Internet Engineering Task Force (IETF)
Request for Comments: 8407
BCP: 216
Obsoletes: 6087
Category: Best Current Practice
ISSN: 2070-1721

A. Bierman
YumaWorks

October 2018

Guidelines for Authors and Reviewers of Documents Containing YANG Data Models

Рекомендации для авторов и рецензентов документов с моделями данных YANG

Аннотация

В этом документе приведены рекомендации для авторов и рецензентов спецификаций с модулями YANG. Заданы рекомендации и процедуры, предназначенные для улучшения взаимодействия и применимости реализаций протоколов NETCONF (Network Configuration Protocol) и RESTCONF, применяющих модули YANG. Документ отменяет RFC 6087.

Статус документа

Документ относится к категории Internet Best Current Practice.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. 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 https://www.rfc-editor.org/info/rfc8407.

Оглавление

[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Стандартизация интерфейсов настройки конфигурации сети для использования с протоколами управления, такими как NETCONF (Network Configuration Protocol) [RFC6241] и RESTCONF [RFC8040], требует модульного набора моделей данных, которые можно будет использовать и расширять в течение долгого времени.

Этот документ задаёт набор рекомендаций для документов, содержащих модели данных YANG 1.1 [RFC7950] и YANG 1.0 [RFC6020]. Язык YANG применяется для определения структур данных, протокольных операций и содержимого уведомлений, используемых на серверах NETCONF и RESTCONF. Сервер, поддерживающий определённый модуль YANG, будет поддерживать операции клиентов NETCONF и RESTCONF, как указано в соответствующем модуле YANG.

Многие конструкции YANG не обязательны для применения (например, операторы description), однако для того, чтобы сделать модули YANG более полезными, желательно задать набор рекомендаций, которые обеспечат более высокий уровень соответствия, нежели заданный спецификацией YANG [RFC7950] минимум.

Кроме того, YANG разрешает такие конструкции, как идентификаторы и строки бесконечного размера, а также обязательные узлы верхнего уровня, которые сервер не обязан поддерживать. В модулях IETF YANG могут применяться лишь конструкции, которые должны поддерживать все серверы.

Этот документ задаёт рекомендации по использованию, относящиеся к уровням операций и содержимого NETCONF, как указано в [RFC6241], а также по использованию методов и ресурсов RESTCONF, как указано в [RFC8040]. Рекомендации предназначены для авторов и рецензентов и направлены на повышение уровня надёжности и взаимодействия публикуемых моделей данных YANG.

Документ не является учебником по YANG и предполагает знакомство читателя с языком моделирования YANG.

1.1. Отличия от RFC 6087

Ниже перечислены изменения, внесённые в рекомендации [RFC6087].

- Обновлена ссылка на NETCONF заменой RFC 4741 на RFC 6241.
- Обновлена ссылка на NETCONF по протоколу SSH заменой RFC 4742 на RFC 6242.
- Обновлена ссылка на YANG Types заменой RFC 6021 на RFC 6991.
- Обновлены устаревшие ссылки URL для ресурсов IETF.
- Изменена рекомендация для узлов данных верхнего уровня.
- Разъяснено использование XPath (XML Path Language) для литеральных значений, представляющих отождествление YANG.
- Разъяснено использование XPath для when-stmt.
- Разъяснено использование XPath для осей preceding-sibling и following-sibling.
- Добавлены терминологические рекомендации.
- Добавлено упоминание RFC 8174, обновляющего RFC 2119 прояснением использования ключевых слов.
- Добавлены рекомендации для диаграмм деревьев YANG.
- Обновлены рекомендации XPath для преобразований и использования библиотеки функций.
- Обновлён раздел «Типы данных».
- Обновлён раздел «Определения уведомлений».
- Разъяснены узлы ключевых листьев.
- Разъяснено применение типов данных uint64 и int64.
- Добавлен текст по использованию свойств YANG.
- Добавлен раздел «Соглашения об именовании идентификаторов».
- Разъяснено использование обязательных узлов с условными дополнениями.
- Разъяснены соглашения о доменах и пространствах имён для примеров модулей.
- Разъяснены соглашения для идентификации компонентов кода.
- Добавлены рекомендации для YANG 1.1.
- Добавлен раздел «Ограничения узла данных YANG».
- Добавлено упоминание протокола RESTCONF.
- Добавленные рекомендации для хранилищ данных, пересмотренные в архитектуре хранилищ данных управления сетью (Network Management Datastore Architecture или NMDA).

2. Терминология

Ниже приведены определения используемых в документе терминов.

published - опубликованный
Стабильный выпуск модуля или субмодуля. Например, Request for Comments из параграфа 2.1 в [RFC2026] считается стабильной публикацией.

unpublished - неопубликованный
Стабильный выпуск модуля или субмодуля. Например, Internet-Draft из параграфа 2.2 в [RFC2026] считается нестабильной публикацией незавершённой работы, которая может измениться в любой момент.

YANG fragment - фрагмент YANG
Набор операторов YANG, не предназначенных для представления полного модуля или субмодуля YANG. Эти операторы не предназначены для реального применения за исключением представления примеров использования операторов YANG. Для указания дополнительных операторов YANG, присутствующих в реальном модуле, иногда применяется специальный синтаксис «...».

YANG tree diagram - диаграмма дерева YANG
Диаграмма (схема) представляющая содержимое модуля YANG в соответствии с [RFC8340]. Называется также диаграммой дерева (tree diagram).

2.1. Термины NETCONF

Ниже перечислены термины, определённые в [RFC6241]:

- capabilities - возможности;
- client - клиент;
- operation - операция;

- server - сервер.

2.2. Термины YANG

Ниже перечислены термины, определённые в [RFC7950]:

- data node - узел данных;
- module - модуль;
- namespace - пространство имён;
- submodule - субмодуль;
- version - версия;
- YANG;
- YIN.

Отметим, что термин module может использоваться в качестве базового для модулей и субмодулей YANG. При описании специфических свойств субмодуля применяется термин submodule.

2.3. Термины NMDA

Ниже перечислены термины, определённые в [RFC8342]:

- configuration - конфигурация;
- conventional configuration datastore - традиционное хранилище конфигурации;
- datastore - хранилище конфигурации;
- operational state - рабочее состояние;
- operational state datastore - хранилище рабочей конфигурации.

2.4. Уровни требований

Ключевые слова **должно** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

3. Общие рекомендации для документов

Рассматриваемые модули YANG вероятно будут содержаться в документах Internet-Draft (I-D). Все рекомендации для авторов I-D [ID-Guidelines] **должны** выполняться. Следует выполнять рекомендации для RFC, заданные в [RFC7322] (и будущих RFC, заменяющий его), [RFC-STYLE], [RFC7841].

В I-D, содержащих модули **должны** присутствовать разделы:

- Narrative (описание);
- Definition (определения);
- Security Considerations (вопросы безопасности);
- IANA Considerations (взаимодействие с IANA);
- References (списки ссылок).

Имеется три варианта присутствия YANG в I-D или RFC:

- нормативный модули или субмодуль;
- пример модуля или субмодуля;
- пример фрагмента YANG, не являющийся частью модуля или субмодуля.

Рекомендации этого документа относятся прежде всего к нормативным моделям (субмодулям), но могут применяться к примерам и фрагментам YANG.

3.1. Авторские права на модуль

Оператор description в модуле должен включать упоминание последнего одобренного заявления IETF Trust Copyright, которое доступно по ссылке <https://trustee.ietf.org/license-info/>.

3.2. Компоненты кода

Каждый нормативный модуль или субмодуль YANG в документе I-D или RFC считается компонентом кода. Для идентификации таких компонентов **должны** применяться строки <CODE BEGINS> и <CODE ENDS>. За тегом <CODE BEGINS> **следует** помещать строку, указывающую имя файла, заданное в параграфе 5.2 [RFC7950]. В строку имени **следует** включать дату выпуска, которая **должна** совпадать с датой самого свежего выпуска модуля. Ниже приведён пример для модуля ietf-foo с датой выпуска 20.03.2016 г.

```
<CODE BEGINS> file "ietf-foo@2016-03-20.yang"

    module ietf-foo {
      namespace "urn:ietf:params:xml:ns:yang:ietf-foo";
```

```
        prefix "foo";
        organization "...";
        contact "...";
        description "...";
        revision 2016-03-20 {
          description "Последний выпуск";
          reference "RFC XXXX: Foo Protocol";
        }
        // ... операторы
      }

<CODE ENDS>
```

3.2.1. Модули примеров

Модули примеров не являются компонентами кода и соглашение <CODE BEGINS> **недопустимо** применять для них. Примеры следует именовать с использованием префикса example, за которым следует дефис и описательное имя, например, example-toaster. Рекомендации для пространств имён применительно к примерам даны в параграфе 4.9.

3.3. Раздел терминологии

Раздел терминологии **должен** включаться, если документ определяет какие-либо термины или импортирует термины из других документов.

3.4. Диаграммы деревьев

Диаграммы деревьев YANG дают краткое представление модуля YANG и их **следует** включать для облегчения понимания структуры модуля YANG. Рекомендации для деревьев приведены в разделе 3 [RFC8340].

При использовании диаграммы дерева YANG в документ **должна** включаться информационная ссылка на спецификацию диаграмм деревьев YANG. Пример такой ссылки имеется в параграфе 2.2 [RFC8349].

3.5. Описательный раздел

Описательная часть **должна** включать обзорный раздел, описывающий область действия и сферу применения модулей, описанных в спецификации, а также указывать (при наличии) связи этих модулей с другими стандартами (в частности, содержащими модули YANG). В описательную часть **следует** включать один или несколько абзацев с кратким описанием заданных спецификацией модулей.

Если заданные в спецификации модули импортируют определения из других модулей (за исключением определений из [RFC7950] и [RFC6991]) или всегда реализуются совместно с другими модулями, этот факт **должен** быть отмечен в обзорном разделе. Также **должны** отмечаться все особые интерпретации определений из других модулей. Пример описательного раздела приведён в параграфе 2.3 [RFC8349].

Если документ содержит модули YANG, соответствующие архитектуре NMDA [RFC8342], это следует отметить в спецификации. Например,

> Модель данных YANG в этом документе соответствует архитектуре хранилищ данных управления сетью (NMDA), определённой в RFC 8342.

