начиная с узла верхнего уровня, в соответствии с правилом api-path из параграфа 3.5.3.1. Имена узлов каждого предка узла целевого ресурса кодируются по порядку, заканчивая именем узла самого целевого ресурса. Если узел в пути определён в модуле, отличающемся от модуля предка, или предком является хранилище данных, перед именем узла в идентификаторе ресурса **должно** указываться имя модуля, заканчивающееся символом двоеточия (:). Подробности приведены в параграфе 3.5.3.1.

Если узлом данных в выражении пути является лист-список YANG, значение leaf-list **должно** кодироваться в соответствии с приведёнными ниже правилами.

- Идентификатор узла-списка **должен** кодироваться с использованием одного сегмента пути.
- Сегмент пути создаётся из имени узла-списка, за которым следует символ = и значение узла списка (например, /restconf/data/top-leaflist=fred).
- Значение узла-списка указывается строкой с использованием канонического представления для соответствующего типа данных YANG. Для всех зарезервированных символов **должно** применяться %-кодирования в соответствии с параграфами 2.1 и 2.5 [RFC3986].
- YANG 1.1 позволяет дублирование значений leaf-list для данных, не относящихся к конфигурации. Для таких случаев нет механизма точного указания элемента листа-списка.
- Для символа запятой (,) используется %-кодирование [RFC3986] даже в тех случаях, когда для leaf-list множественные значения невозможны. Это улучшается согласованность и избавляет от специальных правил обработки.

Если узел данных в выражении пути является списком YANG, значения ключей списка (если они имеются) **должны** кодироваться в соответствии с приведёнными ниже правилами.

- Значения ключевых листьев для ресурсов данных, представляющих список YANG **должны** кодироваться с использованием одного сегмента пути [RFC3986].
- При наличии единственного ключевого листа сегмент пути создаётся из имени списка, за которым следует символ = и значение ключевого листа.
- При наличии множества ключевых листьев сегмент пути создаётся из имени списка, за которым следует значение каждого ключа, указанного в операторе key в соответствии с порядком в этом операторе. За каждым значением ключа, кроме последнего, следует запятая (,).
- Значение ключа указывается строкой с использованием канонического представления для соответствующего типа данных YANG. Для всех зарезервированных символов **должно** применяться %-кодирования в соответствии с параграфами 2.1 и 2.5 [RFC3986]. Для символа запятой (,) используется %-кодирование [RFC3986], если в списке имеются запятые.
- Кодироваться **должны** все компоненты оператора key. Частичные идентификаторы экземпляров не поддерживаются.
- Пропуск значений ключей не допускается, поэтому две запятые подряд считаются запятой, за которой следует пустая строка и вторая запятая. Например, «list1=foo,,baz» будет интерпретироваться как список list1 с тремя ключами, из который второй является пустым.
- Отметим, что для данных не относящихся к конфигурации, не требуется определять ключи. В таких случаях один экземпляр списка не будет доступен.
- Правило ABNF[1] [RFC5234] list-instance, определённое в параграфе 3.5.3.1, представляет синтаксис идентификатора экземпляра списка.

```
container top {
    list list1 {
        key "key1 key2 key3";
         ...
         list list2 {
             key "key4 key5";
             ...
             leaf X { type string; }
         }
     }
     leaf-list Y {
       type uint32;
     }
}
```

Например, в приведённом выше определении YANG контейнер top указан в модулей example-top, и идентификатор URI целевого ресурса X будет кодироваться в виде

```
/restconf/data/example-top:top/list1=key1,key2,key3/\
   list2=key4,key5/X
```

A URI целевого ресурса для узла-списка Y будет иметь вид

[1] Augmented Backus-Naur Form - расширенная форма Бэкуса-Наура.

```
/restconf/data/example-top:top/Y=instance-value
```

Приведённый ниже пример показывает %-кодирование для зарезервированных символов в значении ключа. Значение key1 в этом примере содержит запятую, одинарную и двойную кавычку, двоеточие, двойную кавычку, пробел и символ обратной дробной черты (","":" /). Отметим, что символы двойных кавычек не являются зарезервированными, и не требуют %-кодирования. Значением key2 является пустая строка, а key3 - строка «foo».

```
/restconf/data/example-top:top/list1=%2C%27"%3A"%20%2F,,foo
```

3.5.3.1. ABNF для идентификаторов ресурсов данных

Синтаксис ABNF [RFC5234] api-path используется для создания идентификаторов пути RESTCONF. Отметим, что этот синтаксис применяется для всех ресурсов и путь API начинается с корневого ресурса RESTCONF. Ресурсы данных должны указываться в субдереве {+restconf}/data.

Идентификаторы недопустимо начинать с последовательности символов XML (в любом регистре) в соответствии с правилами для идентификаторов YANG. Синтаксис api-identifier и key-value **должен** соответствовать правилам кодирования идентификаторов JSON из раздела 4 [RFC7951] - требуется путь к корневому ресурсу RESTCONF. Дополнительные субидентификаторы ресурсов являются не обязательны. Для символов в значениях ключей имеются ограничения и некоторые символы нужно указывать %-кодами, как описано в параграфе 3.5.3.

```
api-path = root *("/" (api-identifier / list-instance))
root = string  ;; замена строки {+restconf}
api-identifier = [module-name ":"] identifier
module-name = identifier
list-instance = api-identifier "=" key-value *("," key-value)
key-value = string  ;; зарезервированные символы указываются ;-кодом
string = <an unquoted string>
identifier = (ALPHA / "_")
             *(ALPHA / DIGIT / "_" / "-" / ".")
```

3.5.4. Принятая по умолчанию обработка

RESTCONF требует, чтобы сервер указывал принятый по умолчанию режим обработки (см. параграф 9.1.2). Если сервер поддерживает необязательный параметр with-defaults, клиент может применять его для управления извлечением принятых по умолчанию значений (параграф 4.8.9).

Если отсутствует лист или лист-список из конфигурации и есть принятое по умолчанию значение в определении YANG для этого ресурса данных, сервер **должен** использовать значение из определения YANG как заданное.

Если целью метода GET является узел данных, который представляет leaf или leaf-list с принятым по умолчанию значением, а экземпляр этого листа или листа-списка ещё не создан, сервер **должен** возвращать принятое по умолчанию значение или значения, используемые сервером. В таких случаях сервер **должен** игнорировать параметр basic-mode, описанный в параграфе 4.8.9, и возвращать принятое по умолчанию значение.

Если целью метода GET является узел данных, который представляет контейнер или список, в котором имеется любой дочерний ресурс с принятым по умолчанию значением, для дочерних ресурсов, у которых ещё нет значений, сервер **может** вернуть используемые им по умолчанию значения в соответствии с указанным режимом обработки по умолчанию и переданными клиентом параметрами запроса.

3.6. Ресурсы операций

Ресурсы операций представляют операции RPC, определённые оператором YANG rpc, или зависящие от модели действия, определённые оператором YANG action. Они вызываются с помощью метода POST.

Операция RPC вызывается в форме

```
POST {+restconf}/operations/<operation>
```

Поле <operation> указывает имя модуля и строку идентификатора rpc для желаемой операции.

Например, если module-A определяет операцию RPC reset, вызов этой операции будет иметь вид

```
POST /restconf/operations/module-A:reset HTTP/1.1
Server: example.com
```

Вызов действия (action) имеет вид

```
POST {+restconf}/data/<data-resource-identifier>/<action>
```

где <data-resource-identifier> указывает путь к узлу данных, в котором определено действие, а <action> - имя действия.

Например, если module-A определяет действие reset-all в контейнере interfaces, вызов действия будет иметь вид

```
POST /restconf/data/module-A:interfaces/reset-all HTTP/1.1
Server: example.com
```

Если вызванная операция RPC не привела к ошибке, а оператор rpc или action не имеет раздела output, в сообщение с откликом **недопустимо** включать message-body, а вместо этого **должна** передаваться строка «204 No Content».

Все ресурсы операций, представляющие операции RPC, которые поддерживаются сервером, **должны** указываться в субдереве {+restconf}/operations, определённом в параграфе 3.3.2. Ресурсы операций, представляющие действия YANG, не указываются в этом субдереве, поскольку они вызываются с использованием URI в субдереве {+restconf}/data.

3.6.1. Кодирование входных параметров ресурса операции

Если оператор rpc или action имеет раздел input, экземпляры входных параметров кодируются с использованием пространства имён модуля, где определён оператор rpc или action, в элемент XML или объект JSON с именем input с использованием пространства имён модуля, где определён оператор rpc или action.

Если оператор rpc или action имеет раздел input и дерево объекта input содержит какие-либо дочерние узлы, которые считаются обязательными, клиент **должен** включить в запрос message-body.

Если оператор rpc или action имеет раздел input и дерево объекта input не содержит дочерних узлов, которые считаются обязательными, клиент **может** не включать message-body в запрос.

Если оператор rpc или action не имеет раздела input, включение message-body в запрос недопустимо.

Приведённый ниже модуль YANG используется здесь в качестве примера операции RPC.

```
module example-ops {
  namespace "https://example.com/ns/example-ops";
  prefix "ops";

  organization "Example, Inc.";
  contact "support at example.com";
  description "Example Operations Data Model Module.";
  revision "2016-07-07" {
    description "Первый выпуск.";
    reference "example.com document 3-3373.";
  }

  rpc reboot {
    description "Операция перезагрузки.";
    input {
      leaf delay {
        type uint32;
        units "seconds";
        default 0;
        description
          "Число секунд ожидания перед операцией перезагрузки.";
      }
      leaf message {
        type string;
        description
          "Сообщение для вывода в начале перезагрузки.";
      }
      leaf language {
        type string;
        description "Строка идентификатора языка.";
        reference "RFC 5646.";
      }
    }
  }

  rpc get-reboot-info {
    description
      "Получение параметров, использованных при последней перезагрузке.";
    output {
      leaf reboot-time {
        type uint32;
        description
          "Параметр delay при последней перезагрузке.";
      }
      leaf message {
        type string;
        description
          "Параметр message при последней перезагрузке.";
      }
      leaf language {
        type string;
        description
          "Параметр language при последней перезагрузке.";
      }
    }
  }
}
```

Приведённый ниже модуль YANG служит примером действия (action) YANG.

```
module example-actions {
  yang-version 1.1;
  namespace "https://example.com/ns/example-actions";
  prefix "act";
  import ietf-yang-types { prefix yang; }

  organization "Example, Inc.";
  contact "support at example.com";
  description "Example Actions Data Model Module.";
  revision "2016-07-07" {
    description "Первый выпуск.";
    reference "example.com document 2-9973.";
  }

  container interfaces {
    description "Интерфейсы системы.";
    list interface {
      key name;
      description "Один интерфейс.";
      leaf name {
        type string;
```

```
          description "Имя интерфейса.";
        }

        action reset {
          description "Сброс интерфейса.";
          input {
            leaf delay {
              type uint32;
              units "seconds";
              default 0;
              description
                "Число секунд ожидания перед сбросом интерфейса.";
            }
          }
        }

        action get-last-reset-time {
          description
            "Получение времени последнего сброса интерфейса.";
          output {
            leaf last-reset {
              type yang:date-and-time;
              mandatory true;
              description
                "Дата и время последнего сброса интерфейса или
                 время последней перезагрузки устройства.";
            }
          }
        }
      }
    }
  }
```

Клиент может передать сообщение с запросом POST для вызова операции RPC reboot.

```
POST /restconf/operations/example-ops:reboot HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<input xmlns="https://example.com/ns/example-ops">
 <delay>600</delay>
 <message>Отключение системы для обслуживания</message>
 <language>ru-RU</language>
</input>
```

Сервер может возвратить отклик

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
```

Ниже приведён тот же пример с использованием кодирования JSON

```
POST /restconf/operations/example-ops:reboot HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "example-ops:input" : {
    "delay" : 600,
    "message" : "Отключение системы для обслуживания",
    "language" : "ru-RU"
  }
}
```

Клиент может передать сообщение с запросом POST для вызова действия reset.

```
POST /restconf/data/example-actions:interfaces/\
   interface=eth0/reset HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<input xmlns="https://example.com/ns/example-actions">
  <delay>600</delay>
</input>
```

Сервер может возвратить отклик

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
```

Тот же пример запросного сообщения с кодированием JSON представлен ниже.

```
POST /restconf/data/example-actions:interfaces/\
  interface=eth0/reset HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{ "example-actions:input" : {
    "delay" : 600
  }
}
```

3.6.2. Кодирование выходных параметров ресурса операции

Если оператор rpc или action имеет раздел output, экземпляры этих выходных параметров кодируются с использованием пространства имён модуля, где определён оператор rpc или action, в элемент XML или объект JSON с именем input с использованием пространства имён модуля, где определён оператор rpc или action.

Если вызванная операция RPC завершилась без ошибок и оператор rpc или action имеет раздел output, а дерево объекта output содержит какие-либо дочерние узлы данных, которые считаются обязательными, сервер должен передать message в своём отклике.

Если вызванная операция RPC завершилась без ошибок и оператор rpc или action имеет раздел output, а дерево объекта output содержит каких-либо дочерних узлов данных, которые считаются обязательными, сервер **может** включить message-body в свой отклик.

Идентификатор URI для запроса не возвращается в отклике. Знание этого идентификатора может потребоваться для связывания вывода с конкретным оператором rpc или action, использованным в запросе.

В примере с использованием модуля YANG example-ops, определённого в параграфе 3.6.1, клиент может передать сообщение с запросом POST для вызова операции get-reboot-info.

```
POST /restconf/operations/example-ops:get-reboot-info HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "example-ops:output" : {
    "reboot-time" : 30,
    "message" : "Отключение системы для обслуживания",
    "language" : "ru-RU"
  }
}
```

Тот же отклик с использованием кодирования XML будет иметь вид

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+xml

<output xmlns="https://example.com/ns/example-ops">
  <reboot-time>30</reboot-time>
  <message>Отключение системы для обслуживания</message>
  <language>ru-RU</language>
</output>
```

В примере с использованием модуля YANG example-actions, определённого в параграфе 3.6.1, клиент может передать сообщение с запросом POST для вызова операции get-last-reset-time.

```
POST /restconf/data/example-actions:interfaces/\
   interface=eth0/get-last-reset-time HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "example-actions:output" : {
    "last-reset" : "2015-10-10T02:14:11Z"
  }
}
```

3.6.3. Кодирование ошибок ресурса операции

Если при попытке вызова операции или действия возникает какая-либо ошибка, возвращается тип носителя errors с соответствующим статусом ошибки.

Если (1) операция RPC недействительна или (2) операция RPC вызвана, но завершилась ошибкой, сервер **должен** передать сообщение с message-body с ресурсом errors, как определено в параграфе 3.9.

Для использования операции RPC reboot из примера в параграфе 3.6.1 клиент может передать запрос POST.