Следует применять согласованный отступ от начала строки во всех примерах, включая фрагменты YANG и данные протокольных сообщений. Слишком длинные строки **следует** переносить с включением символа \, как показано ниже.

```
[Примечание: \ служит лишь для форматирования]

<myleaf xmlns="tag:example.com,2017:example-two">\
  это слишком длинное значение, поэтому строку нужно\
  перенести, чтобы она была не длиннее 72 символов\
</myleaf>
```

3.6. Раздел определений

Этот раздел содержит модули, определяемые спецификацией. Эти модули **следует** записывать с использованием синтаксиса YANG 1.1 [RFC7950]. **Можно** применять синтаксис YANG 1.0 [RFC6020], если в модуле не требуется семантика или конструкции YANG 1.1. Если какой-либо из импортируемых модулей YANG написан на YANG 1.1, данный модуль **должен** также быть написан на YANG 1.1.

В документе также **может** быть представлена версия модуля с YIN-синтаксисом и могут присутствовать другие типы модулей, такие как SMIv2 (Structure of Management Information Version 2), на которые эти рекомендации не влияют.

Если сам модуль является нормативным, а не примером модуля или фрагмента YANG, все операторы YANG в модуле YANG считаются нормативными. Использование ключевых слов, описанное в [RFC2119] и [RFC8174], применимо к операторам YANG description в нормативных модулях так же, как в других нормативных разделах.

В примеры модулей YANG и фрагменты YANG **недопустимо** включать нормативный текст (в том числе слова заглавными буквами, указанные в [RFC2119] и [RFC8174]).

Для всех операторов YANG в модуле **следует** применять согласованные отступы от начала строки.

Рекомендации по использованию YANG приведены в разделе 4.

3.7. Раздел «Вопросы безопасности»

Каждая спецификация, определяющая модули, **должна** включать раздел с обсуждением вопросов безопасности, связанных с модулями. Этот раздел **должен** соответствовать последнему одобренному шаблону (см. <https://trac.ietf.org/trac/ops/wiki/yang-security-guidelines>). В параграфе 3.7.1 приведён шаблон раздела, соответствующий шаблону от 08.05.2013 г., обновлённому 02.07.2018 г. Авторы **должны** проверять страницу по

указанному выше идентификатору URL на предмет выпуска новой версии шаблона. В частности следует выполнять указанные ниже требования.

- Узлы данных с возможностью записи, которые могут быть особенно разрушительными при злоупотреблении, **должны** явно указываться по именам, а связанные риски безопасности **должны** быть разъяснены.
- Доступные для чтения узлы, содержащие конфиденциальные сведения или вызывающие проблемы приватности, **должны** явно указываться по именам, а связанные риски безопасности или приватности **должны** быть разъяснены.
- Операции (т. е. операторы YANG rpc), которые могут оказывать вредное влияние на поведение системы, **должны** явно указываться по именам, а связанные риски безопасности или приватности **должны** быть разъяснены.

3.7.1. Шаблон раздела «Вопросы безопасности»

X. Вопросы безопасности

Заданный этим документом модуль YANG определяет схему для данных, которые организованы для доступа по протоколам управления сетью, таким как NETCONF [RFC6241] и RESTCONF [RFC8040]. Нижним уровнем NETCONF является защищённый транспортный уровень с обязательной реализацией протокола SSH (Secure Shell) [RFC6242]. Нижним уровнем RESTCONF является HTTPS с обязательной реализацией защищённого транспорта TLS [RFC8446].

Модель управления доступом NETCONF [RFC8341] позволяет разрешать доступ конкретным пользователям NETCONF или RESTCONF лишь к предопределённому набору содержимого и операций протокола NETCONF или RESTCONF.

> При наличии узлов с возможностью записи (принятое по умолчанию значение config true) следует описать их конкретные уязвимости.

В этом модуле YANG имеются узлы с возможностью записи, создания, удаления (т. е. с принятым по умолчанию значением config true). Эти узлы могут быть уязвимы в некоторых сетевых средах. Операции записи (например, edit-config) в такие узлы без подобающей защиты могут оказывать негативное влияние на работу сети. Имеются субдеревья и узлы данных с уязвимостями или конфиденциальными сведениями

> **<список субдеревьев и узлов данных и указание их уязвимостей>**
> Для всех модулей YANG должна выполняться оценка конфиденциальности и уязвимости узлов, доступных для чтения (узлы с config false и другие узлы, поскольку они могут считываться такими операциями как get-config). Например, они могут раскрывать сведения о заказчике или персональные данные, защищаемые Европейским союзом, при доступе к узлам неуполномоченных лиц.

Некоторые доступные для чтения узлы в данном модуле YANG могут быть уязвимыми или конфиденциальными в некоторых сетевых средах. Важно контролировать доступ для чтения (например, get, get-config, notification) таких узлов. Имеются субдеревья и узлы данных с уязвимостями или конфиденциальными сведениями

> **<список субдеревьев и узлов данных и указание их уязвимостей>**
> Если модуль YANG определяет операции RPC, для них нужно описать конкретные уязвимости.

Некоторые операции RPC в этом модуле YANG могут быть уязвимыми в некоторых сетевых средах. Важно контролировать доступ к таким операциям и их конфиденциальность и уязвимость

> **<список операций RPC и указание их уязвимостей>**

3.8. Раздел «Взаимодействие с IANA»

В соответствии с правилами IESG, установленными в <https://www.ietf.org/id-info/checklist.html>, каждый документ I-D, представляемый IESG для публикации, **должен** содержать раздел IANA Considerations. Требования к этому разделу зависят от действий, которые нужны от IANA. Если документ не запрашивает действий IANA, в раздел включается фраза «This document has no IANA actions». Более подробные рекомендации приведены в [RFC8126].

Каждый нормативный модуль YANG **должен** регистрироваться в реестрах IETF XML Registry [RFC3688] [IANA-XML] и YANG Module Names [RFC6020] [IANA-MOD-NAMES]. Это применимо как к новым, так и к обновляемым модулям. Пример обновления регистрации для модуля ietf-template представлен в разделе 5.

3.8.1. Документы, создающие новые пространства имён

Если I-D определяет новое пространство имён, администрируемое IANA, документ **должен** включать раздел IANA Considerations, описывающий управление пространством имён. В частности, если значение оператора namespace в модуле YANG ещё не зарегистрировано в IANA, **должна** запрашиваться регистрация новой записи в субреестре ns реестра IETF XML Registry.

3.8.2. Документы с расширением пространств имён

Имеющееся пространство имён может расширяться субмодулем YANG, относящимся к модулю, уже зарегистрированному IANA. В этом случае документ, содержащий основной модуль, **должен** быть обновлён для использования последнего выпуска субмодуля.

3.9. Разделы ссылок

Для каждого присутствующего в модуле оператора import или include, который связан с модулем, заданным в другом документе, **должна** указываться нормативная ссылка на этот документ в разделе Normative References. Ссылка **должна** соответствовать конкретному выпуску модуля, используемому в спецификации.

Для каждого присутствующего в модуле нормативного оператора reference, указывающего отдельный документ, **следует** включать ссылку на этот документ в раздел Normative References. Ссылке **следует** указывать версию документа, используемую в спецификации. Если оператор reference содержит информационную ссылку, соответствующий документ **может** указываться в разделе Informative References.

3.10. Инструменты для проверки

Все модули необходимо проверять перед представлением I-D. Компилятор YANG pyang доступен на GitHub по ссылке <https://github.com/mbj4668/pyang>. При использовании pyang для проверки нормативного модуля в команде **должна** указываться опция --ietf для обнаружения любых несоответствий требованиям IETF. Если pyang применяется для проверки примера модуля, **можно** указывать в команде опцию --ietf для обнаружения несоответствий требованиям IETF.

Программа yanglint, также доступная на GitHub по ссылке <https://github.com/CESNET/libyang>, может служить для проверки операторов XPath в модулях YANG.

3.11. Средства извлечения модулей

Доступна версия программы rfcstrip, извлекающая модули YANG из документов I-D и RFC. Программа доступна по ссылке <https://github.com/mbj4668/rfcstrip>. Этот инструмент можно применять для проверки корректности тегов <CODE BEGINS> и <CODE ENDS> и возможности извлечения нормативных модулей YANG.

Программа xym, доступная на GitHub по ссылке <https://github.com/xym-tool/xym>, также позволяет извлекать модули YANG из документов.

3.12. Примеры применения модулей

Каждой спецификации, определяющей модули, **следует** включать примеры использования в самом документе или в приложении. Сюда входят фрагменты кода в подходящем формате (например, XML или JSON) для демонстрации предполагаемого использования модулей YANG. Примеры модулей **должны** быть проверены, средства проверки указаны в параграфе 3.10. При использовании в примерах адресов IP **следует** указывать сочетание адресов IPv4 и IPv6 или исключительно адреса IPv6.

4. Рекомендации по использованию YANG

Модули в спецификациях IETF Standards Track должны соответствовать всем синтаксическим и семантическим требованиям YANG 1.1 [RFC7950] (см. исключение для YANG 1.0 в параграфе 3.6). Рекомендации этого параграфа служат дополнением к спецификации YANG [RFC7950], предназначенным для задания минимального набора требований.

Для повышения уровня взаимодействия и использования обретённого опыта в следующих параграфах приведены рекомендации для конкретных конструкций YANG. Включены лишь рекомендации, которые разъясняют минимальные требования по соответствию.

4.1. Соглашения по именованию модулей

Нормативные модули в документах Standards Track **должны** именоваться в соответствии с рекомендациями раздела «Взаимодействие с IANA» в [RFC7950]. В именах модулей **следует** применять отличительное слово или сокращение (например, имя протокола или рабочей группы). Если новые определения задаются для расширения имеющихся модулей, следует сохранять имена, а не создавать новые.

Имена всех публикуемых модулей **должны** быть уникальными. Для модулей YANG, публикуемых в RFC, такую уникальность гарантирует IANA. Для неопубликованных модулей авторам нужно убедиться, что другой разрабатываемый модуль не имеет такое же имя.

Примеры модулей не являются нормативными и для них **следует** использовать префикс example-.

Предлагается выбирать стабильный префикс, представляющий организацию в целом. Все нормативные модули YANG, публикуемые IETF, **должны** начинаться с префикса ietf-. Другие органы стандартизации, такие как IEEE, могут применять свой префикс (например, ieee-) для всех модулей YANG.

После публикации имени модуля **недопустимо** применять его повторно даже при переводе содержащего модуль RFC в статус Historic. Имя модуля не может быть изменено в YANG и модуль будет считаться новым, а не переименованным.