```
POST /restconf/operations/example-ops:reboot HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<input xmlns="https://example.com/ns/example-ops">
  <delay>-33</delay>
  <message>Отключение системы для обслуживания</message>
  <language>ru-RU</language>
</input>
```

Сервер может вернуть сообщение с ошибкой invalid-value

```
HTTP/1.1 400 Bad Request
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+xml

<errors xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <error>
    <error-type>protocol</error-type>
    <error-tag>invalid-value</error-tag>
    <error-path xmlns:ops="https://example.com/ns/example-ops">
      /ops:input/ops:delay
    </error-path>
    <error-message>Недействительный входной параметр</error-message>
  </error>
</errors>
```

Тот же отклик при использовании кодирования JSON будет иметь вид

```
HTTP/1.1 400 Bad Request
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{ "ietf-restconf:errors" : {
    "error" : [
      {
        "error-type" : "protocol",
        "error-tag" : "invalid-value",
        "error-path" : "/example-ops:input/delay",
        "error-message" : "Недействительный входной параметр"
      }
    ]
  }
}
```

3.7. Ресурс схемы

Сервер может поддерживать извлечение применяемых им модулей YANG. Если такое извлечение поддерживается, **должен** присутствовать лист schema в связанной записи списка module, определённой в [RFC7895].

Для извлечения модуля YANG клиенту нужно сначала получить URL для схемы, хранящийся в листе schema. Отметим, что требований к структуре этого URL не задано. Представленное ниже значение URL служит лишь примером.

Клиент может передать сообщение с запросом GET

```
GET /restconf/data/ietf-yang-library:modules-state/\
    module=example-jukebox,2016-08-15/schema HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "ietf-yang-library:schema" :
   "https://example.com/mymodules/example-jukebox/2016-08-15"
}
```

Затем клиенту нужно извлечь актуальную схему YANG, для чего клиент может передать сообщение с запросом GET

```
GET https://example.com/mymodules/example-jukebox/\
   2016-08-15 HTTP/1.1
Host: example.com
Accept: application/yang
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang
   // Содержимое модуля YANG в этом примере не показано.
```

3.8. Ресурс потока событий

Этот ресурс представляет собой источник генерируемых системой уведомлений о событиях. Каждый поток создаётся и изменяется только сервером. Клиент может найти ресурс потока или инициировать продолжительный опрос потока SSE [W3C.REC-eventsource-20150203] с использованием процедуры, описанной в параграфе 6.3.

Поток событий функционирует в соответствии со спецификацией уведомлений о событиях NETCONF [RFC5277]. Доступные потоки можно узнать из списка stream, который указывает синтаксис и семантику потоковых ресурсов.

3.9. Шаблон данных YANG errors

Шаблон данных YANG errors моделирует набор информации об ошибках, которые передаются в message-body откликов сервера при обработке запроса. Это не считается типом ресурсов, поскольку результаты не возвращаются с помощью запросов GET.

Модуль YANG ietf-restconf содержит шаблон данных yang-errors, который задаёт синтаксис и семантику контейнера errors в откликах RESTCONF. Поведение RESTCONF в части обработки ошибок описано в разделе 7.

4. Методы RESTCONF

RESTCONF использует методы HTTP для указания операций CRUD, запрашиваемых для конкретного ресурса.

В таблице показана связь между операциями протоколов RESTCONF и NETCONF.

Методы CRUD в RESTCONF.

RESTCONF	***NETCONF***
OPTIONS	нет
HEAD	<get-config>, <get>
GET	<get-config>, <get>
POST	<edit-config> (nc:operation="create")
POST	Вызов операции RPC
PUT	<copy-config> (PUT в хранилище данных)
PUT	<edit-config> (nc:operation="create/replace")
PATCH	<edit-config> (nc:operation зависит от содержимого PATCH)
DELETE	<edit-config> (nc:operation="delete")

Атрибут remove операции редактирования NETCONF <edit-config> не поддерживается методом HTTP DELETE. Ресурс должен существовать, иначе метод DELETE будет приводить к отказу. Метод PATCH эквивалентен операции редактирования merge при использовании протокола исправления (параграф 4.6.1), другие типы носителей могут обеспечивать более тонкое управление.

Для ограничения операций CRUD могут применяться механизмы контроля доступа. В частности, протокол RESTCONF совместим с моделью управления доступом NACM [RFC6536], поскольку имеются конкретные сопоставления между операциями RESTCONF и NETCONF. Пути к ресурсам сервер должен преобразовывать внутренними средствами в соответствующие идентификаторы экземпляров YANG. Используя эту информацию, сервер может применять правила контроля доступа NACM к сообщениям RESTCONF.

Серверу **недопустимо** разрешать какие-либо операции RESTCONF по отношению к любым ресурсам, для которых у клиента нет полномочий доступа.

Реализация всех методов (кроме PATCH [RFC5789]) определена в [RFC7231]. В этом разделе определено использование протокола RESTCONF с каждым из методов HTTP.

4.1. Метод OPTIONS

Метод OPTIONS передаётся клиентом для определения поддерживаемых сервером методов для конкретного ресурса (например, GET, POST, DELETE). Сервер **должен** реализовать этот метод.

Поле заголовка Accept-Patch **должно** поддерживаться и возвращаться в отклике на запрос OPTIONS, как указано в [RFC5789].

4.2. Метод HEAD

Сервер RESTCONF **должен** поддерживать метод HEAD, который применяется клиентами для извлечения лишь полей заголовков (которые содержат метаданные для ресурса), которые вернул бы и сравнимый метод GET, но без message-body в отклике. Это поддерживается для всех ресурсов, разрешающих метод GET.

Запрос **должен** содержать идентификатор URI, который включает по меньшей мере корневой ресурс. Все параметры, поддерживаемые для запросов GET, допустимы и с методом HEAD.

Контроль доступа выполняется так же, как происходило бы для метода GET. Сервер **должен** давать такой же отклик, как при запросе GET (взамен HEAD), но без включения в отклик message-body.

4.3. Метод GET

Сервер RESTCONF **должен** поддерживать метод GET, который применяется клиентами для извлечения данных и метаданных ресурса. Запрос **должен** содержать идентификатор URI, который включает по меньшей мере корневой ресурс.

Серверу **недопустимо** возвращать какие-либо ресурсы данных, для которых у пользователя нет прав на чтение. Если пользователь не уполномочен читать целевой ресурс, **следует** возвращать отклик об ошибке со статусом «401 Unauthorized». В этом случае возвращается тег ошибки access-denied. Сервер **может** возвращать статус «404 Not Found», как описано в параграфе 6.5.4 [RFC7231]. В этом случае возвращается тег ошибки invalid-value.

Если пользователь имеет полномочия на чтение лишь части целевого ресурса, не разрешённые для этого пользователя компоненты исключаются из message-body в отклике и возвращается только разрешённая часть.

При возврате клиенту какого-либо содержимого сервер **должен** включить в отклик действительное тело сообщения. **Недопустим** возврат более одного элемента для кодирования XML. Если передаётся множество элементов в JSON message-body, они **должны** отправляться как массив JSON. В этом случае временная метка и тег объекта в отклике **должны** относиться к первому возвращаемому элементу.

Если запрос на извлечение ресурса данных, представляющего лист-список или список YANG даёт более одного экземпляра и отклике применяется кодирование XML, сервер **должен** возвращать отклик об ошибке с кодом «400 Bad Request». В этом случае применяется тег ошибки invalid-value. Отметим, что список неконфигурационных данных не требуется для определения каких-либо ключей. В этом случае получение одного экземпляра списка невозможно.

Если запрос на извлечение ресурса данных представляет несуществующий экземпляр, сервер **должен** возвращать отклик об ошибке с кодом «404 Not Found». В этом случае возвращается тег ошибки invalid-value.

Если целевым ресурсом запроса на извлечение является ресурс операции, сервер должен возвращать статус «405 Method Not Allowed». В этом случае возвращается тег ошибки operation-not-supported.

Отметим, что способ контроля доступа к ресурсам данным не полностью совместим с кэшированием HTTP. Поля заголовка Last-Modified и Etag, поддерживаемые для ресурса данных, не меняются в результате смены правил доступа к этому ресурсу. Возможно, что представление ресурса данных для конкретного клиента будет изменено без отражения этого через поля Last-Modified и ETag.

Например, клиент может запросить поля заголовков отклика для XML-представления определённого ресурса album.

```
GET /restconf/data/example-jukebox:jukebox/\
   library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+xml
Cache-Control: no-cache
ETag: "a74eefc993a2b"
Last-Modified: Thu, 26 Jan 2017 14:02:14 GMT

<album xmlns="http://example.com/ns/example-jukebox"
       xmlns:jbox="http://example.com/ns/example-jukebox">
  <name>Wasting Light</name>
  <genre>jbox:alternative</genre>
  <year>2011</year>
</album>
```

Дополнительные примеры извлечения ресурсов представлены в Приложении B.1.

4.4. Метод POST

Сервер RESTCONF **должен** поддерживать метод POST, используемый клиентом для создания ресурсов данных и вызова ресурсов операций. Сервер определяет способ обработки запроса по типу целевого ресурса.

Типы ресурсов, поддерживающих метод POST.

Тип	***Описание***
Datastore	Создаёт ресурс конфигурационных данных верхнего уровня
Data	Создаёт дочерний ресурс конфигурационных данных
Operation	Вызывает операцию RPC

4.4.1. Режим создания ресурса

Если целевым является ресурс хранилища или данных, метод POST считается запросом на создание ресурса верхнего уровня или дочернего ресурса, соответственно. Предполагается, что message-body содержит дочерний ресурс для создания в родительском (целевом) ресурсе. Поле message-body **должно** содержать в точности один экземпляр ожидаемого типа ресурса. Моделью данных для дочернего дерева является субдерево, определённое в YANG для дочернего ресурса.

Параметры запроса insert (параграф 4.8.5) и point (параграф 4.8.6) **должны** поддерживаться методом POST для ресурсов хранилищ и данных. Эти параметры разрешены только для списков и листьев-списков, упорядочиваемых пользователем (ordered-by user).

При успешном выполнении метода POST возвращается отклик «201 Created» и сообщение не содержит message-body. Поле заголовка Location, указывающее созданный дочерний ресурс **должно** в этом случае присутствовать в отклике.

Если ресурс данных уже имеется, метод POST **должен** вызывать ошибку и возврат отклика с кодом «409 Conflict». В этом случае применяется тег ошибки resource-denied.

Если пользователь не имеет прав доступа к целевому ресурсу, **следует** возвращать статус ошибки «403 Forbidden». В этом случае применяется тег ошибки access-denied. Сервер **может** возвращать статус «404 Not Found», как описано в параграфе 6.5.4 [RFC7231]. В этом случае применяется тег ошибки invalid-value. Все прочие отклики об ошибках обрабатываются в соответствии с процедурами раздела 7.

Например, для создания нового ресурса jukebox клиент может передать запрос вида

```
POST /restconf/data HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{ "example-jukebox:jukebox" : {} }
```

Если ресурс создан, сервер может возвратить отклик

```
HTTP/1.1 201 Created
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Location: https://example.com/restconf/data/\
    example-jukebox:jukebox
Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
ETag: "b3a3e673be2"
```

Дополнительные примеры создания ресурсов представлены в Приложении B.2.1.

4.4.2. Вызов режима операции

Если целевым является ресурс операции, метод POST считается запросом на вызов этой операции. Тело сообщения (если оно имеется) с запросом обрабатывается как входные параметры операции. Дополнительная информация о ресурсах операций представлена в параграфе 3.6.

Если запрос POST выполнен успешно, возвращается отклик «200 OK» при наличии message-body и «204 No Content» при отсутствии тела сообщения в отклике.

Если пользователь не имеет прав доступа к целевой операции, **следует** возвращать статус ошибки «403 Forbidden». В этом случае применяется тег ошибки access-denied. Сервер **может** возвращать статус «404 Not Found», как описано в параграфе 6.5.4 [RFC7231]. В этом случае применяется тег ошибки invalid-value. Все прочие отклики об ошибках обрабатываются в соответствии с процедурами раздела 7.

Например, клиент может вызвать операцию play, определённую в модуле YANG example-jukebox.

```
POST /restconf/operations/example-jukebox:play HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "example-jukebox:input" : {
    "playlist" : "Foo-One",
    "song-number" : 2
  }
}
```

Сервер может возвратить отклик

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
```

4.5. Метод PUT

Сервер RESTCONF **должен** поддерживать метод PUT, с помощью которого клиент создаёт или заменяет ресурс данных. В запросе **должно** присутствовать поле message-body, представляющее новый ресурс данных, иначе сервер **должен** возвращать код состояния «400 Bad Request». В этом случае применяется тег ошибки invalid-value.

Для создания ресурса могут применяться методы POST и PUT. Различие состоит в том, что в методе POST клиент не указывает идентификатор для создаваемого ресурса, поскольку целевой ресурс метода POST является родителем для создаваемого ресурса. Для метода PUT целевым является создаваемый ресурс.

Метод PUT **должен** поддерживаться для ресурсов данных и хранилищ. PUT для хранилища данных служит для полной замены содержимого хранилища, а применительно к ресурсу данных PUT заменяет в хранилище лишь этот ресурс.

Параметры запроса insert (параграф 4.8.5) и point (параграф 4.8.6) **должны** поддерживаться методом PUT для ресурсов данных. Эти параметры разрешены только для списков и листьев-списков, упорядочиваемых пользователем (ordered-by user).

В соответствии с [RFC7231] при создании методом PUT нового ресурса возвращается статус «201 Created». Если же изменяется имеющийся ресурс, кодом возврата будет «204 No Content».

Если пользователь не имеет полномочий на создание или замену целевого ресурса, **следует** возвращать статус ошибки «403 Forbidden». В этом случае применяется тег ошибки access-denied.

Сервер **может** возвращать статус «404 Not Found», как описано в параграфе 6.5.4 [RFC7231]. В этом случае применяется тег ошибки invalid-value. Прочие отклики об ошибках обрабатываются в соответствии с процедурами раздела 7.

Если целевой ресурс представляет лист-список YANG, методу PUT **недопустимо** менять значение экземпляра leaf-list.

Если целевой ресурс представляет список YANG, значения ключевых листьев в представлении message-body **должны** совпадать со значениями ключевых листьев в URI запроса. Метод PUT **недопустимо** применять для смены значений ключевых листьев экземпляра ресурса данных.

Например, клиент может изменить дочерний ресурс album в модуле YANG example-jukebox или создать новый, если альбома ещё нет. Для замены содержимого ресурса album клиент может передать запрос вида

```
PUT /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "example-jukebox:album" : [
    {
      "name" : "Wasting Light",
      "genre" : "example-jukebox:alternative",
      "year" : 2011
    }
  ]
}
```

Если ресурс обновлён, сервер может возвратить отклик

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
```

```
ETag: "b27480aeda4c"
```

Ниже приведён тот же запрос с использованием кодирования XML.

```
PUT /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<album xmlns="http://example.com/ns/example-jukebox"
       xmlns:jbox="http://example.com/ns/example-jukebox">
  <name>Wasting Light</name>
  <genre>jbox:alternative</genre>
  <year>2011</year>
</album>
```

Пример использования метода PUT для замены содержимого ресурса хранилища приведён в Приложении B.2.4.

4.6. Метод PATCH

Сервер RESTCONF **должен** поддерживать метод PATCH для простого исправления (plain patch) и **может** поддерживать другие типы носителей. Типы носителей для метода PATCH, поддерживаемые сервером, клиент может узнать, передав запрос OPTIONS и проверив в отклике поле заголовка Accept-Patch (параграф 4.1).

RESTCONF использует метод HTTP PATCH, определённый в [RFC5789], для реализации расширяемой схемы механизмов «наложения заплаток» (patching) на ресурсы. Каждый механизм должен иметь уникальный тип носителя.

В этом документе определён один механизм (параграф 4.6.1). Другой механизм - YANG Patch определён в [YANG-Patch]. Будущие спецификации могут определять дополнительные механизмы.

Если целевого экземпляра ресурса не существует, серверу **недопустимо** создавать его.

При успешном выполнении запроса PATCH возвращается отклик «200 OK» при наличии message-body и «204 No Content», если тела сообщения не передаётся.

Если пользователь не уполномочен изменять целевой ресурс, **следует** возвращать отклик об ошибке со статусом «403 Forbidden». Сервер **может** возвращать статус «404 Not Found», как описано в параграфе 6.5.4 [RFC7231]. В этом случае применяется тег ошибки invalid-value. Все прочие отклики об ошибках обрабатываются в соответствии с процедурами, определёнными в разделе 7.

4.6.1. Механизм простого исправления

Механизм простого исправления (plain patch) объединяет тело сообщения с целевым ресурсом. Поле message-body **должно** присутствовать для простого исправления и **должно** иметь тип носителя application/yang-data+xml или application/yang-data+json.

Простое исправление можно применять для создания или обновления, но не для удаления дочернего ресурса внутри целевого. Поддержка другого типа носителя для возможности удаления дочерних ресурсов описана в [YANG-Patch]. Тип носителя YANG Patch позволяет выполнить множество субопераций (например, merge, delete) в одном запросе PATCH.

Если целевой ресурс представлен узлом YANG, методу PATCH **недопустимо** менять значение экземпляра leaf-list.

Если целевой ресурс представлен экземпляром YANG list, значения ключевых листьев в представлении message-body **должны** совпадать со значениями ключевых листьев в URI запроса. Метод PATCH **недопустимо** применять для изменения значений ключевых листьев для экземпляра ресурса данных.

После обработки простого исправления сервером клиенту будет возвращаться отклик, как описано в параграфе 4.6.

Например, для замены поля year в ресурсе album (вместо замены всего ресурса методом PUT) клиент может передать простое исправление

```
PATCH /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1
Host: example.com
If-Match: "b8389233a4c"
Content-Type: application/yang-data+xml

<album xmlns="http://example.com/ns/example-jukebox">
 <year>2011</year>
</album>
```

Если поле будет обновлено, сервер может вернуть отклик вида

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
ETag: "b2788923da4c"
```

4.7. Метод DELETE

Сервер RESTCONF **должен** поддерживать метод DELETE, используемый клиентом для удаления целевого ресурса. При успешном удалении возвращается статус «204 No Content».

Если пользователь не уполномочен удалять целевой ресурс, **следует** возвращать отклик об ошибке со статусом «403 Forbidden». Сервер **может** возвращать статус «404 Not Found», как описано в параграфе 6.5.4 [RFC7231]. В этом случае применяется тег ошибки invalid-value. Все прочие отклики об ошибках обрабатываются в соответствии с процедурами, определёнными в разделе 7.

Если целевой узел представляет список или лист-список конфигурационных данных, он **должен** представлять один экземпляр листа списка или списка YANG. Серверу **недопустимо** использовать метод DELETE для удаления более одного такого экземпляра.

Например, для удаления ресурса album с ключом Wasting Light клиент может передать запрос вида

```
DELETE /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1
Host: example.com
```

Если ресурс удалён, сервер может возвратить отклик

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
```

4.8. Параметры запроса

Каждая операция RESTCONF позволяет включать (необязательно) параметры в URI запроса. Возможность включения конкретного параметра зависит от типа ресурса, а иногда и от конкретного целевого ресурса в запросе.

- Параметры запроса могут размещаться в любом порядке.
- Каждый параметр может включаться в URI запроса не более одного раза.
- при наличии в запросе более одного экземпляра параметра сервер **должен** возвращать статус «400 Bad Request». В этом случае применяется тег ошибки invalid-value.
- При отсутствии параметра используется заданное по умолчанию значение.
- Регистры символов в именах и значениях параметров принимаются во внимание.
- Сервер **должен** возвращать статус «400 Bad Request» при получении неожиданного параметра в запросе. В этом случае применяется тег ошибки invalid-value.

Параметры запроса RESTCONF.

Имя	*Методы*	*Описание*
content	GET, HEAD	Выбирает ресурсы конфигурационных и/или неконфигурационных данных
depth	GET, HEAD	Запрашивает ограниченную глубину субдерева в содержимом отклика
fields	GET, HEAD	Запрашивает подмножество содержимого целевого ресурса
filter	GET, HEAD	Логический фильтр уведомлений для ресурса потока событий
insert	POST, PUT	Режим вставки для упорядочиваемых пользователем ресурсов данных
point	POST, PUT	Место вставки для упорядочиваемых пользователем ресурсов данных
start-time	GET, HEAD	Время начала в буфере воспроизведения для ресурса потока событий
stop-time	GET, HEAD	Время остановки в буфере воспроизведения для ресурса потока событий
with-defaults	GET, HEAD	Управляет извлечением принятых по умолчанию значений

Примеры использования параметров запросов приведены в Приложении В.3.

Если производители определяют дополнительные параметры запросов, им **следует** использовать префикс (такой, как название предприятия) для имён параметров запроса с целью предотвращения конфликтов с другими параметрами.

4.8.1. Параметр content

Параметр content управляет обработкой в отклике узлов-потомков запрашиваемого узла. Разрешённые значения приведены в таблице.

Значение	*Описание*
config	Возвращаются только конфигурационные узлы-потомки.
nonconfig	Возвращаются только узлы-потомки, не являющиеся конфигурационными.
all	Возвращаются все узлы-потомки.

Этот параметр разрешён только для метода GET применительно к ресурсам данных и хранилищ. При использовании другого метода или типа ресурса возвращается отклик «400 Bad Request».

Если этот параметр запроса не указан, применяется значение all. Параметр **должен** поддерживаться сервером.

4.8.2. Параметр depth

Параметр запроса depth указывает глубину возвращаемого сервером субдерева. Узлы данных со значением глубины, превышающим параметр depth, не возвращаются в отклике на запрос с методом GET.

Запрошенный узел данных имеет глубину 1. Если применяется параметр fields (параграф 4.8.3) для выбора узлов-потомков, эти узлы и все их предки будут иметь «глубину» 1 (это эффект включения узлов, заданных полем fields, даже если значение depth меньше реальной глубины заданных полей). Все прочие дочерние узлы имеют глубину на 1 больше глубины для их родителя.

Параметр depth может принимать целочисленные значения от 1 до 65535 или строку «unbounded» (используется по умолчанию).

Этот параметр разрешён только для запросов GET к ресурсам API, хранилищ и данных. В остальных случаях будет возвращаться статус «400 Bad Request».

По умолчанию сервер будет включать все субресурсы найденного ресурса, которые имеют тот же тип, что и запрошенный ресурс. Исключением является ресурс хранилища данных, для которого по умолчанию возвращается хранилище данных и все его дочерние ресурсы данных.

Если URI параметра запроса depth указан в листе списке capability, определённом в параграфе 9.3, это говорит о поддержке сервером параметра запроса depth.

4.8.3. Параметр fields

Параметр запроса fields опционально применяется для указания узлов данных в целевом ресурсе для извлечения методом GET. Клиент может использовать этот параметр для задания подмножества извлекаемых узлов ресурса.

Сервер будет возвращать тело сообщения, представляющее целевой ресурс, с узлами-потомками, «обрезанными» в соответствии со значением fields-expr. Сервер не возвращает отдельных субресурсов.

Значение параметра fields следует приведённому ниже правилу

```
fields-expr = path "(" fields-expr ")" / path ";" fields-expr / path
path = api-identifier [ "/" path ]
```

Идентификатор api-identifier определён в параграфе 3.5.3.1, символ «;» служит для выбора множества узлов. Например, для извлечение лишь жанра (genre) и года выпуска (year) альбома можно указать fields=genre;year.

Скобки служат для задания субселекторов узла. Отметим отсутствие символа-разделителя «/» между полем path и левой скобкой (.

Предположим, например, что целевым ресурсом является список album. Для извлечения только label и catalogue-number из контейнера admin в альбоме, можно использовать fields=admin(label;catalogue-number).

Символ / используется в пути для извлечения дочернего узла. Например, для извлечения лишь метки (label) альбома можно указать fields=admin/label.

Этот параметр разрешён только для запросов GET к ресурсам API, хранилищ и данных. В остальных случаях будет возвращаться статус «400 Bad Request».

Если URI параметра запроса fields указан в листе списке capability, определённом в параграфе 9.3, это говорит о поддержке сервером параметра запроса fields.

4.8.4. Параметр filter

Параметр запроса filter служит для указания интересующего подмножества всех возможных событий. При отсутствии этого параметра будут передаваться все события, не исключённые другими параметрами.

Этот параметр разрешён только для запросов GET к ресурсам потока событий. В остальных случаях будет возвращаться статус «400 Bad Request».

Для этого параметра применяется формат выражений XPath 1.0 [XPath], которые вычисляются в описанном ниже контексте.

- Набором деклараций пространств имён является набор пар префиксов и пространств имён для всех поддерживаемых модулей YANG, где префиксом является имя модуля YANG, а пространство имён определяется оператором namespace в модуле YANG.
- Библиотекой функций служит библиотека функций ядра, определённая в XPath 1.0, с добавлением любых функций, определённых моделью данных.
- Набор привязок переменных пуст.
- Узлом контекста является корневой узел.

Параметр запроса filter применяется в соответствии с определением параграфа 3.6 в [RFC5277]. Если логическим результатом выражения для концептуального корня документа notification является true, уведомление о событии доставляется клиенту.

Если URI параметра запроса filter указан в листе списке capability, определённом в параграфе 9.3, это говорит о поддержке сервером параметра запроса filter.

4.8.5. Параметр insert

Параметр запроса insert служит для указания способа размещения ресурса в упорядочиваемом пользователем списке. Возможные значения приведены в таблице.

Значение	*Описание*
first	Вставка новых данных как первой записи.
last	Вставка новых данных как последней записи.
before	Вставка новых данных перед позицией, указанной параметром point.
after	Вставка новых данных после позиции, указанной параметром point.

По умолчанию применяется значение last.

Этот параметр поддерживается только для методов POST и PUT и лишь для целевых ресурсов, которые являются данными, представляющими список или лист-список YANG, упорядочиваемый пользователем (ordered-by user).

При использовании значения before или after **должен** указываться также параметр point. В противном случае будет возвращаться статус ошибки «400 Bad Request».

Сервер **должен** поддерживать параметр insert.

4.8.6. Параметр point

Параметр point служит для указания позиции вставки создаваемого или перемещаемого ресурса данных в список или лист-список, упорядочиваемый пользователем.

Значением параметра point является строка, указывающая путь для вставки объекта. Формат строки совпадает с форматом строки URI для целевого ресурса.

Этот параметр поддерживается только для методов POST и PUT и лишь для целевых ресурсов, которые являются данными, представляющими список или лист-список YANG, упорядочиваемый пользователем (ordered-by user).

Если в запросе не указан также параметр insert или он имеет значение, отличное от before и after, будет возвращаться статус ошибки «400 Bad Request».

Этот параметр содержит идентификатор экземпляра ресурса, служащего точкой вставки для метода POST или PUT.

Сервер **должен** поддерживать параметр point.

4.8.7. Параметр start-time

Параметр запроса start-time служит для инициирования функции воспроизведения уведомлений, определённой в [RFC5277], и указывает начальный момент для воспроизведения записанных уведомлений. Если сервер не поддерживает воспроизведения уведомлений согласно атрибуту replay-support, возвращённому записью списка stream для потокового ресурса, сервер **должен** возвращать статус «400 Bad Request».

Значение параметра start-time имеет тип date-and-time, определённый в модуле YANG ietf-yang-types [RFC6991].

Этот параметр разрешён только для запросов GET к ресурсам данных text/event-stream. В остальных случаях будет возвращаться статус «400 Bad Request».

Если этот параметр отсутствует, подписка на воспроизведение не запрашивается. Время начала воспроизведения, превышающее текущее время, будет недействительным. Если указано время начала раньше, чем имеется в записи, воспроизведение начнётся с самого раннего из доступных уведомлений. Клиент может узнать текущее время сервера из поля заголовка Date, возвращаемого сервером в откликах в соответствии с [RFC7231].

Если этот параметр поддерживается сервером, идентификатор URI параметра запроса replay **должен** быть указан в листе-списке capability, определённом в параграфе 9.3, и сервер **должен** поддерживать также параметр stop-time.

Если лист replay-support в записи stream (параграф 9.3) имеет значение true, сервер **должен** поддерживать для этого потока параметры start-time и stop-time.

4.8.8. Параметр stop-time

Параметр запроса stop-time используется с функцией воспроизведения для указания последнего из интересующих уведомлений. Этот параметр **должен** использоваться вместе с параметром start-time и иметь большее значение.

Значение параметра stop-time имеет тип date-and-time, определённый в модуле YANG ietf-yang-types [RFC6991].

Этот параметр разрешён только для запросов GET к ресурсам данных text/event-stream. В остальных случаях будет возвращаться статус «400 Bad Request».

Если этот параметр отсутствует, уведомления будут воспроизводиться до завершения подписки. Допускается указание будущего времени.

Если этот параметр поддерживается сервером, идентификатор URI параметра запроса replay **должен** быть указан в листе-списке capability, определённом в параграфе 9.3, и сервер **должен** поддерживать также параметр start-time.

Если лист replay-support в записи stream (параграф 9.3) имеет значение true, сервер **должен** поддерживать для этого потока параметры start-time и stop-time.

4.8.9. Параметр with-defaults

Параметр запроса with-defaults служит для управления возвратом информации из узлов с принятыми по умолчанию значениями при запросах GET к ресурсам данных.

Если сервер поддерживает такую возможность, он **должен** реализовать поведение, описанное в параграфе 4.5.1 [RFC6243], применительно к операциям RESTCONF GET взамен операций NETCONF.

Значение	*Описание*
report-all	Возвращаются все узлы данных
trim	Узлы данных с принятыми по умолчанию значениями YANG не возвращаются
explicit	Узлы данных с принятыми по умолчанию значениями YANG возвращаются
report-all-tagged	Возвращаются все узлы данных, а узлы с принятыми по умолчанию значениями помечаются

Если для параметра with-defaults задано значение report-all, сервер **должен** придерживаться режима возврата принятых по умолчанию значений, определённого в параграфе 3.1 [RFC6243].

При with-defaults=trim сервер **должен** сервер **должен** придерживаться режима возврата принятых по умолчанию значений, определённого в параграфе 3.2 [RFC6243].

Для with-defaults=explicit сервер **должен** сервер **должен** придерживаться режима возврата принятых по умолчанию значений, определённого в параграфе 3.3 [RFC6243].

Если установлено with-defaults=report-all-tagged, сервер **должен** придерживаться режима возврата принятых по умолчанию значений, определённого в параграфе 3.4 [RFC6243]. Метаданные возвращаются сервером в соответствии с параграфом 5.3. XML-кодирование для атрибута default, передаваемого сервером для узлов с принятыми по умолчанию значениями, определено в разделе 6 [RFC6243]. Кодирование JSON для атрибута default **должно** использовать значения, определённые в [RFC6243], но с правилами кодирования [RFC7952]. Для атрибута default **должно** применяться имя модуля ietf-netconf-with-defaults.

Если параметр with-defaults не задан, сервер **должен** следовать при передаче принятых по умолчанию значений поведению, заданному параметром basic-mode для URI возможности defaults и описанному в параграфе 9.1.2.

Если сервер включает URI параметра запроса with-defaults в свой лист-список capability, определённый в параграфе 9.3, он **должен** поддерживать параметр запроса with-defaults.

Поскольку сервер не сообщает параметр also-supported, как описано в параграфе 4.3 [RFC6243], возможно, что некоторые значения параметра with-defaults не будут поддерживаться. Если сервер не поддерживает запрошенное значение with-defaults, он **должен** возвращать статус «400 Bad Request». В таких случаях применяется тег ошибки invalid-value.

5. Сообщения

Протокол RESTCONF использует сообщения HTTP. Одно сообщение HTTP соответствует одному методу протокола. Большинство сообщений может выполнять одну задачу с одним ресурсом, например, извлечение или редактирование ресурса. Исключением является метод PATCH, который позволяет одним сообщением выполнить множество операций редактирования хранилища данных.

5.1. Структура URI запроса

Ресурсы указываются идентификаторами URI, имеющими структуру базовых URI из [RFC3986].

Операции RESTCONF выводятся из метода HTTP и URI запроса с использованием приведённых ниже полей.