4.2. Префиксы

Областью действия всех определений YANG является модуль, содержащий соответствующее определение. Это позволяет применять определения из разных модулей, даже если их имена не уникальны. В приведённом ниже примере идентификатор foo применяется во всех трёх модулях.

```
module example-foo {
  namespace "tag:example.com,2017:example-foo";
  prefix f;

  container foo;
}

module example-bar {
  namespace "tag:example.com,2017:example-bar";
  prefix b;

  typedef foo { type uint32; }
}

module example-one {
  namespace "tag:example.com,2017:example-one";
  prefix one;
  import example-foo { prefix f; }
  import example-bar { prefix b; }
```

```
        augment "/f:foo" {
           leaf foo { type b:foo; }
        }
      }
```

YANG задаёт для использования префиксов ряд правил:

- префиксы никогда не применяются для встроенных типов данных и ключевых слов YANG;
- префикс **должен** применяться для любого внешнего оператора (т. е. оператора, определённого с помощью YANG extension);
- **должен** применяться подобающий префикс для всех идентификаторов, импортируемых из других модулей;
- **должен** применяться подобающий префикс для всех идентификаторов, включённых из субмодуля.

Ниже приведены рекомендации по использованию префиксов текущего (локального) модуля:

- префикс локального модуля **следует** указывать во всех выражениях путей;
- префикс локального модуля **должен** указываться во всех операторах default для типов данных identityref и instance-identifier;
- префикс локального модуля **можно** использовать для ссылок на typedef, grouping, extension, feature, identity, определённых в модуле.

Для префиксов **следует** выбирать короткие, но, вероятно, уникальные значения. **Не следует** выбирать значения префиксов, уже используемые в опубликованных модулях.

4.3. Идентификаторы

Идентификаторы в публикуемых модулях YANG **должны** иметь размер от 1 до 64 символов. К ним относятся любые конструкции, заданные как маркеры identifier-arg-str в ABNF раздела 14 [RFC7950].

4.3.1. Соглашения об именовании идентификаторов

В идентификаторах модуля **следует** применять единый шаблон, применяя только строчные буквы, цифры и тире. **Можно** применять Заглавные буквы, точку и символ подчёркивания, если идентификатор представляет общеизвестное значение, использующее эти символы. YANG не разрешает применять в идентификаторах другие символы.

В идентификаторы **следует** включать полные слова и/или общеизвестные сокращения и акронимы. Дочерним узлам контейнера или списка **не следует** реплицировать родительский идентификатор. Идентификаторы YANG являются иерархическими и уникальность требуется лишь в рамках набора «братских» узлов, определённых в одном пространстве имён модуля.

Допускается применение общих идентификаторов, таких как name или id, в операторах определения данных, особенно если узлы имеют один тип данных.

Идентификаторам **не следует** нести какую-либо особую семантику, указывающую свойства моделирования данных, для передачи которых в машиночитаемом виде предназначены лишь операторы YANG (включая операторы расширения YANG). Например, именование объекта config или state не означает, что это данные конфигурации или состояния. Для назначения семантики в машиночитаемом формате YANG могут служить лишь операторы YANG и операторы расширения YANG.

4.4. Подразумеваемые значения

В общем случае предполагается, что **не следует** включать субоператоры, содержащие очень общие значения, принятые по умолчанию. В таблице приведены субоператоры, которые обычно применяются с принятым по умолчанию значением, указание которого будет усложнять чтение модулей.

Оператор	*Принятое по умолчанию значение*
config	true
mandatory	false
max-elements	unbounded
min-elements	0
ordered-by	system
status	current
yin-element	false

4.5. Условные операторы

Модуль можно концептуально разделить на части с помощью операторов if-feature и/или when. Разработчикам моделей данных следует внимательно относиться к модульности, включая применение условных операторов YANG. Если определение является необязательным в зависимости от поддержки сервером возможностей протокола NETCONF или RESTCONF, **следует** применять оператор YANG feature. Определённый оператор feature следует применять в операторе условия if-feature, указывающем условное определение данных.

Если какие-либо данные уведомления или не являющегося конфигурационным узла не обязательны, от сервера может не требоваться возврат экземпляра этого узла данных. При наличии условия для возврата узла данных для уведомления или запроса извлечения это условие **должно** отражаться в документации. Например, для узла данных может применяться оператор when или if-feature или условие может быть разъяснено в операторе description внутри узла данных или одного из его потомков.

Если операторы if-feature применяются к узлу списка, те же операторы if-feature **должны** применяться ко всем узлам ключевых листьев списка. **Недопустимо** наличие операторов if-feature, применяемых к листьям ключей, но не применяемых к родительскому списку.

Не следует применять оператор when к узлам ключевых листьев. Возможно, что оператор when для узла-предка ключевого листа будет иметь такой же результат установки узла (node-set), что и ключевой лист. В таких случаях **следует** избегать оператора when для ключевых листьев, поскольку он будет избыточным.

4.6. Использование XPath

В этом параграфе даны рекомендации по использованию языка XML Path (XPath) [W3C.REC-xpath] в модулях YANG.

4.6.1. Контекст оценки XPath

YANG определяет 5 различных вариантов контекста оценки операторов XPath.

1. Рабочее хранилище (running) - множество всех конфигурационных узлов данных. Корень документа является концептуальным контейнером (например, config в операции edit-config), который является родителем всех операторов определения данных верхнего уровня с config true.
2. Данные состояния + рабочее хранилище (running) - множество всех узлов данных YANG. Корень документа является концептуальным контейнером для всех операторов определения данных верхнего уровня.
3. Уведомления - документ уведомлений о событиях. Корень документа является элементом уведомления.
4. Ввод RPC. Корень документа является концептуальным узлом input, который служит родителем всех определений входных параметров RPC.
5. Вывод RPC. Корень документа является концептуальным узлом output, который служит родителем всех определений выходных параметров RPC.

Отметим, что эти контексты XPath не могут смешиваться. Например, оператор when в контексте уведомлений не может ссылаться на данные конфигурации.

```
notification foo {
  leaf mtu {
    // НЕ ВЕРНО, поскольку оператор when находится в контексте уведомлений
    when "/if:interfaces/if:interface[name='eth0']";
    type leafref {
      // Верно, поскольку оператор path имеет иной контекст
      path "/if:interfaces/if:interface/if:mtu";
    }
  }
}
```

Особенно важно учитывать контекст оценки XPath для выражений XPath, определённых в группировках. Такие выражения могут быть непереносимыми, что означает невозможность их использования в нескольких контекстах с корректным результатом. Если определённое в группировке выражение XPath предназначено для определённого контекста, этот контекст **следует** указывать в операторе description для группировки.

4.6.2. Библиотека функций

Функции position и last **не следует** применять. Это относится и к неявному использованию функции position (например, //chapter[42]). Серверу требуется поддерживать лишь относительный порядок документов XML для всех экземпляров определённого списка, упорядоченного пользователем, или leaf-list. Функции position и last можно использовать, если они вычисляются в контексте, где узлом является упорядоченный пользователем list или leaf-list.

Не следует применять функцию id. Атрибут ID не присутствует в документах YANG, поэтому функция не имеет смысла. Компилятору YANG **следует** возвращать для этой функции пустую строку.

Функции namespace-uri и name применять **не следует**. Расширенные имена в XPath отличаются от YANG, где конкретного канонического представления не существует.

Не следует применять функцию lang, которая не подходит для YANG, поскольку документы не имеют атрибута lang. Компилятору YANG **следует** возвращать для этой функции значение false.

Функции local-name, namespace-uri, name, string, number **не следует** использовать с аргументом node-set, поскольку в этом случае результат функции будет зависеть от порядка набора узлов, который может отличаться на каждом сервере. Для вызовов функции с неявным преобразованием node-set в string проблема сохраняется.

Не следует применять функцию local-name для ссылки на локальные имена вне модуля YANG, определяющего выражения must или when с функцией local-name. Например, не следует использовать

```
/*[local-name()='foo']
```

Следует использовать функцию derived-from-or-self вместо равенств для значений identityref. Это позволяет концептуально дополнять отождествления. Например,

```
// не следует использовать
when "md-name-format = 'name-format-null'";

// предпочтительно
when "derived-from-or-self(md-name-format, 'name-format-null')";
```

4.6.3. Отношения порядка

«Оси» attribute и namespace не поддерживаются в YANG и **могут** быть пустыми в серверах NETCONF и RESTCONF.

Отношения preceding и following применять **не следует**. Эти конструкции полагаются на порядок документов XML в конфигурационной базе данных сервера NETCONF или RESTCONF, который может быть не согласованным или не давать одинаковых результатов в разных реализациях. Вместо этого **следует** применять выражения-предикаты на основе статических свойств узла (например, имя или значение элемента и отношения ancestor или descendant). Отношения preceding и following **можно** использовать, если порядок документов не влияет на результат выражения (например, для проверки глобальной уникальности значения параметра).

Отношения preceding-sibling и following-sibling **не следует** применять, однако они допустимы, если порядок документов не влияет на результат выражения. От сервера требуется поддержка лишь относительного порядка документов XML для всех экземпляров определённо упорядоченного пользователем list или leaf-list. Отношения preceding-sibling и following-sibling **можно** применять, если они оцениваются в контексте, где узлом является упорядоченный пользователем list или leaf-list.

4.6.4. Типы

Узлы данных, использующие встроенные типы int64 и uint64, **не следует** применять в числовых и логических выражениях. Имеются граничные условия, при которых преобразование 64-битового типа YANG в число XPath могут давать некорректные результаты. В частности, число XPath с плавающей запятой «двойной точности» не может представить очень большое положительное или отрицательное 64-битовое число, поскольку оно обеспечивает суммарную «точность» лишь в 53 бита. Данные типа int64 и uint64 **можно** применять в числовых выражениях, если для представления значений достаточно 53 битов.

Создателям моделей данных следует быть осторожными и не путать пространства значений YANG и XPath. Типы данных в них не одинаковы и преобразования между YANG и XPath **следует** выполнять с осторожностью. **Можно** выполнять явные преобразования типов XPath (например, функции string, boolean, number) вместо неявных.

В выражения XPath с литеральными значениями, представляющими отождествления YANG, **следует** всегда включать объявленный префикс модуля, где определено отождествление.

Выражениям XPath для операторов when **не следует** ссылаться на узлы контекста или их потомков. **Можно** ссылаться на узлы-потомки, если оператор when содержится внутри оператора augment, а указываемый узел не определён внутри оператора augment. Например,

```
augment "/rt:active-route/rt:input/rt:destination-address" {
   when "rt:address-family='v4ur:ipv4-unicast'" {
     description
       "Это дополнение действительно лишь для IPv4 unicast.";
   }
   // Узлы, заданные здесь в операторе augment,
   // не могут быть указаны в операторе when
}
```

4.6.5. Шаблоны

Можно создать выражения XPath, которые будут давать разные результаты с разными модулями в одной реализации сервера. Это связано с узлами дополнения из других модулей.