```
<OP> /<restconf>/<path>?<query>

  ^       ^        ^       ^
  |       |        |       |
method  entry  resource  query

  M       M        O       O

M=обязательно, O=не обязательно
```

<OP> - метод HTTP;

<restconf> - корневой ресурс RESTCONF;

<path> - URI целевого ресурса;

<query> - список параметров запроса.

- method - метод HTTP, указывающий операцию RESTCONF, запрошенную клиентом, для выполнения применительно к целевому ресурсу, заданному в URI запроса. Подробное описание операций RESTCONF дано в разделе 4.
- entry - корень RESTCONF API, заданного на этом сервере HTTP, определяемый из ресурса /.well-known/host-meta, как описано в параграфе 3.1.
- resource - выражение пути, указывающее ресурс для доступа с помощью операции RESTCONF. Если это поле не указано, целевым ресурсом является интерфейс API, представленный шаблоном данных YANG с именем yang-api (раздел 8).
- query - набор параметров, связанных с сообщением RESTCONF (см. параграф 3.4 в [RFC3986]). Параметры RESTCONF имеют форму пар name=value. Большинство параметров являются необязательными для реализации сервером и применения клиентами. Каждый необязательный параметр запроса указывается идентификатором URI. Сервер **должен** перечислять URI поддерживаемых параметров запроса в листе-списке capability, определённом в параграфе 9.3.

Определён конкретный набор параметров, хотя сервер **может** поддерживать параметры, не определённые в этом документе. Содержимое любого параметра запроса **должно** кодироваться в соответствии с параграфом 3.4 [RFC3986]. Для всех зарезервированных символов **должно** применяться %-кодирование в соответствии с параграфами 2.1 и 2.5 [RFC3986].

Отметим, что компонент фрагментов не используется протоколом RESTCONF и фрагмент исключается сервером из целевого URI, как описано в параграфе 5.1 [RFC7230].

При создании клиентом нового ресурса возвращается поле заголовка Location, который указывает путь для созданного ресурса. Клиент применяет именно этот идентификатор пути для доступа к ресурсу после его создания.

Целью операций RESTCONF являются ресурсы. Поле path в URI запроса представляет целевой ресурс для операции RESTCONF.

Примеры URI запросов RESTCONF представлены в Приложении B.

5.2. Кодирование сообщений

Сообщения RESTCONF кодируются в HTTP согласно [RFC7230] с использованием кодировки utf-8 для всех сообщений. Содержимое сообщения RESTCONF передаётся в теле сообщения HTTP.

Содержимое кодируется в формате JSON или XML. Сервер **должен** поддерживать один из этих форматов и **может** поддерживать оба. Клиенту нужно поддерживать XML и JSON для взаимодействия со всеми серверами RESTCONF.

Правила кодирования XML для узлов данных определены в [RFC7950] и применяются для всего содержимого XML. Правила кодирования JSON определены в [RFC7951], а дополнительные правила кодирования метаданных - в [RFC7952]. Это кодирование является корректным JSON, но применяет специальные правила для указания пространства имён модуля и обеспечения согласованного типа обработки данных YANG.

Формат кодирования входных данных запроса указывается полем заголовка Content-Type, которое **должно** присутствовать, если клиент передаёт message-body.

Сервер **должен** поддерживать поле заголовка Accept и код статуса «406 Not Acceptable», как указано в [RFC7231]. Формат кодирования содержимого выхода отклика , который клиент будет воспринимать, указывается полем Accept в

заголовке запроса. Если поле не задано, **следует** использовать формат кодирования входных данных запроса или сервер **может** выбрать любой поддерживаемый формат кодирования содержимого.

Если запрос не имеет входных данных, по умолчанию применяется кодирование XML или JSON, в зависимости от предпочтений сервера. Расширения файлов, закодированные в запросе, не применяются для указания формата кодирования.

Клиент может определить, поддерживает ли сервер RESTCONF формат кодирования, передав запрос с указанием конкретного формата в поле Content-Type и/или Accept. Если сервер не поддерживает запрошенное кодирование ввода из запроса, он **должен** вернуть отклик с кодом «415 Unsupported Media Type». Если сервер не поддерживает ни один из запрошенных методов кодирования вывода, он **должен** вернуть отклик «406 Not Acceptable».

5.3. Метаданные RESTCONF

Протокол RESTCONF должен поддерживать извлечение тех же метаданных, которые используются в протоколе NETCONF. Информация о листьях с принятыми по умолчанию значениями, временных метках последнего изменения и т. п. обычно служит для аннотирования представлений содержимого хранилища данных.

При кодировании XML метаданные представляются как атрибуты XML в соответствии с параграфом 3.3 [W3C.REC-xml-20081126], при кодировании JSON - в соответствии с [RFC7952].

Приведённые ниже примеры основаны на Приложении B.3.9. Режим report-all-tagged для параметра запроса with-defaults требует возврата атрибута default для узлов с принятыми по умолчанию значениями. В примерах этот атрибут используется для листа mtu.

5.3.1. Пример XML-кодирования метаданных

```
GET /restconf/data/interfaces/interface=eth1
    ?with-defaults=report-all-tagged HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+xml

<interface
  xmlns="urn:example.com:params:xml:ns:yang:example-interface">
  <name>eth1</name>
  <mtu xmlns:wd="urn:ietf:params:xml:ns:netconf:default:1.0"
    wd:default="true">1500</mtu>
  <status>up</status>
</interface>
```

5.3.2. Пример JSON-кодирования метаданных

Отметим, что RFC 6243 определяет атрибут default с XSD[1], а не YANG, поэтому имя модуля YANG указано напрямую, а не берётся из модуля YANG. Значение ietf-netconf-with-defaults назначено для JSON-кодирования метаданных.

```
GET /restconf/data/interfaces/interface=eth1\
    ?with-defaults=report-all-tagged HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "example:interface" : [
    {
      "name" : "eth1",
      "mtu" : 1500,
      "@mtu" : {
         "ietf-netconf-with-defaults:default" : true
      },
      "status" : "up"
    }
  ]
}
```

5.4. Статус возврата

Каждое сообщение представляет тот или иной вариант доступа к ресурсу и для каждого запроса возвращается поле заголовка HTTP status-line. Если код статуса относится к группе 4xx, в отклике **следует** возвращать информацию об ошибке в соответствии с форматом, описанным в параграфе 7.1. Для кодов группы 5xx также **может** возвращаться информация об ошибке в описанном в параграфе 7.1 формате. Коды из групп 1xx, 2xx, 3xx не связаны с ошибками, поэтому для таких откликов **недопустимо** включать информацию об ошибке.

[1]XML Schema Definition - определение схемы XML.

5.5. Кэширование сообщений

Поскольку содержимое хранилища данных может меняться в любой момент, отклики от сервера RESTCONF в обычно **не следует** кэшировать. Сервер **должен** включать поле заголовка Cache-Control в каждый отклик для указания режима кэширования.

Вместо того чтобы полагаться на кэширование HTTP, клиенту **следует** контролировать поля заголовка Etag и/или Last-Modified, возвращаемые сервером для ресурса хранилища (или ресурса данных, если сервер поддерживает это). Запрос на извлечение ресурса может включать поля заголовка If-None-Match и/или If-Modified-Since, которые будут требовать от сервера возврата статуса «304 Not Modified», если ресурс не был изменён. Клиент **может** применять метод HEAD для извлечения только полей заголовков, в который **следует** включать поля Etag и Last-Modified, если для целевого ресурса поддерживаются метаданные.

Отметим, что для ресурсов данных может применяться контроль доступа, поэтому значения в полях заголовка Last-Modifiedи Etag, поддерживаемые для ресурса данных, могут оказаться ненадёжными, как описано в параграфе 4.3.

6. Уведомления

Протокол RESTCONF поддерживает определённые в YANG уведомления о событиях. Решение сохраняет возможности уведомления о событиях протокола NETCONF [RFC5277] на основе транспортной стратегии SSE [W3C.REC-eventsource-20150203].

6.1. Поддержка на сервере

Сервер RESTCONF **может** поддерживать уведомления RESTCONF. Клиент может узнать о поддержке уведомлений RESTCONF с помощью метода HTTP OPTIONS, HEAD или GET из списка stream. Если сервер не поддерживает уведомлений RESTCONF, он будет возвращать код ошибки HTTP (например, статус «404 Not Found»).

6.2. Потоки событий

Сервер RESTCONF, поддерживающий уведомления, будет заполнять ресурс потока для каждой точки доступа к службе доставки уведомлений. Клиент RESTCONF может получить список поддерживаемых потоков событий от сервера RESTCONF, использовав метод GET для списка stream.

Определение контейнера restconf-state/streams в модуле ietf-restconf-monitoring (параграф 9.3) используется для задания синтаксиса и структуры концептуальных дочерних ресурсов в ресурсе streams.

Например, клиент может передать запрос вида

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/\
    streams HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Content-Type: application/yang-data+xml

<streams
  xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">
   <stream>
      <name>NETCONF</name>
      <description>Принятый по умолчанию поток событий NETCONF</description>
      <replay-support>true</replay-support>
      <replay-log-creation-time>\
         2007-07-08T00:00:00Z\
      </replay-log-creation-time>
      <access>
         <encoding>xml</encoding>
         <location>https://example.com/streams/NETCONF\
         </location>
      </access>
      <access>
         <encoding>json</encoding>
         <location>https://example.com/streams/NETCONF-JSON\
         </location>
      </access>
   </stream>
   <stream>
      <name>SNMP</name>
      <description>Уведомления SNMP</description>
      <replay-support>false</replay-support>
      <access>
         <encoding>xml</encoding>
         <location>https://example.com/streams/SNMP</location>
      </access>
   </stream>
   <stream>
      <name>syslog-critical</name>
      <description>Критические и важные события</description>
      <replay-support>true</replay-support>
      <replay-log-creation-time>
         2007-07-01T00:00:00Z
      </replay-log-creation-time>
      <access>
         <encoding>xml</encoding>
```

```
            <location>\
              https://example.com/streams/syslog-critical\
            </location>
          </access>
        </stream>
      </streams>
```

6.3. Подписка на уведомления

Клиенты RESTCONF могут определить URL для ресурса подписки (на уведомления) путём отправки запроса GET для листа location с элементом списка stream. Возвращённое значение можно использовать для подписки на уведомления.

Клиент будет передавать запрос HTTP GET для возвращённого сервером URL с Accept типа text/event-stream.

Сервер будет считать соединение потоком событий, используя транспортную стратегию SSE [W3C.REC-eventsource-20150203].

Сервер **может** поддерживать параметры запроса для метода GET применительно к этому ресурсу. Эти параметры зависят от потока событий.

Например, клиент может передать запрос

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/\
    streams/stream=NETCONF/access=xml/location HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может вернуть приведённый ниже отклик.

```
HTTP/1.1 200 OK
Content-Type: application/yang-data+xml

<location
  xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">\
  https://example.com/streams/NETCONF\
</location>
```

Клиент RESTCONF может затем использовать URL для мониторинга потока уведомлений.

```
GET /streams/NETCONF HTTP/1.1
Host: example.com
Accept: text/event-stream
Cache-Control: no-cache
Connection: keep-alive
```

Клиент RESTCONF **может** запросить у сервера сжатие событий с помощью поля в заголовке HTTP Accept-Encoding.

```
GET /streams/NETCONF HTTP/1.1
Host: example.com
Accept: text/event-stream
Cache-Control: no-cache
Connection: keep-alive
Accept-Encoding: gzip, deflate
```

6.3.1. Поток событий NETCONF

Серверам **следует** поддерживать поток событий NETCONF, определённый в параграфе 3.2.3 [RFC5277]. Уведомления для этого потока кодируются в XML.

Сервер **может** поддерживать дополнительные потоки, которые представляют смысловое содержание потока событий NETCONF, но используют представление с другим типом носителя.

Сервер **может** поддерживать параметры запросов start-time, stop-time и filter, определённые в параграфе 4.8. Примеры использования параметров представлены в Приложении B.3.6.

6.4. Приём потока событий

Уведомления RESTCONF кодируются в соответствии с определением потока событий.

Структура данных уведомления основана на определении элемента <notification> в разделе 4 [RFC5277]. Она **должна** соответствовать схеме для элемента <notification> из раздела 4 [RFC5277] с использованием пространства имён XML, определённого в XSD как показано ниже.

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

При кодировании JSON именем модуля для элемента notification является ietf-restconf.

В контейнере notification предполагаются два дочерних узла, представляющих время события и его данные. Узел eventTime определён в том же пространстве имён XML, что и элемент <notification>. Для кодирования JSON используется пространство имён модуля ietf-restconf.

Имя и пространство имён элемента данных определяется модулем YANG, содержащим оператор notification-stmt, который представляет уведомления.

В приведённом ниже примере используется модуль YANG example-mod.

```
module example-mod {
  namespace "http://example.com/event/1.0";
  prefix ex;

  organization "Example, Inc.";
  contact "support at example.com";
  description "Пример модуля модели данных уведомления.";
  revision "2016-07-07" {
```

```
        description "Первый выпуск.";
        reference "example.com document 2-9976.";
      }

      notification event {
        description "Пример уведомления о событии.";
        leaf event-class {
          type string;
          description "Идентификатор класса события.";
        }
        container reporting-entity {
          description "Информация для конкретного события.";
          leaf card {
            type string;
            description "Идентификатор линейной платы.";
          }
        }
        leaf severity {
          type string;
          description "Описание важности события.";
        }
      }
    }
```

Пример уведомления SSE для события в формате XML может иметь вид

```
data: <notification
data:    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
data:    <eventTime>2013-12-21T00:01:00Z</eventTime>
data:    <event xmlns="http://example.com/event/1.0">
data:       <event-class>fault</event-class>
data:       <reporting-entity>
data:           <card>Ethernet0</card>
data:       </reporting-entity>
data:       <severity>major</severity>
data:     </event>
data: </notification>
```

Пример уведомления SSE для события в формате JSON может иметь вид

```
data: {
data:   "ietf-restconf:notification" : {
data:     "eventTime" : "2013-12-21T00:01:00Z",
data:     "example-mod:event" : {
data:       "event-class" : "fault",
data:       "reporting-entity" : { "card" : "Ethernet0" },
data:       "severity" : "major"
data:     }
data:   }
data: }
```

Поскольку оба варианта кодирования игнорируют пробельные символа, их можно представить одной строкой

XML

```
data: <notification xmlns="urn:ietf:params:xml:ns:netconf:notif\
ication:1.0"><eventTime>2013-12-21T00:01:00Z</eventTime><event \
xmlns="http://example.com/event/1.0"><event-class>fault</event-\
class><reportingEntity><card>Ethernet0</card></reporting-entity>\
<severity>major</severity></event></notification>
```

JSON

```
data: {"ietf-restconf:notification":{"eventTime":"2013-12-21\
T00:01:00Z","example-mod:event":{"event-class": "fault","repor\
tingEntity":{"card":"Ethernet0"},"severity":"major"}}}
```

Спецификация SSE поддерживает дополнительные поля event, id и retry. Сервер RESTCONF **может** передавать поле retry и в таких случаях клиентам RESTCONF **следует** применять его. Серверу RESTCONF **не следует** передавать поля event и id, поскольку нет осмысленных значений, которые могут использоваться для них, не дублируя содержимого самих уведомлений. Серверам RESTCONF, не передающим поле id, не требуется также поддержка поля заголовка HTTP Last-Event-ID [W3C.REC-eventsource-20150203]. Серверам RESTCONF, передающим поле id, **следует** поддерживать параметр start-time как предпочтительный способ задания клиентом точки воспроизведения потока событий.

7. Информация об ошибках

Для отчёта об успехе или отказе операций RESTCONF используются коды статуса HTTP. Информация об ошибках, которую отклики NETCONF содержат в элементе <rpc-error>, адаптирована для применения в RESTCONF и информация дерева данных <errors> возвращается для кодов состояния 4xx и 5xx.

Поскольку ресурс операций определён с оператором YANG rpc, а ресурс действий - с оператором YANG action, требуется отображение значений NETCONF <error-tag> на коды состояния HTTP. Конкретные теги ошибок и коды отклика зависят от используемой модели и могут указываться в операторах YANG description для action и rpc.

Сопоставление <error-tag> и Status Code.

Тег ошибки	***Код состояния***
in-use	409
invalid-value	400, 404, 406

(запрос) too-big	413
(отклик) too-big	400
missing-attribute	400
bad-attribute	400
unknown-attribute	400
bad-element	400
unknown-element	400
unknown-namespace	400
access-denied	401 или 403
lock-denied	409
resource-denied	409
rollback-failed	500
data-exists	409
data-missing	409
operation-not-supported	405 или 501
operation-failed	412 или 500
partial-operation	500
malformed-message	400

7.1. Сообщения с откликом на ошибку

При возникновении ошибки для запросного сообщения к любому типу ресурса и возврате кода 4xx (кроме «403 Forbidden») серверу **следует** включать в отклик message-body с информацией, описанной шаблоном данных YANG yang-errors из модуля ietf-restconf, определённого в разделе 8. Поле Content-Type в таких откликах должно иметь значение application/yang-data и может включать суффикс имени структурированного синтаксиса.

Клиенту **следует** указать желаемое кодирование (можно несколько) откликов путём задания подходящих типов носителей в заголовке Accept. Если клиент не указал заголовок Accept, **следует** использовать суффикс имени структурированного синтаксиса из запроса или сервер **может** выбрать поддерживаемый вариант кодирования сообщений. Если сообщения с запросом нет, сервер **должен** выбрать предпочтительный для него вариант из application/yang-data+xml и application/yang-data+json. Во всех примерах этого документа, за исключением приведенного ниже, предполагается возврат в случае ошибки сообщений с кодированием XML.

Дерево YANG для данных <errors> показано ниже.

```
+---- errors
      +---- error*
         +---- error-type       enumeration
         +---- error-tag        string
         +---- error-app-tag?   string
         +---- error-path?      instance-identifier
         +---- error-message?   string
         +---- error-info?
```

Семантика и синтаксис сообщений об ошибках RESTCONF определены в модуле YANG yang-errors (раздел 8).

Приведённый ниже пример показывает ошибку lock-denied, которая возникает, если клиент NETCONF уже заблокировал ресурс. Клиент RESTCONF пытается удалить ресурс данных. Отметим, что поле заголовка Accept здесь указывает желаемую кодировку для сообщения об ошибке. После успешной операции message-body не будет в отклике.

```
DELETE /restconf/data/example-jukebox:jukebox/\
   library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 409 Conflict
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "ietf-restconf:errors" : {
    "error" : [
      {
        "error-type" : "protocol",
        "error-tag" : "lock-denied",
        "error-message" : "Отказ в блокировке - уже заблокировано"
      }
    ]
  }
}
```

Приведённый ниже пример показывает ошибку data-exists для ресурса данных. Ресурс jukebox уже имеется и не может быть создан по запросу клиента.

```
POST /restconf/data/example-jukebox:jukebox HTTP/1.1
Host: example.com
```

Сервер может возвратить отклик

```
HTTP/1.1 409 Conflict
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+xml

<errors xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
```

```
      <error>
        <error-type>protocol</error-type>
        <error-tag>data-exists</error-tag>
        <error-path
          xmlns:rc="urn:ietf:params:xml:ns:yang:ietf-restconf"
          xmlns:jbox="https://example.com/ns/example-jukebox">\
          /rc:restconf/rc:data/jbox:jukebox
        </error-path>
        <error-message>
          Данные уже имеются и создать новый ресурс нельзя
        </error-message>
      </error>
    </errors>
```

8. Модуль RESTCONF

Модуль ietf-restconf содержит концептуальные определения в расширении и двух группировках, которые не предназначены для реализации в качестве содержимого хранилища данных на сервере. Например, контейнер restconf не предназначен для реализации в качестве узла данных верхнего уровня (под /restconf/data URI).

Отметим, что модуль ietf-restconf не имеет доступных для протокола объектов, поэтому дерево YANG не приводится.

```
<CODE BEGINS>

file "ietf-restconf@2017-01-26.yang"

module ietf-restconf {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf";
  prefix "rc";

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

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Author:   Andy Bierman <mailto:andy@yumaworks.com>

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

     Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

  description
    "Этот модуль содержит концептуальные спецификации YANG для
     определений базовых типов носителей RESTCONF, используемых
     в сообщениях протокола RESTCONF.

     Отметим, что определения YANG в этом модуле не представляют
     данных конфигурации. Оператор расширения restconf-media-type
     обеспечивает нормативный синтаксис для кодирования сообщений
     с использованием XML и JSON.

     Авторские права (Copyright (c) 2017) принадлежат IETF Trust
     и лицам, указанным как авторы кода. Все права защищены.

     Распространение и использование в исходной и двоичной форме
     с изменениями или без них разрешается в соответствии с условиями,
     указанными в упрощённой лицензии BSD, изложенной в разделе 4.c
     Правового положения IETF Trust применительно к документам IETF
     (http://trustee.ietf.org/license-info).

     Эта версия модуля YANG является частью RFC 8040, где
     правовые аспекты выражены более полно.";

  revision 2017-01-26 {
    description
      "Первый выпуск.";
    reference
      "RFC 8040: RESTCONF Protocol.";
  }

  extension yang-data {
    argument name {
      yin-element true;
    }
    description
      "Это расширение служит для задания шаблона данных YANG,
       представляющего концептуальные данные, определённые в YANG.
       Это предназначено для описания иерархических данных,
       независимых от контекста протокола и формата кодирования
       сообщений. Операторы определения данных в yang-data задают
       базовый синтаксис для конкретного шаблона данных YANG,
       имя которого является аргументом оператора yang-data.

       Отметим, что это расширение не определяет тип носителя.
```

```
        Спецификация применения этого расширения ДОЛЖНА указывать
        правила кодирования сообщений, включая тип носителя.

        Значение обязательного параметра name указывает шаблон
        данных YANG, который будет определён и служит именем шаблона.

        Это расширение игнорируется, если оно не указано в операторе
        верхнего уровня. Оно ДОЛЖНО содержать операторы определения
        данных, которые создают в точности одно определение узла-
        контейнера. Экземпляр шаблона данных YANG может транслироваться
        в экземпляр документа XML, чей элемент верхнего уровня
        соответствует контейнеру верхнего уровня.

        Значения имени и пространства имён для модуля YANG,
        применяющего оператор расширения назначаются для экземпляра
        документа данных, соответствующего оператору определения
        данных в этом расширении.

        Субоператоры в этом расширении ДОЛЖНЫ следовать правилу
        data-def-stmt в YANG ABNF.

        Корень документа XPath является самим оператором расширения,
        так что дочерние узлы корня документа представляются
        субоператорами data-def-stmt в этом расширении. Этот
        концептуальный документ является контекстом для операторов:
          - must-stmt
          - when-stmt
          - path-stmt
          - min-elements-stmt
          - max-elements-stmt
          - mandatory-stmt
          - unique-stmt
          - ordered-by
          - instance-identifier data type

        Приведённые ниже субоператоры data-def-stmt ограничены при
        использовании в операторе расширения yang-data.
          - От list-stmt не требуется наличия определённого key-stmt.
          - if-feature-stmt игнорируется при наличии.
          - config-stmt игнорируется при наличии.
          - Доступные значения для identity любых листьев и листьев-
            списков identityref ограничены модулем, содержащим
            этот оператор расширения и модулями, импортированными
            в этот модуль.";
    }

    rc:yang-data yang-errors {
      uses errors;
    }

    rc:yang-data yang-api {
      uses restconf;
    }

    grouping errors {
      description
        "Группировка, содержащая контейнер YANG, представляющий
         синтаксис и семантику отчётов об ошибках YANG Patch в
         сообщениях с откликами.";

      container errors {
        description
          "Представляет отчёт об ошибке, возвращаемый сервером,
           если запрос вызывает ошибку.";

        list error {
          description
            "Запись, содержащая информацию о конкретной ошибке
             при обработке запроса RESTCONF.";
          reference
            "RFC 6241, Section 4.3.";

          leaf error-type {
            type enumeration {
              enum transport {
                description
                  "Транспортный уровень.";
              }
              enum rpc {
                description
                  "Уровень rpc или notification.";
              }
              enum protocol {
                description
                  "Уровень протокольных операций.";
              }
```

```
            enum application {
              description
                "Уровень серверного приложения.";
              }
            }
            mandatory true;
            description
              "Уровень протокола, где произошла ошибка.";
          }

          leaf error-tag {
            type string;
            mandatory true;
            description
              "Перечисляемое значение error-tag.";
          }

          leaf error-app-tag {
            type string;
            description
              "Связанное с приложением значение error-tag.";
          }

          leaf error-path {
            type instance-identifier;
            description
              "Идентификатор экземпляра YANG, связанного с узлом ошибки.";
          }

          leaf error-message {
            type string;
            description
              "Сообщение, описывающее ошибку.";
          }

          anydata error-info {
             description
               "Это значение anydata ДОЛЖНО представлять контейнер,
                который может содержать узлы данных с дополнительными
                сведениями об ошибке.";
          }
        }
      }
    }

    grouping restconf {
      description
        "Концептуальная группировка, представляющая корневой ресурс
         RESTCONF.";

      container restconf {
        description
          "Концептуальный контейнер, представляющий корневой ресурс
           RESTCONF.";

        container data {
          description
            "Контейнер, представляющий ресурс хранилища данных.
             Представляет концептуальный корень всех данных состояния
             и конфигурации, поддерживаемых сервером. Дочерними
             узлами этого контейнера могут быть любые ресурсы данных,
             которые определены как узлы данных верхнего уровня, из
             модулей YANG, анонсируемых сервером в модуле
             'ietf-yang-library'.";
        }

        container operations {
          description
            "Контейнер для всех ресурсов операций.

             Каждый ресурс представляется как пустой лист с именем
             операции RPC из оператора YANG 'rpc'.

             Например, операция RPC 'system-restart', определённая
             в модуле 'ietf-system', будет представлена как пустой
             лист в пространстве имён 'ietf-system'. Это
             концептуальный лист, который реально не будет найден
             в модуле:
                module ietf-system {
                  leaf system-reset {
                    type empty;
                  }
                }

             Для вызова операции RPC 'system-restart'
                POST /restconf/operations/ietf-system:system-restart
```

```
            Для определения поддерживаемых сервером операций RPC
               GET /restconf/operations

            В XML пространство имён YANG указывает модуль
              <system-restart
                 xmlns='urn:ietf:params:xml:ns:yang:ietf-system'/>

            В JSON имя модуля YANG указывает модуль
              { 'ietf-system:system-restart' : [null] }
           ";
        }

        leaf yang-library-version {
          type string {
            pattern '\d{4}-\d{2}-\d{2}';
          }
          config false;
          mandatory true;
          description
            "Указывает дату выпуска модуля 'ietf-yang-library',
             реализованного этим сервером RESTCONF. Указывает
             год, месяц и число в числовом формате YYYY-MM-DD.";
        }
      }
    }

  }

  <CODE ENDS>
```

9. Мониторинг RESTCONF

Модуль ietf-restconf-monitoring обеспечивает информацию о возможностях протокола RESTCONF и потоках событий, доступных на сервере. Сервер RESTCONF **должен** реализовать модуль ietf-restconf-monitoring.

Ниже приведено дерево YANG для модуля ietf-restconf-monitoring.

```
     +--ro restconf-state
        +--ro capabilities
        |  +--ro capability*   inet:uri
        +--ro streams
           +--ro stream* [name]
              +--ro name                        string
              +--ro description?                string
              +--ro replay-support?             boolean
              +--ro replay-log-creation-time?   yang:date-and-time
              +--ro access* [encoding]
                 +--ro encoding  string
                 +--ro location  inet:uri
```

9.1. Контейнер restconf-state/capabilities

Этот обязательный контейнер содержит идентификаторы URI поддерживаемых сервером возможностей протокола RESTCONF.

Сервер **может** поддерживать временную метку для последнего изменения этого контейнера и возвращать поле заголовка Last-Modified, когда этот узел данных извлекается с помощью метода GET или HEAD. Отметим, что изменения в этом субдереве не оказывают влияния на временную метку последнего изменения хранилища данных.

Серверу **следует** поддерживать entity-tag для этого контейнера и возвращать поле заголовка Etag, когда этот узел данных извлекается с помощью метода GET или HEAD. Отметим, что изменения в этом субдереве не меняют entity-tag для ресурса хранилища.

Сервер **должен** включать запись URI листа-списка capability для режима defaults, используемого сервером, как определено в параграфе 9.1.2.

Сервер **должен** включать запись URI листа-списка capability, указывающую каждую дополнительную возможность, поддерживаемую сервером. Это включает дополнительные параметры запросов и **может** включать URI других возможностей, не определённых в этом документе.