Преобразование шаблонов на сервере выполняется для всех узлов из всех пространств имён, поэтому выражение must или when с оператором * всегда даёт результат false при оценка в рамках одного модуля YANG. В таких случаях **следует** указывать в операторах description, что дополняющие объекты предполагаются соответствующими преобразованию шаблона.

```
when /foo/services/*/active {
  description
    "В этом модуле нет прямых определений служб.
     Соответствовать будут объекты, добавленные контейнером служб.";
}
```

4.6.6. Логические выражения

Операторы YANG must и when используют логические выражения XPath при проверке условий. Важно задавать эти выражения так, чтобы не возникло непреднамеренной смены результатов, если включённые в выражение объекты будут изменены в новой версии модуля. Например, лист foo2 должен существовать при foo1 равном one или three.

```
leaf foo1 {
  type enumeration {
     enum one;
     enum two;
     enum three;
  }
}

leaf foo2 {
  // Некорректно
  must "/f:foo1 != 'two'";
  type string;
}

leaf foo2 {
  // Корректно
  must "/f:foo1 = 'one' or /f:foo1 = 'three'";
  type string;
}
```

В следующем выпуске модуля лист foo1 включает дополнительное значение four

```
leaf foo1 {
  type enumeration {
     enum one;
     enum two;
     enum three;
     enum four;
  }
}
```

В этом случае первое выражение XPath позволяет воспринять four в дополнение к one и three.

4.7. Управление жизненным циклом определений YANG

Оператор YANG status **должен** присутствовать в определении, если он имеет значение deprecated или obsolete. Значение status = current **не следует** менять напрямую на obsolete. Доступность объекта **следует** сохранять по меньшей мере в течение года после спены статуса на deprecated, прежде чем сменить статус на obsolete.

После публикации модуля или субмодуля **недопустимо** менять его имя.

Идентификатор URI для пространства имён модуля **недопустимо** менять после публикации модуля.

Субоператору даты выпуска внутри операторов import и include **следует** присутствовать, если применяется какая-либо группировка из внешнего модуля.

Если оператор import относится к модулю из стабильного источника (например, RFC для модуля IETF), в него **следует** включать оператор reference.

```
import ietf-yang-types {
   prefix yang;
   reference "RFC 6991: Common YANG Data Types";
}
```

При использовании субмодулей документ с основным модулем **должен** обновляться так, чтобы дата выпуска основного модуля совпадала с датой наиболее свежего из субмодулей, включённых в него (напрямую или косвенно).

Определения для использования в будущем не следует включать в модуль. **Не следует** включать «заменители» вроде reserved, как показано ниже.

```
leaf reserved {
  type string;
  description
    "Этот объект в настоящее время не нужен, но в будущем
     выпуске модуля для него может быть дано определение.";
  }
}
```

4.8. Заголовок, метаданные, операторы revision

Для опубликованных модулей пространство имён **должно** быть глобально уникальным URI, как указано в [RFC3986]. Значение обычно выделяет IANA.

Должен присутствовать оператор organization. Если модуль содержится в документе, предназначенном получить статус IETF Standards Track, в качестве организации **следует** указывать рабочую группу IETF (WG), уполномоченную на создание документа. Для других органов стандартизации предлагается применять аналогичный подход.

Должен присутствовать оператор contact. Если модуль содержится в документе, предназначенном получить статус IETF Standards Track, **следует** указывать WG-страницу и почтовый адрес рабочей группы, а также **следует** включать контактные данные основного автора или редактора документа. При наличии дополнительных авторов или редакторов, **можно** включать их контактные данные. Включать сведения о руководителе WG не требуется.

Должен присутствовать оператор description. Для модулей, публикуемых в документах IETF, **должен** включаться подходящий текст IETF Trust Copyright, как указано в параграфе 3.1.

Если модуль полагается на сведения из других документов, которые не совпадают с предполагаемыми операторами import в модуле, эти документы **должны** быть указаны в операторе reference.

Оператор revision **должен** присутствовать для каждой опубликованной версии модуля и **должен** включать субоператор reference, который **должен** указывать опубликованный документ с модулем. Модули часто извлекаются из исходных документов, а разработчикам и операторам полезно знать, где можно найти оригинальный документ. Оператор revision **может** включать субоператор description. Ниже дан пример оператора revision для опубликованного модуля YANG.

```
revision "2012-02-22" {
  description
    "Исходный выпуск";
  reference
    "RFC 8341: Network Configuration
               Access Control Model";
}
```

Для неопубликованных модулей не требуется указывать полную историю всех неопубликованных выпусков, т. е. в последовательности черновых версий в модуле требуется указать лишь наиболее свежий выпуск. Не следует удалять или повторно использовать оператор revision для публикуемого модуля. Новую дату выпуска не требуется указывать, пока содержимое модуля не будет изменено. Если содержимое модуля поменялось, дата выпуска новой версии модуля **должна** быть обновлена. Ниже приведён пример двух операторов revision для неопубликованного обновления опубликованного модуля YANG.

```
revision "2017-12-11" {
  description
    "Добавлена поддержка действий и уведомлений YANG 1.1, привязанных
     к узлам данных. Разъяснено применение расширений NACM в других
     моделях данных.";
  reference
    "RFC 8407: Network Configuration Protocol (NETCONF)
               Access Control Model";
}

revision "2012-02-22" {
  description
    "Исходный выпуск";
  reference
    "RFC 8341: Network Configuration
```

```
                    Access Control Model";
    }
```

4.9. Назначение пространств имён

Рекомендуется включать в документы лишь действительные модули YANG независимо от того, опубликованы ли эти модули. Это позволит:

- корректно компилировать модули вместо получения множества сообщений о критических ошибках;
- использовать модули в ранних реализациях без выбора случайных значений для пространства имён XML;
- заранее тестировать взаимодействие, поскольку независимые реализации будут использовать одно пространство имён XML.

Пока идентификатор URI не выделен агентством the IANA, идентификатор URI для пространства имён **должен** предоставляться для оператора namespace в модуле YANG. **Следует** выбирать значение, которое не будет приводить к конфликтам с другими пространствами имён YANG. **Недопустимо** применять имена, префиксы и URI стандартных модулей, указанных в реестре YANG Module Names.

Стандартному значению оператора namespace **следует** иметь форму

```
<URN prefix string>:<module-name>
```

Для публикуемых и непубликуемых модулей YANG **следует** использовать строку префикса URN в форме

```
urn:ietf:params:xml:ns:yang:
```

Ниже приведены корректные значения URN для оператора namespace в модулях Standards Track.

```
urn:ietf:params:xml:ns:yang:ietf-netconf-partial-lock
urn:ietf:params:xml:ns:yang:ietf-netconf-state
urn:ietf:params:xml:ns:yang:ietf-netconf
```

Отметим, что для модулей, не являющихся Standards Track **следует** применять иную строку префикса URN, которую **следует** выбирать в соответствии с рекомендациями [RFC7950].

Ниже приведены URI, иллюстрирующие возможное использование в модулях, не относящихся к Standards Track. Отметим, что в примерах IETF I-D **следует** использовать домен example.com. Эти URI не предназначены для разыменования и служат лишь для указания пространства имён модуля.

Примеры URI, использующие URL в соответствии с [RFC3986]

```
https://example.com/ns/example-interfaces
https://example.com/ns/example-system
```

Примеры URI, использующие теги в соответствии с [RFC4151]

```
tag:example.com,2017:example-interfaces
tag:example.com,2017:example-system
```

4.10. Определения данных верхнего уровня

Организацию данных верхнего уровня **следует** тщательно продумывать. Разработчикам моделей данных необходимо учитывать расширение функциональности протокола или семейства протоколов с течением времени.

Следует тщательно продумать разделение данных конфигурации и рабочего состояния. Иногда полезно задать раздельные контейнеры верхнего уровня для конфигурации и прочих данных. Для некоторых имеющихся узлов данных верхнего уровня конфигурационные данные могут быть вне зоны действия и контейнер будет представлять лишь данные рабочего состояния. Дополнительные подробности приведены в NMDA [RFC8342].

Следует минимизировать в модуле число узлов данных верхнего уровня. Часто полезно получать связанные между собой сведения из одного субдерева. Если данные слишком распределены, становится сложно получить их сразу.

В именах и организации данных **следует** отражать постоянные сведения, такие как имя протокола. Имя рабочей группы использовать **не следует**, поскольку оно может измениться.

Обязательное определение данных в базе задаётся как узел, который клиент должен предоставить, чтобы база данных стала действительной. Сервер не обязан указывать его значение.

Определениям данных верхнего уровня в базе **недопустимо** быть обязательными. Если на верхнем уровне присутствует обязательный узел, это сразу сделает базу данных недействительной. Такое может произойти при загрузке сервера или при динамической загрузке модуля во время работы.

4.11. Типы данных

Выбор подходящего типа данных (т. е. встроенного, имеющегося или нового производного типа) очень субъективен и можно задать лишь несколько требований к нему.

В моделях данных **следует** применять наиболее подходящие для конкретного случая встроенные типы.

Числовые типы данных со знаком (int8, int16, int32, int64) **не следует** применять, если семантика не предполагает отрицательных значений.

4.11.1. Расширяемость фиксированных значений

Если набор значений и содержимое типа данных контролируется одним органом (naming authority), **следует** применять перечисляемый (enumeration) тип данных.

```
leaf foo {
  type enumeration {
    enum one;
    enum two;
  }
```

```
}
```

Если нужна расширяемость перечисляемых значений, **следует** применять тип identityref вместо enumeration или иного встроенного типа данных.

```
identity foo-type {
  description "Base for the extensible type";
}

identity one {
  base f:foo-type;
}
identity two {
  base f:foo-type;
}

leaf foo {
  type identityref {
    base f:foo-type;
  }
}
```

Отметим, что любой модуль может объявить отождествление с базой foo-type, действительное для листа foo. Значения identityref считаются пригодными (qualified) именами.

4.11.2. Шаблоны и диапазоны

Если для строковых типов данных семантика позволяет определить машиночитаемый шаблон, **следует** включать хотя бы один такой шаблон. Для задания шаблона следует указывать строку в одинарных кавычках (´), поскольку строка в двойных кавычках может менять содержимое. При использовании шаблонов в определении типа с известными ограничениями, таким как ложные совпадения (несовпадения), такие ограничения **следует** указывать в определении данных или typedef. Приведённый ниже оператор typedef из [RFC6991] даёт пример использования оператора pattern.

```
typedef ipv4-address-no-zone {
  type inet:ipv4-address {
    pattern '[0-9\.]*';
  }
  ...
}
```

Если для строкового типа данных во всех реализациях требуется ограничить размер, **должен** включаться оператор length, как показано ниже в примере typedef из [RFC6991].

```
typedef yang-identifier {
  type string {
    length "1..max";
    pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*';
    pattern '.|..|[^xX].*|.[^mM].*|..[^lL].*';
  }
  ...
}
```

Если для численных типов семантика вносит дополнительные ограничения, **следует** включать оператор range, как показано ниже в примере из [RFC6991].

```
typedef dscp {
  type uint8 {
     range "0..63";
  }
  ...
}
```

4.11.3. Перечисляемые и биты

Для типов данных enumeration или bits семантику каждого enum или bit **следует** указывать в отдельном операторе description (внутри оператора enum или bit).

```
leaf foo {
  // Некорректно
  type enumeration {
    enum one;
    enum two;
  }
  description
    "Перечисляемый тип foo ...
     one: первый элемент
     two: второй элемент";
}

leaf foo {
  // Корректно
  type enumeration {
    enum one {
      description "Первый элемент";
    }
    enum two {
      description "Второй элемент";
    }
  }
  description
    "Перечисляемый тип foo ...  ";
```

```
    }
```

4.11.4. Тип union

Тип YANG union оценивается путём сравнения с каждым элементом объединения. Первое определение типа, принимающее значение как допустимое, является используемым типом элемента. В общем случае типы элементов **следует** упорядочивать от более ограниченных к менее ограниченным типам. В приведённом ниже примере тип enumeration никогда не даст совпадения, поскольку тип string будет соответствовать чему угодно.

Некорректное определение

```
type union {
  type string;
  type enumeration {
    enum up;
    enum down;
  }
}
```

Корректное определение

```
type union {
  type enumeration {
    enum up;
    enum down;
  }
  type string;
}
```

Возможна зависимость совпадения от используемой на входе кодировки. В XML все значения передаются как строковые узлы, но в JSON имеются разные типы для чисел, логических значений и строк. В приведённом ниже примере численное значение JSON всегда будет соответствовать типу int32, а в XML представляющая число строка будет соответствовать типу string. Во втором примере значение будет соответствовать типу int32 независимо от входной кодировки.

Некорректное определение

```
type union {
  type string;
  type int32;
}
```

Корректное определение

```
type union {
  type int32;
  type string;
}
```

4.11.5. Пустой и логический тип

YANG поддерживает пустой тип данных empty, который имеет одно значение (присутствие - present). По умолчанию установлено not present (отсутствие), что фактически не является значением. При использовании в ключе списка для этого ключевого листа может (и должно) существовать единственное значение. Тип empty **не следует** применять для листа ключей, поскольку это бессмысленно. Лист типа empty и leaf-list этого типа не различаются и оба ограничены одним экземпляром. Тип empty **не следует** применять для leaf-list.

Преимуществом применения типа empty вместо boolean является то, что принятое по умолчанию значение (not present) не занимает каких-либо байтов в представлении. Недостаток состоит в том, что клиент не всегда может определить, отсутствует лист leaf потому, что он был отфильтрован или потому, что не был реализован. У клиента может не быть полной и точной схемы данных, возвращаемых сервером, и он может не знать об отсутствующем листе.

Тип данных YANG boolean предоставляет 2 значения true и false. При использовании ключе списка для ключевого листа может существовать две записи. Значения по умолчанию игнорируются для ключевых листьев, но оператор default часто применяется для обычных (plain) логических листьев. Преимущество типа boolean состоит в том, что лист или лист-список имеют представление обоих значений. Принятое по умолчанию значение обычно не возвращается, если клиент не запросил его явно, поэтому типичное представление места не занимает.

В общем случае тип данных boolean **следует** применять вместо empty, как показано ниже.

Некорректное определение

```
leaf flag1 {
  type empty;
}
```

Корректное определение

```
leaf flag2 {
  type boolean;
  default false;
}
```

4.12. Определения типов с многократным использованием

При наличии подходящего производного типа в каком-либо стандартном модуле (например, [RFC6991]) **следует** применять этот тип вместо определения нового производного типа.

Если подходящий идентификатор units может быть связан с желаемой семантикой, **следует** включать оператор units.

Если с желаемой семантикой можно связать подходящее значение, принятое по умолчанию, **следует** включать оператор default.

Если определено много производных типов и предполагается их неоднократное применение в разных модулях, эти производные типы **следует** собрать в отдельный модуль или субмодуль для упрощения их использования.

Оператор description **должен** присутствовать.

Если семантика определения типа задана во внешнем документе (отличном от модуля YANG, указанного в операторе import), **должен** присутствовать оператор reference.

4.13. Группировки с многократным использованием

Группировкой многократного использования является группировка YANG, которая может быть импортирована в другой модуль и предназначена для использования другими модулями. Это отличается от группировки, применяемой внутри модуля, где она определена, и группировка многократного применения может экспортироваться в другие модули, поскольку она определяется на верхнем уровне модуля YANG. Ниже приведены рекомендации по повышению отказоустойчивости групп многократного применения.

- Чёткое указание назначения группировки в операторе description.
- Чёткое указание одного из 5 контекстов XPath в YANG (rpc/input, rpc/output, notification, узлы config true и прочие узлы данных), применимого или неприменимого для группировки.
- Отсутствие ссылок на данные вне группировки в операторах path, must, when.
- Отсутствие субоператоров default в leaf и choice, если значение применимо не во всех возможных контекстах.
- Отсутствие субоператоров config в узлах данных, если значение применимо не во всех возможных контекстах.
- Чёткое указание в операторе группировки description внешних зависимостей, таких как узлы, указанные абсолютным путём в операторах path, must, when.

4.14. Определения данных

Оператор description **должен** присутствовать в операторах YANG:

- anyxml;
- augment;
- choice;
- container;
- extension;
- feature;
- grouping;
- identity;
- leaf;
- leaf-list;
- list;
- notification;
- rpctypedef.

Если семантика определения данных задана во внешнем документе (отличном от модуля YANG, указанного в операторе import), **должен** присутствовать оператор reference.

Конструкция anyxml может быть полезна для представления HTML-баннера с элементами разметки, такими как и , и в таких случаях **может** применяться. Однако эту конструкцию **не следует** использовать, если можно применить другие типы узлов данных YANG для представления желаемого синтаксиса и семантики. Было обнаружено, что оператор anyxml не реализован единообразно на всех серверах. Возможно отсутствие поддержки смешанного режима XML или конфигурационных узлов anyxml.

При наличии ограничений целостности ссылок, связанных с желаемой семантикой, которая может быть представлена XPath, **следует** включать по меньшей мере один оператор must.

Для определений данных list и leaf-list при наличии ограничений возможного числа экземпляров во всех реализациях **следует** включать оператор max-elements.

При использовании операторов must или when в определении данных в оператор description этого определения **следует** включать описание каждого из них.

Оператор choice можно напрямую включать в оператор case в YANG 1.1, но это нужно тщательно продумать. Следует рассмотреть возможность простого включения вложенного choice в качестве дополнительных операторов case внутри родительского choice. Отметим, что операторы mandatory и default во вложенном операторе choice применяются лишь в том случае, когда сначала выбирается вариант case, содержащий вложенный оператор choice.

Если в списке заданы ключевые листья, их **следует** указывать как первые дочерние узлы в списке. В некоторых случаях ключевые листья **могут** размещаться в ином порядке (например, определяться в группировке, а не внутри list.

4.14.1. Контейнеры отсутствия

Контейнер отсутствия (non-presence) применяется для организации данных в конкретные субдеревья и не имеет в модели данных какой-либо дополнительной семантики, хотя YANG разрешает её (например, оператор must внутри контейнера non-presence).

Пример с «заворачиванием» в контейнеры

```
container top {
   container foos {
      list foo { ... }
   }
   container bars {
```

```
            list bar { ... }
        }
    }
```

Пример без заворачивания

```
    container top {
        list foo { ... }
        list bar { ... }
    }
```

Применение контейнеров отсутствия для организации данных является субъективным вопросом, подобно применению каталогов в файловой системе. Хотя эти контейнеры не имеют какой-либо семантики, они могут влиять на операции протокола для узлов-потомков в контейнере non-presence, поэтому к их использованию **следует** подходить осторожно.

Протоколы NETCONF и RESTCONF в настоящее время не поддерживают возможность удаления разом всех записей списка (или leaf-list). Этого недостатка иногда можно избежать путём использования родительского контейнера (удаление контейнера будет удалять все его дочерние записи).

4.14.2. Узлы данных верхнего уровня

При использовании объектов верхнего уровня требуется учитывать ряд важных аспектов:

- «братские» (sibling) отношения на верхнем уровне не упорядочиваются;
- «братские» отношения на верхнем уровне не статичны и зависят от загружаемых модулей;
- для фильтрации субдеревьев извлечение leaf-list верхнего уровня считается узлом с совпадающим содержимым для всех «братьев» верхнего уровня;
- списки верхнего уровня со множеством экземпляров могут влиять на производительность.

4.15. Определения операций

Если семантика операции задана во внешнем документе (отличном от модулей YANG в операторе import), **должен** присутствовать оператор reference.

Если операция влияет на поведение системе, это **следует** указать в операторе description. Если операция может оказать вредное влияние на поведение системы, это **должно** быть указано в разделе «Вопросы безопасности».

4.16. Определения уведомлений

Оператор description **должен** присутствовать.

Если семантика уведомления задана во внешнем документе (отличном от модулей YANG в операторе import), **должен** присутствовать оператор reference.

Если уведомление указывает экземпляр ресурса, этот экземпляр **следует** идентифицировать в данных уведомления. Обычно это делается включением узлов-листьев leafref со значениями ключей для экземпляров ресурса. Например,

```
  notification interface-up {
    description "Отправка при активизации интерфейса.";
    leaf name {
      type leafref {
        path "/if:interfaces/if:interface/if:name";
      }
    }
  }
```

Отметим отсутствие формальных операторов YANG для идентификации каких-либо ресурсов узла данных, связанных с уведомлением. В операторе description для уведомления **следует** указывать, идентифицирует ли уведомление какие-либо ресурсы узла данных, связанные с событием, и как это делается.

4.17. Определения функций

Оператор YANG feature служит для определения меток, задающий набор необязательных функций в модуле. В операторах YANG, связанных с такими функциями, применяется оператор if-feature. Оператор description внутри оператор feature **должен** задавать все взаимодействия с другими функциями.

К наборам функций YANG, определяемых в модуле, следует относиться с осторожностью. Чрезмерное измельчение функций усложняет взаимодействие и его следует избегать. Очевидным примером некорректного применения функций является создание меток (тегов) для отдельных листьев (например, счётчиков) с различными функциями.