9.1.1. Идентификаторы URI для параметров запроса

Определён новый набор идентификаторов URI для RESTCONF, указывающих параметры запроса (параграф 4.8), поддерживаемые сервером.

Сервер **должен** включать запись листа-списка capability для каждого поддерживаемого им необязательного параметра.

URI параметров запроса RESTCONF.

Имя	*Параграф*	*URI*
depth	4.8.2	urn:ietf:params:restconf:capability:depth:1.0
fields	4.8.3	urn:ietf:params:restconf:capability:fields:1.0
filter	4.8.4	urn:ietf:params:restconf:capability:filter:1.0
replay	4.8.7	urn:ietf:params:restconf:capability:replay:1.0
with-defaults	4.8.9	urn:ietf:params:restconf:capability:with-defaults:1.0

9.1.2. Идентификатор URI для возможности defaults

Это значение URI указывает принятый по умолчанию режим basic-mode, используемый сервером для обработки заданных по умолчанию листьев в запросах для ресурсов данных. Этот идентификатор URI для возможности протокола **должен** поддерживаться сервером и **должен** указываться в листе-списке capability, определённом в параграфе 9.3.

URI для возможности defaults.

Имя	*URI*
defaults	urn:ietf:params:restconf:capability:defaults:1.0

Идентификатор URI **должен** содержать параметр запроса basic-mode с одним из приведённых в таблице значений.

Значение	*Описание*
report-all	Нет узлов данных, рассматриваемых по умолчанию
trim	По умолчанию рассматриваются значения, заданные YANG default-stmt
explicit	Установленные клиентом значения никогда не рассматриваются по умолчанию

Определения basic-mode даны в документе «With-defaults Capability for NETCONF» [RFC6243].

При установке basic-mode=report-all сервер **должен** придерживаться режима принятой по умолчанию обработки, определённого в параграфе 2.1 [RFC6243].

При установке basic-mode=trim сервер **должен** придерживаться режима принятой по умолчанию обработки, определённого в параграфе 2.2 [RFC6243].

При установке basic-mode=explicit сервер **должен** придерживаться режима принятой по умолчанию обработки, определённого в параграфе 2.3 [RFC6243].

Ниже приведён пример запроса.

```
urn:ietf:params:restconf:capability:defaults:1.0?basic-mode=explicit
```

9.2. Контейнер restconf-state/streams

Этот необязательный контейнер обеспечивает доступ к потокам событий, поддерживаемым сервером. Сервер **может** опускать этот контейнер, если потоки событий не поддерживаются.

Сервер будет заполнять контейнер записями списка stream для каждого поддерживаемого типа потока. Каждый поток содержит лист events, в котором указывается идентификатор URI, представляющий ресурс потока событий.

Ресурсы потоков определены в параграфе 3.8, уведомления - в разделе 6.

9.3. Модуль мониторинга RESTCONF

Модуль ietf-restconf-monitoring определяет информацию мониторинга для протокола RESTCONF.

Модули ietf-yang-types и ietf-inet-types из [RFC6991] используются этим модулем для некоторых определений типов.

```
<CODE BEGINS>

file "ietf-restconf-monitoring@2017-01-26.yang"

module ietf-restconf-monitoring {
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring";
  prefix "rcmon";

  import ietf-yang-types { prefix yang; }
  import ietf-inet-types { prefix inet; }

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

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Author:   Andy Bierman <mailto:andy@yumaworks.com>

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

     Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

  description
    "Этот модуль содержит информацию мониторинга для протокола RESTCONF.

     Авторские права (Copyright (c) 2017) принадлежат IETF Trust
     и лицам, указанным как авторы кода. Все права защищены.

     Распространение и использование в исходной и двоичной форме
     с изменениями или без них разрешается в соответствии с условиями,
     указанными в упрощённой лицензии BSD, изложенной в разделе 4.c
     Правового положения IETF Trust применительно к документам IETF
     (http://trustee.ietf.org/license-info).

     Эта версия модуля YANG является частью RFC 8040, где
     правовые аспекты выражены более полно.";

  revision 2017-01-26 {
```

```
      description
        "Первая версия.";
      reference
        "RFC 8040: RESTCONF Protocol.";
    }

    container restconf-state {
      config false;
      description
        "Содержит информацию мониторинга для протокола RESTCONF.";

      container capabilities {
        description
          "Содержит список URI для возможностей протокола.";

        leaf-list capability {
          type inet:uri;
          description
            "Идентификатор URI протокольной возможности RESTCONF.";
        }
      }

      container streams {
        description
          "Контейнер, представляющий потоки уведомлений о событиях,
           поддерживаемых сервером.";
         reference
           "RFC 5277, Section 3.4, <streams> element.";

        list stream {
          key name;
          description
            "Каждая запись описывает поддерживаемый сервером поток
             событий.";

          leaf name {
            type string;
            description
              "Имя потока.";
            reference
              "RFC 5277, Section 3.4, <name> element.";
          }

          leaf description {
            type string;
            description
              "Описание содержимого потока.";
            reference
              "RFC 5277, Section 3.4, <description> element.";
          }

          leaf replay-support {
            type boolean;
            default false;
            description
              "Указывает поддерживается ли буфер повторного воспроизведения
               (replay) для этого потока. Значение true, говорит, что сервер
               ДОЛЖЕН поддерживать параметры запроса 'start-time' и 'stop-time'
               для этого потока.";
            reference
              "RFC 5277, Section 3.4, <replaySupport> element.";
          }

          leaf replay-log-creation-time {
            when "../replay-support" {
              description
                "Присутствует только при поддержке повтора уведомлений (replay.";
            }
            type yang:date-and-time;
            description
              "Указывает время создания журнала повторов для этого потока.";
            reference
              "RFC 5277, Section 3.4, <replayLogCreationTime> element.";
          }

          list access {
            key encoding;
            min-elements 1;
            description
              "Сервер будет создавать в этом списке запись для каждого формата
               кодирования, поддерживаемого для этого потока. Для всех потоков
               событий предполагается тип носителя text/event-stream. Этот
               список указывает субтипы, поддерживаемые для этого потока.";

            leaf encoding {
              type string;
```

```
            description
              "Это вторичный формат кодирования внутри text/event-stream,
               используемого всеми потоками. Тип xml поддерживается для
               кодирования XML, тип json - для кодирования JSON.";
          }

          leaf location {
            type inet:uri;
            mandatory true;
            description
              "Содержит URL для представления точки входа при организации
               уведомлений с помощью передаваемых сервером событий.";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

10. Библиотечный модуль YANG

Модуль ietf-yang-library, определённый в [RFC7895], обеспечивает информацию о модулях и субмодулях YANG, используемых сервером RESTCONF. Реализация этого модуля обязательна для серверов RESTCONF. Все модули и субмодули YANG, используемые сервером, **должны** быть указаны в библиотечном модуле YANG.

10.1. Список modules-state/module

Этот обязательный список содержит по одной записи для каждого модуля модели данных YANG, поддерживаемого сервером. Экземпляр этого списка **должен** существовать для каждого модуля YANG, используемого сервером.

Содержимого этого списка определено в операторе YANG module документа [RFC7895].

Отметим, что в модуле ietf-restconf нет доступных протоколу объектов для реализации, но сервер может указать модуль ietf-restconf в библиотеке YANG, если он импортируется (напрямую или опосредованно) реализованным модулем.

11. Взаимодействие с IANA

11.1. Тип связи restconf

Эта спецификация регистрирует тип связи restconf в реестре «Link Relation Types», определённом в [RFC5988].

```
Relation Name: restconf
Description: Указывает корень RESTCONF API на данном сервере
             HTTP. Связь restconf задаёт корень API, определённого
             в RFC 8040. Последующие выпуски RESTCONF будут
             применять другие значения связи для поддержки версий.
Reference: RFC 8040
```

11.2. Регистрации новых URI и модулей YANG

Этот документ регистрирует два идентификатора URI как пространства имён в реестре «IETF XML Registry» [RFC3688].

```
URI: urn:ietf:params:xml:ns:yang:ietf-restconf
Registrant Contact: The IESG.
XML: неприменимо; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring
Registrant Contact: The IESG.
XML: неприменимо; the requested URI is an XML namespace.
```

Документ регистрирует два модуля YANG в реестре «YANG Module Names» [RFC6020].

```
name:         ietf-restconf
namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf
prefix:       rc
reference:    RFC 8040

name:         ietf-restconf-monitoring
namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring
prefix:       rcmon
reference:    RFC 8040
```

11.3. Типы носителя

11.3.1. Тип application/yang-data+xml

Имя типа: application

Имя субтипа: yang-data+xml

Требуемые параметры: нет

Необязательные параметры: нет

Кодирование: 8 битов

Каждый концептуальный узел данных YANG кодируется в соответствии с правилами XML и каноническим форматом для конкретного типа узла YANG, определенным в [RFC7950].

Вопросы безопасности: Вопросы безопасности, связанные с генерацией и употреблением сообщений RESTCONF, рассмотрены в разделе 12 RFC 8040. С семантикой конкретных моделей YANG связаны дополнительные вопросы безопасности. Предполагается, что в каждом модуле YANG будут рассматриваться вопросы безопасности, связанные с определёнными в этом модуле данными.

Вопросы взаимодействия: В RFC 8040 заданы форматы сообщение и их интерпретация.

Опубликованная спецификация: RFC 8040

Приложения, использующие этот тип носителя: Анализаторы данных в экземплярах документов, применяемые протоколом или средствами автоматизации для работы со структурами данных YANG.

Идентификация фрагментов: Идентификаторы фрагментов для этого типа не определены. Все узлы данных YANG доступны как ресурсы с использованием пути в URI запроса.

Дополнительная информация:

Запрещённые имена псевдонимов: неприменимо

«Магическое» значение: неприменимо

Расширения имён: нет

Коды типа файлов Macintosh: "TEXT"

Контактные данные: См. раздел «Адреса авторов» в RFC 8040

Предполагаемое применение: общего назначения

Ограничения на использование: неприменимо

Автор: См. раздел «Адреса авторов» в RFC 8040

Контроль изменений: Internet Engineering Task Force (mailto:iesg@ietf.org).

Предварительная регистрация (только дерево стандартов): нет

11.3.2. Тип application/yang-data+json

Имя типа: application

Имя субтипа: yang-data+json

Требуемые параметры: нет

Необязательные параметры: нет

Кодирование: 8 битов

Каждый концептуальный узел данных YANG кодируется в соответствии с [RFC7951]. Аннотации метаданных кодируются в соответствии с [RFC7952].

Вопросы безопасности: Вопросы безопасности, связанные с генерацией и употреблением сообщений RESTCONF, рассмотрены в разделе 12 RFC 8040. С семантикой конкретных моделей YANG связаны дополнительные вопросы безопасности. Предполагается, что в каждом модуле YANG будут рассматриваться вопросы безопасности, связанные с определёнными в этом модуле данными.

Вопросы взаимодействия: В RFC 8040 заданы форматы сообщение и их интерпретация.

Опубликованная спецификация: RFC 8040

Приложения, использующие этот тип носителя: Анализаторы данных в экземплярах документов, применяемые протоколом или средствами автоматизации для работы со структурами данных YANG.

Идентификация фрагментов: Синтаксис и семантика фрагментов совпадают с заданными для типа носителя application/json.

Дополнительная информация:

Запрещённые имена псевдонимов: неприменимо

"Магическое" значение: неприменимо

Расширения имён: нет

Коды типа файлов Macintosh: "TEXT"

Контактные данные: См. раздел «Адреса авторов» в RFC 8040

Предполагаемое применение: общего назначения

Ограничения на использование: неприменимо

Автор: См. раздел «Адреса авторов» в RFC 8040

Контроль изменений: Internet Engineering Task Force (mailto:iesg@ietf.org).

Предварительная регистрация (только дерево стандартов): нет

11.4. Реестр возможностей RESTCONF

Этот документ определяет реестр для идентификаторов возможностей RESTCONF с именем «RESTCONF Capability URNs». Значения в реестр вносятся по процедуре IETF Review [RFC5226]. Каждая запись реестра должна включать:

- имя возможности RESTCONF - по соглашению имена начинаются с двоеточия (:);
- идентификатор URN для возможности RESTCONF;
- ссылку на документ, регистрирующий значение.

Этот документ регистрирует несколько идентификаторов возможностей в реестре «RESTCONF Capability URNs».

Индекс	*Идентификатор возможности*
:defaults	urn:ietf:params:restconf:capability:defaults:1.0
:depth	urn:ietf:params:restconf:capability:depth:1.0
:fields	urn:ietf:params:restconf:capability:fields:1.0
:filter	urn:ietf:params:restconf:capability:filter:1.0
:replay	urn:ietf:params:restconf:capability:replay:1.0
:with-defaults	urn:ietf:params:restconf:capability:with-defaults:1.0

11.5. Регистрация субпространства имён URN restconf

Агентство IANA зарегистрировало новое субпространство имён URN в реестре «IETF URN Sub-namespace for Registered Protocol Parameter Identifiers», заданном [RFC3553].

```
Registry Name: restconf
Specification: RFC 8040
Repository: Реестр "RESTCONF Capability URNs" (параграф 11.4)
Index value: Субпараметры ДОЛЖНЫ указываться в кодировке UTF-8
   с применением стандартного кодирования URI, когда это нужно.
```

12. Вопросы безопасности

В параграфе 2.1 сказано, что сервер RESTCONF **должен** поддерживать протокол TLS [RFC5246]. Это позволяет предположить возможность поддержки сервером RESTCONF будущих версий протокола TLS. В частности, TLS 1.3 [TLS1.3] добавляет поддержку согласования 0-RTT, которая может создавать проблемы безопасности для RESTCONF API, как описано в Приложении B.1 к документу TLS 1.3. Поэтому серверам RESTCONF **рекомендуется** не поддерживать 0-RTT совсем (даже для идемпотентных запросов) , пока обновление этого RFC не укажет иного.

В параграфе 2.5 рекомендуется аутентификация на основе клиентских сертификатов TLS, но разрешено применять любые схемы проверки подлинности, указанные в реестре «Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry». Разработчикам следует принимать во внимание, что строгость этих методов существенно различается, а некоторые методы могут быть экспериментальными. Выбирать любую из этих схем **следует** лишь после прочтения раздела «Вопросы безопасности» в RFC, связанном с соответствующей записью реестра.

Модуль YANG ietf-restconf-monitoring, определённый в этом документе, предназначен для доступа по протоколу NETCONF [RFC6241]. Нижним уровнем NETCONF является защищённый транспорт и обязательна реализация SSH[1] [RFC6242]. Модель контроля доступа NETCONF [RFC6536] обеспечивает способы ограничения доступа для конкретных пользователей NETCONF к заранее заданному набору доступных протокольных операций и содержимого NETCONF.

Нижним уровнем RESTCONF является HTTPS с обязательной реализацией защищённого транспорта TLS [RFC5246]. Протокол RESTCONF использует модель контроля доступа NETCONF [RFC6536], которая позволяет ограничить доступ для конкретных пользователей RESTCONF к заранее заданному набору доступных протокольных операций и содержимого RESTCONF.

В этом разделе рассматриваются вопросы безопасности для ресурсов, определяемых протоколом RESTCONF. Вопросы безопасности HTTPS рассмотрены в [RFC7230]. RESTCONF не задаёт модулей YANG, которые сервер должен поддерживать, помимо модулей ietf-restconf-monitoring (раздел 9) и ietf-yang-library (раздел 10). Вопросы безопасности для других модулей, с которыми работает RESTCONF, будут рассмотрены в документах, определяющих эти модули YANG.

Данные конфигурации конфиденциальны по своей природе. Их передача в открытом виде без контроля целостности открывает устройства для классических атак с перехватом и подменой данных. Конфигурационная информация зачастую включает пароли, имена пользователей, описания услуг и топологические сведения, которые не следует разглашать. Имеется множество атак, которые известны из практики применения существующих интерфейсов управления. Было бы разумно для разработчиков изучить и применить этот опыт при реализации протокола.

В разных средах права до и после аутентификации могут различаться. Когда для операции RESTCONF не предоставлено нужных полномочий, сервер RESTCONF **должен** возвращать статус «401 Unauthorized». Отметим, что обмен данными о полномочиях может происходить в форме параметров конфигурации, что является ещё одним основанием для защиты соединения. Отметим, что клиент может обнаружить изменения конфигурации в ресурсах данных, к которым ему не предоставлен доступ, путём отслеживания изменений в полях заголовка Etag и Last-Modified, возвращаемых сервером для ресурса хранилища.

Реализациям сервера RESTCONF **следует** пытаться предотвратить нарушение работы системы в результате чрезмерного использования системных ресурсов при обработке запросов на редактирование с помощью методов POST, PUT и PATCH. Без этого возникает возможность атак с попыткой использовать всю доступную память и другие ресурсы системы.

[1] Secure Shell - защищённая оболочка.

13. Литература

13.1. Нормативные документы

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <http://www.rfc-editor.org/info/rfc2046>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <http://www.rfc-editor.org/info/rfc3553>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <http://www.rfc-editor.org/info/rfc5277>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <http://www.rfc-editor.org/info/rfc5789>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <http://www.rfc-editor.org/info/rfc6243>.

[RFC6415] Hammer-Lahav, E., Ed., and B. Cook, "Web Host Metadata", RFC 6415, DOI 10.17487/RFC6415, October 2011, <http://www.rfc-editor.org/info/rfc6415>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

[RFC7235] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <http://www.rfc-editor.org/info/rfc7235>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <http://www.rfc-editor.org/info/rfc7320>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <http://www.rfc-editor.org/info/rfc7589>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <http://www.rfc-editor.org/info/rfc7951>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <http://www.rfc-editor.org/info/rfc7952>.

[W3C.REC-eventsource-20150203] Hickson, I., "Server-Sent Events", World Wide Web Consortium Recommendation REC-eventsource-20150203, February 2015, <http://www.w3.org/TR/2015/REC-eventsource-20150203>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

[XPath] Clark, J. and S. DeRose, "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

[REST-Dissertation] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[TLS1.3] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress[1], draft-ietf-tls-tls13-18, October 2016.

[YANG-Patch] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", Work in Progress[2], draft-ietf-netconf-yang-patch-14, November 2016.

Приложение A. Пример модуля YANG

Пример модуля YANG, используемый в этом документе, представляет интерфейс с простым аудио-проигрывателем.

Ниже показано дерево YANG для модуля example-jukebox.

```
+--rw jukebox!
   +--rw library
   |  +--rw artist* [name]
   |  |  +--rw name      string
   |  |  +--rw album* [name]
   |  |     +--rw name      string
   |  |     +--rw genre?    identityref
   |  |     +--rw year?     uint16
   |  |     +--rw admin
   |  |     |  +--rw label?              string
   |  |     |  +--rw catalogue-number?   string
   |  |     +--rw song* [name]
   |  |        +--rw name        string
   |  |        +--rw location    string
   |  |        +--rw format?     string
   |  |        +--rw length?     uint32
   |  +--ro artist-count?   uint32
   |  +--ro album-count?    uint32
   |  +--ro song-count?     uint32
   +--rw playlist* [name]
   |  +--rw name           string
   |  +--rw description?   string
   |  +--rw song* [index]
   |     +--rw index    uint32
   |     +--rw id       instance-identifier
   +--rw player
      +--rw gap?   decimal64
```

[1]Работа опубликована в RFC 8446.
[2]Работа опубликована в RFC 8072.

```
rpcs:

+---x play
    +--ro input
       +--ro playlist       string
       +--ro song-number    uint32
```

A.1. Модуль YANG example-jukebox

```
module example-jukebox {

  namespace "http://example.com/ns/example-jukebox";
  prefix "jbox";

  organization "Example, Inc.";
  contact "support at example.com";
  description "Example Jukebox Data Model Module.";
  revision "2016-08-15" {
    description "Первый выпуск.";
    reference "example.com document 1-4673.";
  }

  identity genre {
    description
      "База для всех типов жанра.";
  }

  // Сокращённый список классификаций жанров
  identity alternative {
    base genre;
    description
      "Альтернативная музыка.";
  }
  identity blues {
    base genre;
    description
      "Блюзы.";
  }
  identity country {
    base genre;
    description
      "Народная музыка.";
  }
  identity jazz {
    base genre;
    description
      "Джаз.";
  }
  identity pop {
    base genre;
    description
      "Поп-музыка.";
  }
  identity rock {
    base genre;
    description
      "Рок.";
  }

  container jukebox {
    presence
      "Пустой контейнер, показывающий доступность проигрывателя.";

    description
      "Представляет ресурс jukebox с библиотекой, списками
       воспроизведения и операцией play (проиграть).";

    container library {

      description
        "Представляет ресурс библиотеки jukebox.";

      list artist {
        key name;
        description
          "Представляет один ресурс artist в ресурсе библиотеки
           jukebox.";

        leaf name {
          type string {
            length "1 .. max";
          }
          description
            "Имя исполнителя.";
        }
```

```
    list album {
      key name;
      description
        "Представляет один ресурс album внутри ресурса
         artist в библиотеке jukebox.";

      leaf name {
        type string {
          length "1 .. max";
        }
        description
          "Название альбома.";
      }
      leaf genre {
        type identityref { base genre; }
        description
          "Жанр музыки в альбоме.";
      }

      leaf year {
        type uint16 {
          range "1900 .. max";
        }
        description
          "год выпуска альбома.";
      }

      container admin {
        description
          "Административные данные для альбома.";

        leaf label {
          type string;
          description
            "Метка выпущенного альбома.";
        }
        leaf catalogue-number {
          type string;
          description
            "Каталожный номер альбома.";
        }
      }

      list song {
        key name;
        description
          "Представляет ресурс song внутри ресурса album
           в библиотеке jukebox.";

        leaf name {
          type string {
             length "1 .. max";
          }
          description
            "Название песни.";
        }
        leaf location {
          type string;
          mandatory true;
          description
            "Строка размещения файла с песней.";
        }
        leaf format {
          type string;
          description
            "Идентификатор строки типа носителя для файла,
             связанного с листом location для этой записи.";
        }
        leaf length {
          type uint32;
          units "seconds";
          description
            "Продолжительность песни в секундах.";
        }
      }   // Конец списка song
    }   // Конец списка album
  }   // Конец списка artist

  leaf artist-count {
     type uint32;
     units "artists";
     config false;
     description
       "Число исполнителей в библиотеке.";
  }
```

```
          leaf album-count {
             type uint32;
             units "albums";
             config false;
             description
               "Число альбомов в библиотеке.";
          }
          leaf song-count {
             type uint32;
             units "songs";
             config false;
             description
               "Число песен в библиотеке.";
          }
        }  // Конец библиотеки

        list playlist {
          key name;
          description
            "Пример ресурса данных конфигурации.";

          leaf name {
            type string;
            description
              "Название списка воспроизведения.";
          }
          leaf description {
            type string;
            description
              "Комментарий с описанием списка воспроизведения.";
          }
          list song {
            key index;
            ordered-by user;

            description
              "Пример вложенного ресурса данных конфигурации.";

            leaf index {    // Реально не требуется
              type uint32;
              description
                "Произвольный целочисленный индекс песни в списке воспроизведения.";
            }
            leaf id {
              type instance-identifier;
              mandatory true;
              description
                "Идентификатор песни. Должен указывать экземпляр
                 /jukebox/library/artist/album/song/name.";
            }
          }
        }

        container player {
          description
            "Представляет ресурс проигрывателя jukebox.";

          leaf gap {
            type decimal64 {
              fraction-digits 1;
              range "0.0 .. 2.0";
            }
            units "tenths of seconds";
            description
              "Интервал между песнями.";
          }
        }
      }

      rpc play {
        description
          "Функция управления для проигрывателя jukebox.";
        input {
          leaf playlist {
            type string;
            mandatory true;
            description
              "Название списка воспроизведения.";
          }
          leaf song-number {
            type uint32;
            mandatory true;
            description
              "Номер песни в списке воспроизведения для проигрывания.";
          }
        }
```

```
      }
    }
```

Приложение B. Примеры сообщений RESTCONF

Примеры в этом документе используют нормативный модуль YANG ietf-restconf, определённый в разделе 8 и ненормативный пример модуля YANG example-jukebox, который определён в Приложении A.1.

В этом приложении приведены примеры некоторых типовых обменов сообщениями RESTCONF.

B.1. Примеры извлечения ресурсов

B.1.1. Извлечение ресурса API верхнего уровня

Клиент начинает извлечение корневого ресурса RESTCONF

```
GET /.well-known/host-meta HTTP/1.1
Host: example.com
Accept: application/xrd+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Content-Type: application/xrd+xml
Content-Length: nnn

<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
    <Link rel='restconf' href='/restconf'/>
</XRD>
```

Затем клиент может извлечь ресурс API верхнего уровня, используя корневой ресурс /restconf.

```
GET /restconf HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "ietf-restconf:restconf" : {
    "data" : {},
    "operations" : {},
    "yang-library-version" : "2016-06-21"
  }
}
```

Для запроса представления содержимого отклика в формате XML можно применить поле заголовка Accept.

```
GET /restconf HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер будет возвращать те же данные, которые могут иметь вид

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+xml

<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <data/>
  <operations/>
  <yang-library-version>2016-06-21</yang-library-version>
</restconf>
```

B.1.2. Получение информации о модуле сервера

Возможно изменение модуля библиотеки YANG с течением времени. Клиент может узнать дату выпуска модуля ietf-yang-library, поддерживаемого сервером, из ресурса API, как описано в предыдущем параграфе.

В этом примере клиент извлекает информацию о модуле в формате JSON.

```
GET /restconf/data/ietf-yang-library:modules-state HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Last-Modified: Thu, 26 Jan 2017 14:00:14 GMT
Content-Type: application/yang-data+json

{
  "ietf-yang-library:modules-state" : {
    "module-set-id" : "5479120c17a619545ea6aff7aa19838b036ebbd7",
    "module" : [
```

```
        {
          "name" : "foo",
          "revision" : "2012-01-02",
          "schema" : "https://example.com/modules/foo/2012-01-02",
          "namespace" : "http://example.com/ns/foo",
          "feature" : [ "feature1", "feature2" ],
          "deviation" : [
            {
              "name" : "foo-dev",
              "revision" : "2012-02-16"
            }
          ],
          "conformance-type" : "implement"
        },
        {
          "name" : "ietf-yang-library",
          "revision" : "2016-06-21",
          "schema" : "https://example.com/modules/\
            ietf-yang-library/2016-06-21",
          "namespace" :
            "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type" : "implement"
        },
        {
          "name" : "foo-types",
          "revision" : "2012-01-05",
          "schema" :
            "https://example.com/modules/foo-types/2012-01-05",
          "namespace" : "http://example.com/ns/foo-types",
          "conformance-type" : "import"
        },

        {
          "name" : "bar",
          "revision" : "2012-11-05",
          "schema" : "https://example.com/modules/bar/2012-11-05",
          "namespace" : "http://example.com/ns/bar",
          "feature" : [ "bar-ext" ],
          "conformance-type" : "implement",
          "submodule" : [
            {
              "name" : "bar-submod1",
              "revision" : "2012-11-05",
              "schema" :
               "https://example.com/modules/bar-submod1/2012-11-05"
            },
            {
              "name" : "bar-submod2",
              "revision" : "2012-11-05",
              "schema" :
               "https://example.com/modules/bar-submod2/2012-11-05"
            }
          ]
        }
      ]
    }
  }
```

B.1.3. Извлечение данных о возможностях сервера

В этом примере клиент извлекает информацию о возможностях сервера в формате XML. Сервер поддерживает все параметры запросов RESTCONF и один фирменный параметр.

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/\
    capabilities HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Last-Modified: Thu, 26 Jan 2017 16:00:14 GMT
Content-Type: application/yang-data+xml

<capabilities
    xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">
 <capability>\
  urn:ietf:params:restconf:capability:defaults:1.0?\
     basic-mode=explicit\
 </capability>
 <capability>\
  urn:ietf:params:restconf:capability:with-defaults:1.0\
 </capability>
 <capability>\
  urn:ietf:params:restconf:capability:depth:1.0\
```

```
 </capability>
 <capability>\
  urn:ietf:params:restconf:capability:fields:1.0\
 </capability>
 <capability>\
  urn:ietf:params:restconf:capability:filter:1.0\
 </capability>
 <capability>\
  urn:ietf:params:restconf:capability:start-time:1.0\
 </capability>
 <capability>\
  urn:ietf:params:restconf:capability:stop-time:1.0\
 </capability>
 <capability>\
  http://example.com/capabilities/myparam\
 </capability>
</capabilities>
```

B.2. Примеры ресурсов данных и хранилища

B.2.1. Создание нового ресурса данных

Для создания нового ресурса artist в ресурсе library клиент может передать приведённый ниже запрос.

```
POST /restconf/data/example-jukebox:jukebox/library HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "example-jukebox:artist" : [
    {
      "name" : "Foo Fighters"
    }
  ]
}
```

Если ресурс создан, сервер может возвратить отклик

```
HTTP/1.1 201 Created
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Location: https://example.com/restconf/data/\
    example-jukebox:jukebox/library/artist=Foo%20Fighters
Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
ETag: "b3830f23a4c"
```

Чтобы создать новый ресурс album для этого исполнителя в ресурсе jukebox, клиент может передать запрос вида

```
POST /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<album xmlns="http://example.com/ns/example-jukebox">
  <name>Wasting Light</name>
  <year>2011</year>
</album>
```

Если ресурс создан, сервер может возвратить отклик

```
HTTP/1.1 201 Created
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Location: https://example.com/restconf/data/\
    example-jukebox:jukebox/library/artist=Foo%20Fighters/\
    album=Wasting%20Light
Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
ETag: "b8389233a4c"
```

B.2.2. Детектирование изменения тега объекта для хранилища

В этом примере сервер просто поддерживает временную метку последнего изменения хранилища данных. Предположим, что клиент кэшировал заголовок Last-Modified из отклика на предыдущий запрос. Это значение будет применяться как заголовок If-Unmodified-Since в следующем запросе для исправления (patch) записи списка album со значением ключа «Wasting Light». Обновляется лишь поле genre.

```
PATCH /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters/album=Wasting%20Light/\
    genre HTTP/1.1
Host: example.com
If-Unmodified-Since: Thu, 26 Jan 2017 20:56:30 GMT
Content-Type: application/yang-data+json

{ "example-jukebox:genre" : "example-jukebox:alternative" }
```