Если с функцией YANG связан большой набор объектов, следует рассмотреть возможность переноса этих объектов в отдельные модули вместо создания функции YANG. Отметим, что набор функций в модуле легко обнаруживается читателем, а идентификация набора связанных модулей во всей библиотеке YANG сложнее. Имена модулей с общим префиксом могут помочь читателям идентифицировать набор связанных модулей, но это предполагает, что читатель нашёл и установил все соответствующие модули.

Ещё одним аспектом решения о создании нового модуля или добавления функции YANG является стабильность соответствующего модуля. Может быть желательным создание стабильного базового модуля, который меняется нечасто. Если новые функции помещаются в отдельный модуль это не будет требовать замены базового модуля. При реализации в форме функции YANG модуль придётся публиковать заново.

Если одна функция требует реализации другой функции, **следует** применять оператор if-feature в зависимом операторе feature. В показанном ниже примере feature2 требует наличия feature1.

```
  feature feature1 {
    description "Некая функция протокола";
  }
```

```
feature feature2 {
  if-feature "feature1";
  description "Другая функция протокола";
}
```

4.18. Ограничения узла данных YANG

4.18.1. Управление числом элементов

Операторы min-elements и max-elements можно использовать для управления числом экземпляров list или leaf-list, требуемых для конкретного узла данных. Операторы ограничений в YANG **следует** применять для указания условий, относящихся ко всем реализациям модели данных. Если с операциями связаны зависимые от платформы ограничения (например, max-elements поддерживается для определённого списка), **следует** использовать оператор определения модели данных (например, лист max-ports) для указания ограничений.

4.18.2. Операторы must и when

Операторы YANG must и when служат для проверки ссылок между объектами и их поведение существенно различается. Оператор when ведёт к удалению данных, когда условие не выполняется (false) и это не считается ошибкой. Оператор when **следует** применять вместе с оператором augment или uses для создания модели по условию. Условие **следует** задавать на основе статических свойств дополняемого объекта (например, листьев списка ключей).

Оператор must вызывает ошибку хранилища данных при невыполнении условия (false). Этот оператор **следует** применять для того, чтобы обеспечить соблюдение ограничений на значения параметров, которые включают более 1 узла данных (например, параметр end-time должен быть больше параметра start-time).

4.19. Оператор augment

Оператор YANG augment служит для задания набора определений данных, которые будут добавлены в качестве дочерних узлов в целевой узел данных. Пространство имён для этих узлов данных будет задавать дополняющий, а не дополняемый модуль.

Оператор augment **не следует** применять, если целевой узел данных находится в том же модуле или субмодуле, что и оцениваемый оператор augment. Вместо этого **следует** добавлять встроенные операторы определения данных.

4.19.1. Условные операторы augment

Оператор augment часто используется вместе с оператором when и/или if-feature для задания условий дополнения к той или иной части модели данных. Ниже приведён пример из [RFC7223], показывающий условное добавление контейнера ethernet в список interface для записей типа ethernetCsmacd.

```
augment "/if:interfaces/if:interface" {
    when "if:type = 'ianaift:ethernetCsmacd'";

    container ethernet {
        leaf duplex {
            ...
        }
    }
}
```

4.19.2. Условные операторы определения обязательных данных

В YANG применяются очень конкретные правила способов обновления данных конфигурации в новых выпусках модуля. Эти правила позволяют «старым клиентам» взаимодействовать с «новыми серверами».

Если добавляются узлы данных в имеющуюся запись, от старых клиентов **недопустимо** требовать предоставления обязательных параметров, которых не было в исходном определении модуля.

Можно добавить условные операторы augment, чтобы старые клиенты не знали о новых условиях и не задавали новых условий. Условный оператор augment может содержать обязательные объекты лишь для случая несоблюдения условия (false), если объекты явно не запрошены клиентом. Таким способом можно применять лишь условные операторы augment, использующие форму условия с оператором when. Включённые на сервере функции YANG в этом случае не могут управляться клиентом, поэтому небезопасно добавлять обязательные узлы дополнения на основе оператора if-feature.

Условию оператора XPath when **недопустимо** включать данные, находящиеся вне целевого узла, поскольку клиент не может контролировать такие внешние данные.

В приведённом ниже умозрительном примере допускается дополнение записи interface листом mandatory-leaf, поскольку это дополнение зависит some-new-iftype. Старый клиент не знает этого типа, поэтому не будет его выбирать и не добавит новый обязательны узел данных.

```
module example-module {

  yang-version 1.1;
  namespace "tag:example.com,2017:example-module";
  prefix mymod;

  import iana-if-type { prefix iana; }
  import ietf-interfaces { prefix if; }

  identity some-new-iftype {
     base iana:iana-interface-type;
  }
```

```
      augment "/if:interfaces/if:interface" {
         when "if:type = 'mymod:some-new-iftype'";

         leaf mandatory-leaf {
            type string;
            mandatory true;
         }
      }
   }
```

Отметим, что такой подход безопасен лишь при создании ресурсов данных. Небезопасно заменять или изменять ресурсы, если клиент не знает о новых условиях. Модель данных YANG **должна** представляться так, чтобы клиент знал об обязательных узлах данных, если ему известно условие для этих данных. В приведённом выше примере some-new-iftype задаётся одном модуле с оператором определения данных mandatory-leaf.

Такой подход небезопасен для отождествлений, заданных в базовом модуле, таком как iana-if-type, поскольку клиент не обязан знать о my-module лишь потому, что он знает о модуле iana-if-type.

4.20. Операторы отклонения

В соответствии с параграфом 7.20.3 RFC 7950, оператор YANG deviation не разрешается включать в модули IETF YANG, но он может быть полезен для документирования возможностей сервера. Операторы deviation не могут применяться многократно и обычно не переносятся с одной платформы на другие.

Есть несколько причин использования deviation в реализациях, например, для объектов, поддерживаемых не всеми платформами или предоставления функции с несколькими этапами разработки. Операторы deviation можно также применять для добавления в модуль аннотаций, не влияющих на требования совместимости.

Предполагается, что операторы deviation будут задаваться в отдельных от обычных определений YANG модулях. Это позволяет создавать временные и/или зависимые от платформы отклонения.

Порядок проверки операторов deviation может влиять на результат, поэтому **не следует** использовать несколько операторов deviation в одном модуле для одного объекта.

Оператор max-elements предназначен для описания архитектурных ограничений числа элементов в списках и не предназначен для задания ограничений платформ. Для платформ с ограниченными ресурсами лучше применять оператор deviation.

Ниже приведены примеры документирования ограничений ресурсов платформы.

Некорректно (max-elements в самом списке)

```
      container backups {
        list backup {
           ...
           max-elements  10;
           ...
        }
      }
```

Корректно (max-elements в операторе deviation)

```
      deviation /bk:backups/bk:backup {
        deviate add {
           max-elements  10;
        }
      }
```

4.21. Оператор расширения

Оператор YANG extension служит для задания внешних определений, которые в синтаксисе YANG имеют вид unknown-statement. К использованию операторов extension в публикуемых модулях требуется подходить с осторожностью. Ниже приведены рекомендации по использованию расширений YANG.

- **Недопустимо** наличие противоречий между семантикой extension и какими-либо операторами YANG. Расширения могут добавлять семантику, не охватываемую обычными операторами YANG.
- Модуль, содержащий оператор extension, **должен** чётко указывать требования по совместимости для расширения. Следует чётко указать, должны ли все реализации модуля YANG, содержащего расширение, реализовать также это расширение. Если этого не требуется, нужно указать условия, при которых реализация расширения требуется.
- Расширение **должно** чётко указывать, где его можно применять в других операторах YANG.
- Расширение **должно** чётко указывать, возможность или необходимость применения операторов YANG или других расширений как субоператоров этого расширения.

4.22. Корреляция данных

Данные могут коррелироваться разными способами с использованием базовых типов, имён и организации данных. Есть несколько способов расширения функциональности модуля на основе степени связи между старой и новой функциональностью.

- Встраивание (inline). Модуль обновляется доступными по протоколу новыми объектами с использованием именования и организации исходных объектов. Новые объекты используют прежнее пространство имён.
- Дополнение (augment). Создаётся новый модуль с доступными про протоколу объектами, дополняющими исходную структуру данных. с использованием именования и организации исходных объектов. Новые объекты используют пространство имён нового модуля.

- Отражение (mirror). Создаются новые объекты в новом или исходном модуле, но применяется новая схема именования и размещения данных. Именование объектов может связываться разными способами. Тесная привязка достигается с помощью типа данных leafref с установкой для require-instance значения true. **Следует** применять этот метод.

Если новые экземпляры данных не ограничиваются значениями, применяемыми в исходной структуре данных, для оператора require-instance **должно** устанавливаться значение false. Слабая привязка достигается использованием ключевых листьев того же типа, что и с исходной структуре данных. Семантика совпадает со случаем установки для require-instance значения false.

Дополнительные связи между конфигурацией и рабочим состоянием разъяснены в NMDA [RFC8342].

4.22.1. Использование leafref для сопоставления ключей

Иногда непрактично дополнять структуру данных. Например, сопоставляемые данные могут иметь другие ключи или содержать обязательные узлы. Ниже приведены примеры использования типа данных leafref для сопоставления.

Не рекомендуется

```
list foo {
   key name;
   leaf name {
     type string;
   }
   ...
}

list foo-addon {
   key name;
   config false;
   leaf name {
     type string;
   }
   ...
}
```

Предпочтительно

```
list foo {
   key name;
   leaf name {
     type string;
   }
   ...
}

list foo-addon {
   key name;
   config false;
   leaf name {
     type leafref {
       path "/foo/name";
       require-instance false;
     }
   }
   leaf addon {
     type string;
     mandatory true;
   }
}
```

4.23. Рабочее состояние

Моделирование рабочего состояния с помощью YANG со временем совершенствовалось. Сначала к рабочему состоянию относили лишь данных, для которых оператор config имел значение false. Эти данные не считались частью сведений в хранилище данных, что значительно усложняло определение YANG XPath.

Сейчас рабочее состояние моделируется на языке YANG в соответствии с архитектурой NMDA [RFC8342] и концептуально содержится в хранилище данных рабочего состояния, которое также включает рабочие значения данных конфигурации. Больше нет необходимости дублировать структуры данных для поддержки отдельных разделов конфигурации и данных рабочего состояния.

В этом параграфе рассматриваются некоторые вопросы моделирования данных, относящиеся к рабочему состоянию и приведены рекомендации по переходу моделей YANG в режим совместимости с архитектурой NMDA.