В этом примере ресурс хранилища данных был изменён с момента, указанного в заголовке If-Unmodified-Since, поэтому отклик сервера может иметь вид

```
HTTP/1.1 412 Precondition Failed
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Last-Modified: Thu, 26 Jan 2017 19:41:00 GMT
```

```
ETag: "b34aed893a4c"
```

B.2.3. Редактирование ресурса хранилища

В этом примере предполагается наличие в модуле example-system ресурса данных верхнего уровня по имени system и наличие у этого контейнера дочернего листа enable-jukebox-streaming.

```
container system {
  leaf enable-jukebox-streaming {
    type boolean;
  }
}
```

В этом примере клиент применяет метод PATCH для изменения сразу двух ресурсов верхнего уровня с целью включения поддержки потоков и добавления субресурса album к каждому из двух ресурсов artist.

```
PATCH /restconf/data HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<data xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <system xmlns="http://example.com/ns/example-system">
    <enable-jukebox-streaming>true</enable-jukebox-streaming>
  </system>
  <jukebox xmlns="http://example.com/ns/example-jukebox">
    <library>
      <artist>
        <name>Foo Fighters</name>
        <album>
          <name>One by One</name>
          <year>2012</year>
        </album>
      </artist>
      <artist>
        <name>Nick Cave and the Bad Seeds</name>
        <album>
          <name>Tender Prey</name>
          <year>1988</year>
        </album>
      </artist>
    </library>
  </jukebox>
</data>
```

B.2.4. Замена ресурса хранилища

В этом примере меняется все содержимое хранилища данных. Все узлы, отсутствующие в элементе <data>, но имеющиеся на сервер, будут удалены.

```
PUT /restconf/data HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<data xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <jukebox xmlns="http://example.com/ns/example-jukebox">
    <library>
      <artist>
        <name>Foo Fighters</name>
        <album>
          <name>One by One</name>
          <year>2012</year>
        </album>
      </artist>
      <artist>
        <name>Nick Cave and the Bad Seeds</name>
        <album>
          <name>Tender Prey</name>
          <year>1988</year>
        </album>
      </artist>
    </library>
  </jukebox>
</data>
```

B.2.5. Редактирование ресурса данных

В этом примере клиент меняет один узел данных, добавляя субресурс album методом PATCH для ресурса данных.

```
PATCH /restconf/data/example-jukebox:jukebox/library/\
   artist=Nick%20Cave%20and%20the%20Bad%20Seeds HTTP/1.1
Host: example.com
Content-Type: application/yang-data+xml

<artist xmlns="http://example.com/ns/example-jukebox">
  <name>Nick Cave and the Bad Seeds</name>
  <album>
    <name>The Good Son</name>
    <year>1990</year>
  </album>
</artist>
```

B.3. Примеры параметров запроса

B.3.1. Параметр content

Параметр content служит для выбора типов дочерних ресурсов данных (конфигурационный или не конфигурационный), которые будут возвращаться сервером для запроса GET.

В приведённом примере используется простой список YANG, имеющий конфигурационные и не конфигурационные дочерние ресурсы.

```
container events {
  list event {
    key name;
    leaf name { type string; }
    leaf description { type string; }
    leaf event-count {
      type uint32;
      config false;
    }
  }
}
```

Пример 1: content=all

Для получения всех дочерних ресурсов указывается параметр content=all или этот параметр опускается. Клиент может передать запрос

```
GET /restconf/data/example-events:events?\
    content=all HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "example-events:events" : {
    "event" : [
      {
        "name" : "interface-up",
        "description" : "Счётчик уведомлений о включении интерфейса",
        "event-count" : 42
      },
      {
        "name" : "interface-down",
        "description" : "Счётчик уведомлений о выключении интерфейса",
        "event-count" : 4
      }
    ]
  }
}
```

Пример 2: content=config

Для извлечения только конфигурационных дочерних ресурсов указывается content=config. Отметим, что в этом случае возвращаются лишь заголовки Etag и Last-Modified.

```
GET /restconf/data/example-events:events?\
    content=config HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Last-Modified: Thu, 26 Jan 2017 16:45:20 GMT
ETag: "eeeada438af"
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "example-events:events" : {
    "event" : [
      {
        "name" : "interface-up",
        "description" : "Счётчик уведомлений о включении интерфейса"
      },
      {
        "name" : "interface-down",
        "description" : "Счётчик уведомлений о выключении интерфейса"
      }
    ]
  }
}
```

Пример 3: content=nonconfig

Для извлечения только не связанных с конфигурацией дочерних ресурсов указывается content=nonconfig. Отметим, что в этом случае возвращаются также конфигурационные предки и ключевые листья, если они имеются. Клиент может передать запрос

```
GET /restconf/data/example-events:events?\
   content=nonconfig HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "example-events:events" : {
    "event" : [
      {
        "name" : "interface-up",
        "event-count" : 42
      },
      {
        "name" : "interface-down",
        "event-count" : 4
      }
    ]
  }
}
```

B.3.2. Параметр depth

Параметр depth служит для ограничения числа уровней дочерних ресурсов, возвращаемых сервером для запроса GET.

Счёт уровней начинается с целевого ресурса, поэтому при depth=1 возвращается только этот ресурс, а при глубине 2 возвращается целевой уровень и его ближайшие потомки.

В примере показано влияние значения depth на содержимое, извлекаемое из ресурса верхнего уровня jukebox.

Пример 1: depth=unbounded

Для извлечения всех дочерних ресурсов параметр depth не указывается или имеет принятое по умолчанию значение unbounded.

```
GET /restconf/data/example-jukebox:jukebox?\
    depth=unbounded HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "example-jukebox:jukebox" : {
    "library" : {
      "artist" : [
        {
          "name" : "Foo Fighters",
          "album" : [
            {
              "name" : "Wasting Light",
              "genre" : "example-jukebox:alternative",
              "year" : 2011,
              "song" : [
                {
                  "name" : "Wasting Light",
                  "location" :
                    "/media/foo/a7/wasting-light.mp3",
                  "format" : "MP3",
                  "length" : 286
                },
                {
                  "name" : "Rope",
                  "location" : "/media/foo/a7/rope.mp3",
                  "format" : "MP3",
                  "length" : 259
                }
              ]
            }
          ]
        }
```

```
          ]
        },
        "playlist" : [
          {
            "name" : "Foo-One",
            "description" : "Пример списка воспроизведения 1",
            "song" : [
              {
                "index" : 1,
                "id" : "/example-jukebox:jukebox/library\
                   /artist[name='Foo Fighters']\
                   /album[name='Wasting Light']\
                   /song[name='Rope']"
              },
              {
                "index" : 2,
                "id" : "/example-jukebox:jukebox/library\
                   /artist[name='Foo Fighters']\
                   /album[name='Wasting Light']\
                   /song[name='Bridge Burning']"
              }
            ]
          }
        ],
        "player" : {
          "gap" : 0.5
        }
      }
    }
```

Пример 2: depth=1

Для определения наличия одного или множества экземпляров ресурса для данной цели указывается глубина 1.

```
GET /restconf/data/example-jukebox:jukebox?depth=1 HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "example-jukebox:jukebox" : {}
}
```

Пример 3: depth=3

Для ограничения глубины целевым ресурсом и двумя дочерними уровнями используется значение 3.

```
GET /restconf/data/example-jukebox:jukebox?depth=3 HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "example-jukebox:jukebox" : {
    "library" : {
      "artist" : {}
    },
    "playlist" : [
      {
        "name" : "Foo-One",
        "description" : "Пример списка воспроизведения 1",
        "song" : {}
      }
    ],
    "player" : {
      "gap" : 0.5
    }
  }
}
```

B.3.3. Параметр fields

В этом примере клиент извлекает ресурс хранилища данных в формате JSON, но нужен лишь список modules-state/module, из каждой записи которого берутся только узлы name и revision. Отметим, что верхний узел, возвращённый сервером, соответствует узлу целевого ресурса (data в этом примере). Лист module-set-id не возвращается, поскольку он не выбран в выражении fields.

```
GET /restconf/data?fields=ietf-yang-library:modules-state/\
```

```
        module(name;revision) HTTP/1.1
    Host: example.com
    Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
    HTTP/1.1 200 OK
    Date: Thu, 26 Jan 2017 20:56:30 GMT
    Server: example-server
    Content-Type: application/yang-data+json

    {
      "ietf-restconf:data" : {
        "ietf-yang-library:modules-state" : {
          "module" : [
            {
              "name" : "example-jukebox",
              "revision" : "2016-08-15"
            },
            {
              "name" : "ietf-inet-types",
              "revision" : "2013-07-15"
            },
            {
              "name" : "ietf-restconf-monitoring",
              "revision" : "2017-01-26"
            },
            {
              "name" : "ietf-yang-library",
              "revision" : "2016-06-21"
            },
            {
              "name" : "ietf-yang-types",
              "revision" : "2013-07-15"
            }
          ]
        }
      }
    }
```

B.3.4. Параметр insert

В этом примере новая песня добавляется в начало списка воспроизведения Foo-One.

Запрос клиента

```
    POST /restconf/data/example-jukebox:jukebox/\
        playlist=Foo-One?insert=first HTTP/1.1
    Host: example.com
    Content-Type: application/yang-data+json

    {
      "example-jukebox:song" : [
        {
          "index" : 1,
          "id" : "/example-jukebox:jukebox/library\
             /artist[name='Foo Fighters']\
             /album[name='Wasting Light']\
             /song[name='Rope']"
        }
      ]
    }
```

Отклик сервера

```
    HTTP/1.1 201 Created
    Date: Thu, 26 Jan 2017 20:56:30 GMT
    Server: example-server
    Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
    Location: https://example.com/restconf/data/\
        example-jukebox:jukebox/playlist=Foo-One/song=1
    ETag: "eeeada438af"
```

B.3.5. Параметр point

В этом примере клиент вставляет запись для новой песни с список воспроизведения Foo-One после первой песни.

Запрос клиента

```
    POST /restconf/data/example-jukebox:jukebox/\
        playlist=Foo-One?insert=after&point=\
        %2Fexample-jukebox%3Ajukebox\
        %2Fplaylist%3DFoo-One%2Fsong%3D1 HTTP/1.1
    Host: example.com
    Content-Type: application/yang-data+json

    {
      "example-jukebox:song" : [
        {
          "index" : 2,
          "id" : "/example-jukebox:jukebox/library\
```

```
              /artist[name='Foo Fighters']\
              /album[name='Wasting Light']\
              /song[name='Bridge Burning']"
          }
        ]
      }
```

Отклик сервера

```
HTTP/1.1 201 Created
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Last-Modified: Thu, 26 Jan 2017 20:56:30 GMT
Location: https://example.com/restconf/data/\
    example-jukebox:jukebox/playlist=Foo-One/song=2
ETag: "abcada438af"
```

B.3.6. Параметр filter

Приведённые ниже идентификаторы URI показывают некоторые примеры задания фильтров уведомлений.

```
// filter = /event/event-class='fault'
GET /streams/NETCONF?filter=%2Fevent%2Fevent-class%3D'fault'

// filter = /event/severity<=4
GET /streams/NETCONF?filter=%2Fevent%2Fseverity%3C%3D4

// filter = /linkUp|/linkDown
GET /streams/SNMP?filter=%2FlinkUp%7C%2FlinkDown

// filter = /*/reporting-entity/card!='Ethernet0'
GET /streams/NETCONF?\
   filter=%2F*%2Freporting-entity%2Fcard%21%3D'Ethernet0'

// filter = /*/email-addr[contains(.,'company.com')]
GET /streams/critical-syslog?\
   filter=%2F*%2Femail-addr[contains(.%2C'company.com')]

// Примечание. Имя модуля используется в качестве префикса.
// filter = (/example-mod:event1/name='joe' and
//           /example-mod:event1/status='online')
GET /streams/NETCONF?\
  filter=(%2Fexample-mod%3Aevent1%2Fname%3D'joe'%20and\
          %20%2Fexample-mod%3Aevent1%2Fstatus%3D'online')

// Для получения уведомлений из двух модулей (например, m1 + m2)
// filter=(/m1:* or /m2:*)
GET /streams/NETCONF?filter=(%2Fm1%3A*%20or%20%2Fm2%3A*)
```

B.3.7. Параметр start-time

Приведённый ниже идентификатор URI содержит пример параметра запроса start-time.

```
// start-time = 2014-10-25T10:02:00Z
GET /streams/NETCONF?start-time=2014-10-25T10%3A02%3A00Z
```

B.3.8. Параметр stop-time

Приведённый ниже идентификатор URI содержит пример параметра запроса stop-time.

```
// start-time = 2014-10-25T10:02:00Z
// stop-time = 2014-10-25T12:31:00Z
GET /mystreams/NETCONF?start-time=2014-10-25T10%3A02%3A00Z\
   &stop-time=2014-10-25T12%3A31%3A00Z
```

B.3.9. Параметр with-defaults

Предположим, что сервер реализует модуль example, определённый в Приложении A.1 к [RFC6243], а хранилище сервера соответствует определённому в Приложении A.2 к [RFC6243].

Если параметр basic-mode в URI протокольной возможности defaults (параграф 9.1.2) имеет значение trim, запрос для интерфейса eth1 может иметь приведённый ниже вид.

Без параметра запроса

```
GET /restconf/data/example:interfaces/interface=eth1 HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "example:interface" : [
    {
      "name" : "eth1",
      "status" : "up"
    }
  ]
```

```
    }
```

Отметим, что лист mtu не указан, поскольку он имеет принятое по умолчанию значение 1500, а для сервера установлено basic-mode=trim.

С параметром запроса

```
GET /restconf/data/example:interfaces/interface=eth1\
    ?with-defaults=report-all HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Сервер может возвратить отклик

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{
  "example:interface" : [
    {
      "name" : "eth1",
      "mtu" : 1500,
      "status" : "up"
    }
  ]
}
```

Отметим, что серверу следует возвращать лист mtu, поскольку для параметра with-defaults запрошен режим report-all.

Благодарности

Авторы благодарны за вклад в документ Ladislav Lhotka, Juergen Schoenwaelder, Rex Fernando, Robert Wilton и Jonathan Hansford. Спасибо также Mehmet Ersue, Mahesh Jethanandani, Qin Wu, Joe Clarke, Bert Wijnen, Ladislav Lhotka, Rodney Cummings, Frank Xialiang, Tom Petch, Robert Sparks, Balint Uveges, Randy Presuhn, Sue Hares, Mark Nottingham, Benoit Claise, Dale Worley и Lionel Morand за отличное техническое рецензирование документа.

Участие Andy Bierman в этой работе поддерживалось S&TCD[1] в рамках контракта W15P7T-13-C-A616. Любые мнения, заключения и выводы в этом документе принадлежат автору (авторам) и могут не отражать точку зрения S&TCD.

Адреса авторов

Andy Bierman

YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund

Tail-f Systems

Email: mbj@tail-f.com

Kent Watsen

Juniper Networks

Email: kwatsen@juniper.net

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

[1]United States Army, Space & Terrestrial Communications Directorate.