4.23.1. Объединение данных рабочего состояния и конфигурации

По возможности рабочее состояние **следует** объединять с соответствующими данными конфигурации. Это предотвратит дублирование ключевых листьев и узлов-предков, а также состязание при получении динамических данных и позволит извлекать рабочее состояние вместе с данными конфигурации с минимальными издержками на сообщения.

```
container foo {
  ...
  // Содержит узлы config true, а также узлы config false, у которых
  // нет соответствующих объектов config true (например, счётчиков)
}
```

4.23.2. Представление рабочих значений данных конфигурации

По возможности **следует** использовать общий тип данных для представления настраиваемых и рабочих значений для данного объекта leaf или leaf-list.

Иногда настраиваемый набор значений отличается от рабочего набора для объекта, например, листья admin-status и oper-status в [RFC8343]. В этом случае **могут** применяться разные объекты для представления настраиваемых и рабочих значений.

Иногда ключевые листья не идентичны для для данных конфигурации и соответствующего рабочего состояния. В этом случае **могут** применяться разные списки для настраиваемых и рабочих значений.

Если невозможно объединить конфигурационное и рабочее состояние, используемым для представления списков ключам **следует** иметь одинаковый тип. Тип leafref **следует** применять в рабочем состоянии для ключевых листьев, имеющих соответствующие экземпляры в конфигурации. Для оператора require-instance **можно** установить значение false (только в модулях YANG 1.1), чтобы указать возможность использования в рабочем состоянии экземпляров, которые не имеют соответствующих данных конфигурации.

Необходимость репликации объектов или определения других объектов рабочего состояния зависит от модели данных. Определить единый подход для всех моделей данных не представляется возможным.

Разработчикам **следует** подробно описывать и обосновывать все отклонения от архитектуры NMDA, такие как использование раздельных субдеревьев и/или листьев. Для этого **следует** применять операторы description в конфигурации и рабочем состоянии.

4.23.3. Рекомендации по переходу к NMDA

Модули данных YANG **следует** разрабатывать с учётом того, что они будут применяться на серверах, поддерживающих хранилище рабочего состояния. С учётом этого в модули YANG **следует** включать узлы config false, где это имеет смысл для модели данных. Такие узлы **не следует** определять для представления конфигурационных значений за исключением случаев, когда пространства рабочих и конфигурационных значений могут различаться. В последнем случае **следует** определять отдельный узел config false для рабочего значения узла конфигурации.

Ниже приведены рекомендации для разработчиков модулей YANG, которые позволят обеспечить максимальную полезность моделей как в имеющихся, так и в новых реализациях.

Новые модули и модули, не связанные с рабочим состоянием данных конфигурации, **следует** сразу же структурировать для совместимости с NMDA, как указано в параграфе 4.23.1. Переход **может** быть отложен, если модуль не содержит объектов хранилища конфигурации.

Ниже приведены варианты, которым **можно** следовать при определении механизмов NMDA.

(a) Модули, которым сразу нужна поддержка функций NMDA, **следует** структурировать для NMDA. **Может** существовать временная версия модуля не-NMDA, как имеющаяся модель или модель, созданная вручную или подходящим инструментом, отражающим текущую стратегию моделирования. Оба варианта (NMDA и не-NMDA) **следует** публиковать в одном документе, помещая модули NMDA в основной части документа, а не-NMDA - в ненормативном приложении. Наличие модуля не-NMDA даёт временный «мост» на период разработки реализаций NMDA.

(b) Для опубликованных модулей модель следует опубликовать заново со совместимой с NMDA структурой, отменяющей конструкции не-NMDA. Например, модель ietf-interfaces из [RFC7223] была реструктурирована для NMDA в [RFC8343]. Иерархия /interfaces-state была помечена как status deprecated. Модели, отмечающие иерархии /foo-state как status deprecated, позволяют поддерживающим NMDA реализациям избежать дублирования узлов состояний, сохраняя реализациям не-NMDA возможность использования этих узлов для доступа к значениям рабочего состояния.

(c) Для моделей, дополняющих не структурированные для NMDA модели, разработчикам следует рассмотреть структуру базовой модели с учётом приведённых выше рекомендаций. По возможности для таких моделей нужно подготовить новый выпуск базовой модели с поддержкой NMDA. Когда это невозможно, **следует** избегать дополнения контейнеров состояния, ожидая нового выпуска базовой модели с отменой устаревших контейнеров состояния. **Рекомендуется** дополнять лишь иерархию /foo в базовой модели. Если эти рекомендации не выполняются, **следует** включать все новые элементы состояния в свой модуль.

4.23.3.1. Временные модули не-NMDA

Временный модуль не-NMDA позволяет не поддерживающим NMDA клиентам получить доступ к рабочему состоянию на сервере NMDA. Модуль содержит узлы данных config false, определённые в традиционном (до NMDA) модуле YANG.

Сервер, которому нужно поддерживать клиентов NMDA и не-NMDA может анонсировать сразу новый модуль NMDA и временный модуль не-NMDA. Клиент без поддержки NMDA может использовать отдельные субдеревья foo и foo-state, исключая субдеревья foo-state, расположенные в других (временных) модулях. Модуль NMDA может использоваться клиентом без поддержки NMDA для доступа к традиционным хранилищам данных и устаревшей операции <get> для доступа к вложенным узлам данных config false.

Операции создания временной модели не-NMDA из модели NMDA перечислены ниже.

- Переименование модуля с добавлением суффикса -state.
- Смена пространства имён путём добавления суффикса -state к имени исходного пространства.
- Смена префикса путём добавления суффикса -s к исходному префиксу.
- Добавление импорта в исходный модуль (например, для определений typedef).
- Сохранение или создание узлов верхнего уровня с config false. Эти субдеревья представляют узлы данных config false, объединённые в субдерево конфигурации и поэтому недоступные клиентам, не понимающим NMDA. Для каждого нового узла устанавливается status deprecated.

- В описании модуля **следует** чётко указать, что это временный модуль не-NMDA.

4.23.3.2. Пример создания нового модуля NMDA

Новый модуль, соответствующий NMDA, воздаётся с использованием субдеревьев конфигурации и состояния, когда это возможно.

```
module example-foo {
  namespace "urn:example.com:params:xml:ns:yang:example-foo";
  prefix "foo";

  container foo {
    // Рабочие значения дочерних узлов данных конфигурации в
    // хранилище рабочего состояния могут содержать лишь
    // узлы config false, если они нужны.
  }
}
```

4.23.3.3. Пример преобразования модуля не-NMDA

Несовместимые объекты не следует удалять из существующих модулей. Вместо этого для них устанавливается статус deprecated. В какой-то момент (обычно по истечении 1 года) статус **можно** сменить на obsolete.

Старый модуль

```
module example-foo {
  namespace "urn:example.com:params:xml:ns:yang:example-foo";
  prefix "foo";

  container foo {
    // Дочерние узлы данных конфигурации
  }

  container foo-state {
    config false;
    // Дочерние узлы рабочего состояния
  }
}
```

Преобразованный модуль NMDA

```
module example-foo {
  namespace "urn:example.com:params:xml:ns:yang:example-foo";
  prefix "foo";

  container foo {
    // Рабочие значения дочерних узлов данных конфигурации в
    // хранилище рабочего состояния могут содержать лишь
    // узлы config false, если они нужны (из старого foo-state).
  }

  // сохранить исходный foo-state, но сменить статус на deprecated
  container foo-state {
    config false;
    status deprecated;
    // Дочерние узлы рабочего состояния
  }
}
```

4.23.3.4. Пример создания временного модуля NMDA

Создаётся новый модуль, содержащий узлы верхнего уровня для данных рабочего состояния, которые будут доступны до объединения с узлами данных конфигурации (т. е. совместимости с NMDA).

```
module example-foo-state {
  namespace "urn:example.com:params:xml:ns:yang:example-foo-state";
  prefix "foo-s";

  // Импорт нового или конвертированного модуля (в примере не применяется)
  import example-foo { prefix foo; }

  container foo-state {
    config false;
    status deprecated;
    // Дочерние узлы рабочего состояния
  }
}
```

4.24. Вопросы производительности

В общем случае некоторые операторы YANG требуют при работе больших ресурсов, нежели другие операторы. Хотя при проверке YANG требования производительности не задаются, при разработке моделей данных YANG можно учитывать приведённые ниже аспекты:

- списки обычно «дороже» контейнеров;
- оценка оператора when обычно дороже, чем операторов if-feature или choice;
- операторы must обычно дороже, чем min-entries, max-entries, mandatory, unique;
- листья identityref обычно дороже, чем листья enumeration;

- типы leafref и instance-identifier с require-instance true обычно дороже, чем при установке require-instance false.

4.25. Рассмотрение открытых систем

В реализации можно считать присутствующими лишь модули, импортируемые конкретным модулем из других модулей. Открытая система **может** включать любые комбинации модулей YANG.

4.26. Рекомендации для конструкций YANG 1.1

Набор рекомендация для YANG 1.1 будет расширяться по мере обретения опыта использования новых функций языка. В этом параграфе приведён первый набор рекомендаций для новых функций языка YANG 1.1.

4.26.1. Импорт нескольких выпусков

Стандартным модулям **не следует** импортировать несколько версий одного модуля. Так поступать **можно** при необходимости использовать независимые определения (например, enumeration typedef) из конкретных выпусков импортируемого модуля.

4.26.2. Использование логики feature

Логика функций в YANG 1.1 значительно выразительней, чем в YANG 1.0. Операторам description **следует** описывать логику if-feature в форме текста, чтобы помочь читателям понять модуль.

Функции YANG (feature) **следует** применять вместо оператора when, если это возможно. Функции анонсируются сервером, а объекты, с которыми связаны условия операторов if-feature группируются концептуально. Такой общности нет в операторах when.

Реализация функций на сервере обычно проще, а ресурсов при работе требуется меньше, чем при использовании операторов when. Функции в общем случае статичны (т. е. задаются при загрузке модуля и не меняются в процессе работы). Для оператора when каждое редактирование клиента может приводить к изменениям.

4.26.3. Операторы anyxml и anydata

Оператор anyxml **недопустимо** применять для представления концептуального субдерева узлов данных YANG. Взамен **должен** использоваться оператор anydata.

4.26.4. Операторы action и rpc

Использование операторов action или rpc определяется разработчиком. Операции RPC не связаны с каким-либо конкретным узлом данных, а действия (action) связаны с конкретным определением узла. Оператор action **следует** применять в операциях протокола, связанных с подмножеством узлов данных, а не со всеми возможными узлами.

Одно и то же имя действия **может** применяться в разных определениях разных узлов данных. Например, действие reset в определении узла данных для интерфейса может иметь иные параметры, нежели в определении для источника питания или VLAN. **Следует** применять одно имя действия для представления похожей семантики.

Модель управления доступом NETCONF (NETCONF Access Control Model или NACM) [RFC8341] не поддерживает управление доступом на основе параметров для операций RPC. Пользователю предоставляется (или не предоставляется) возможность вызова операции RPC, независимо от её параметров. Например, если каждому клиенту разрешено сбрасывать лишь свой интерфейс, NACM не может применяться, поскольку не может обеспечить контроль доступа на основе значения параметра interface (управляет лишь доступом к операции reset).

```
rpc reset {
  input {
    leaf interface {
      type if:interface-ref;
      mandatory true;
      description "Интерфейс для сброса.";
    }
  }
}
```

Однако NACM может обеспечивать контроль доступа для отдельных экземпляров интерфейсов с использованием действия reset. Если у пользователя нет доступа для чтения к конкретному экземпляру interface, он не сможет вызвать действие reset для этого экземпляра.

```
container interfaces {
  list interface {
    ...
    action reset { }
  }
}
```

4.27. Обновление модулей YANG (опубликованных и неопубликованных)

Модули YANG могут меняться со временем. Обычно новые определения модели данных нужны для поддержки новых функций. Для опубликованных модулей **должны** выполняться правила обновления YANG из раздела 11 в [RFC7950]. Они **могут** применяться и для непубликуемых модулей.

Правила обновления YANG применяются лишь для опубликованных выпусков модулей. Каждая организация может применять свой способ определения опубликованных работ, которые считаются стабильными и неопубликованных работ, которые стабильными не считаются. Например, в IETF для опубликованных работ используются RFC, для неопубликованных - I-D.

5. Взаимодействие с IANA

Перечисленные ниже регистрации в субреестре ns реестра IETF XML Registry [RFC3688] были уточнены в [RFC6087] и обновлены IANA с указанием этого документа.

```
URI: urn:ietf:params:xml:ns:yang:ietf-template
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

Приведённые в таблице назначения были уточнены в [RFC6087] и обновлены IANA в реестре YANG Module Names. Этот документ также был добавлен в ссылку для самого реестра YANG Module Names как содержащий шаблон, требуемый для регистрации, в Приложении B.

Назначение реестров YANG.

Поле	***Значение***
Name	ietf-template
Namespace	urn:ietf:params:xml:ns:yang:ietf-template
Prefix	temp
Reference	RFC8407

6. Вопросы безопасности

Этот документ даёт рекомендации для содержимого NETCONF и RESTCONF, определяемого с использованием языка моделирования данных YANG, поэтому не вносит новых и не меняет имеющихся рисков безопасности для систем управления.

7. Литература

7.1. Нормативные документы

[ID-Guidelines] Housley, R., "Guidelines to Authors of Internet-Drafts", December 2010, <https://www.ietf.org/standards/ids/guidelines/>.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://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, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5378] Bradner, S., Ed. and J. Contreras, Ed., "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, DOI 10.17487/RFC5378, November 2008, <https://www.rfc-editor.org/info/rfc5378>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

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

7.2. Дополнительная литература

[IANA-MOD-NAMES] IANA, "YANG Module Names", <https://www.iana.org/assignments/yang-parameters/>.

[IANA-XML] IANA, "IETF XML Registry", <https://www.iana.org/assignments/xml-registry/>.

[RFC-STYLE] RFC Editor, "Style Guide", <http://www.rfc-editor.org/styleguide/>.

[RFC2026] Bradner, S., "The Internet Standards Process — Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC4151] Kindberg, T. and S. Hawke, "The 'tag' URI Scheme", RFC 4151, DOI 10.17487/RFC4151, October 2005, <https://www.rfc-editor.org/info/rfc4151>.

[RFC4181] Heard, C., Ed., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, DOI 10.17487/RFC4181, September 2005, <https://www.rfc-editor.org/info/rfc4181>.

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <https://www.rfc-editor.org/info/rfc6087>.

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

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[RFC7322] Flanagan, H. and S. Ginoza, "RFC Style Guide", RFC 7322, DOI 10.17487/RFC7322, September 2014, <https://www.rfc-editor.org/info/rfc7322>.

[RFC7841] Halpern, J., Ed., Daigle, L., Ed., and O. Kolkman, Ed., "RFC Streams, Headers, and Boilerplates", RFC 7841, DOI 10.17487/RFC7841, May 2016, <https://www.rfc-editor.org/info/rfc7841>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

Приложение A. Контрольный список рецензирования модуля

Этот раздел является переработкой текста из RFC 4181.

Целью рецензирования модуля YANG является проверка его технической корректности и соответствия требованиям IETF к документации. Приведённый ниже список может помочь при рецензировании документов I-D.

- Шаблон I-D (Boilerplate). Проверяется указание в документе шаблона I-D (см. https://www.ietf.org/id-info/guidelines.html), включая соответствующее заявление для разрешения публикации RFC и отсутствие в шаблоне I-D ссылок и номеров разделов.

- Аннотация (Abstract). Проверяется отсутствие в аннотации ссылок и номера параграфа, а также соответствие рекомендациям https://www.ietf.org/id-info/guidelines.html.

- Авторские права (Copyright Notice). Проверяется наличие корректного текста, относящегося к правам, которые участники работы предоставили IETF Trust [RFC5378]. Проверяется наличие полного указания прав IETF Trust в начале документа. Документ IETF Trust Legal Provisions (TLP) доступен по ссылке https://trustee.ietf.org/license-info/.

- Раздел «Вопросы безопасности» (Security Considerations). Проверяется использование в документе последнего одобренного шаблона с сайта направления Operations and Management (OPS) (https://trac.ietf.org/area/ops/trac/wiki/yang-security-guidelines) и соблюдение приведённых там рекомендаций.

- Раздел «Взаимодействие с IANA (IANA Considerations). Это обязательный раздел и для каждого модуля в документе этот раздел должен включать записи для реестров:

  XML Namespace Registry - реестр пространств имён модулей YANG;

  YANG Module Registry - реестр имён, префиксов пространств имён и номеров RFC для модулей YANG в соответствии с правилами [RFC6020].

- Литература (References). Проверяется корректность разделения ссылок на нормативные и информационные, включение RFC 2119 и RFC 8174 в нормативные ссылки если используются упомянутые в этих документах термины, наличие всех ссылок, указанных в шаблоне, указание всех импортируемых и цитируемых модулей YANG в разделе нормативных ссылок, указание цитат из последних действующих RFC, если нет действительной причины указывать иные (например, включение информационной ссылки на предыдущую публикацию для разъяснений свойства, обеспечивающего совместимость с прежними версиями). Проверяется наличие ссылок на импортируемые модули в тексте документа (вне модуля YANG). Если модуль YANG содержит ссылки или операторы description, указывающие I-D, этот документ включается в информационные ссылки.

- Лицензия. Проверяется указание Simplified BSD License в каждом модуле и субмодуле YANG. Некоторые рекомендации для этого требования даны в параграфе 3.1. Проверяется корректность года в указании авторских прав и использование одобренного текста из последнего документа TLP, доступного по ссылке https://trustee.ietf.org/license-info/.

- Другие вопросы. Проверяется отсутствие проблем, отмеченных в https://www.ietf.org/id-info/checklist.html.

- Техническое содержание. Рецензируется фактическое содержание документа на предмет соответствия рекомендациям этого документа. Рекомендуется использовать компилятор модуля YANG для обнаружения синтаксических ошибок. Список свободно распространяемых инструментов и другая информация, включая советы по форматированию, доступны по ссылкам https://trac.ietf.org/trac/netconf/wiki и https://trac.ietf.org/trac/netmod/wiki

Проверка корректности синтаксиса является лишь частью работы. Важно прочитать документ с модулем YANG с точки зрения его возможной реализации. Особенно важно проверить операторы description на предмет чёткости и однозначности описания, чтобы можно было создавать совместимые реализации.

Приложение B. Шаблон модуля YANG

```
<CODE BEGINS> file "ietf-template@2016-03-20.yang"

module ietf-template {
  yang-version 1.1;

  // Замените строку уникальным значением URN пространства имён
  namespace "urn:ietf:params:xml:ns:yang:ietf-template";

  // Замените строку, пытаясь указать уникальный префикс
  prefix temp;

  // Здесь помещаются операторы import, например,
  // import ietf-yang-types { prefix yang; }
  // import ietf-inet-types { prefix inet; }
  // Указывается рабочая группа IETF, если это применимо.

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  // Укажите реальные данные для контактов
  contact
    "WG Web:   <http://datatracker.ietf.org/wg/your-wg-name/>
     WG List:  <mailto:your-wg-name@ietf.org>

     Editor:   your-name
               <mailto:your-email@example.com>";

  // Замените первое предложение в этом операторе description.
  // Замените сведения об авторских правах наиболее свежими,
  // если они обновились после публикации этого документа.
  description
    "Этот модуль задаёт шаблон для других модулей YANG.

     Copyright (c) <год>. Авторские права принадлежат IETF Trust
     и лицам, указанным как авторы документа. Все права защищены.

     Распространение и применение модуля в исходной или двоичной
     форме с изменениями или без таковых разрешено в соответствии с
     лицензией Simplified BSD License, изложенной в параграфе 4.c
     IETF Trust's Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

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

  // RFC Ed.: замените XXXX фактическим номером RFC и удалите этот
  // комментарий.
  // Замените 2016-03-20 фактической датой публикации модуля в
  // формате (год-месяц-число)
  revision 2016-03-20 {
    description
      "Что изменено в этом выпуске";
    reference "RFC XXXX: <Replace With Document Title>";
  }

  // Операторы extension
  // Операторы feature
  // Операторы identity
  // Операторы typedef
  // Операторы grouping
  // Операторы определения данных
  // Операторы augment
  // Операторы rpc
  // Операторы notification
  // НЕ ВКЛЮЧАЙТЕ операторы deviation в публикуемые модули
}

<CODE ENDS>
```

Благодарности

Структура и содержимое документа основаны на документе «Guidelines for Authors and Reviewers of MIB Documents» [RFC4181], созданном C. M. Heard.

Рабочая группа благодарна Martin Bjorklund, Juergen Schoenwaelder, Ladislav Lhotka, Jernej Tuljak, Lou Berger, Robert Wilton, Kent Watsen, William Lupton за их всеобъемлющие рецензии и вклад в этот документ.

Адрес автора

Andy Bierman

YumaWorks

Email: andy@yumaworks.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru