Network Working Group
Request for Comments: 3986
STD: 66
Updates: 1738
Obsoletes: 2732, 2396, 1808
Category: Standards Track

T. Berners-Lee
W3C/MIT
R. Fielding
Day Software
L. Masinter
Adobe Systems

January 2005

Базовый синтаксис идентификаторов URI

Uniform Resource Identifier (URI): Generic Syntax

Статус документа

В этом документе содержится проект стандарта для протокола Internet, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Авторские права

Тезисы

Унифицированный идентификатор ресурса (URI[1]) представляет собой компактную последовательность символов, указывающую абстрактный или физический ресурс. Данная спецификация определяет базовый синтаксис URI и процесс преобразования ссылок URI, которые могут иметь относительную форму, а также содержит рекомендации и обсуждение вопросов безопасности при использовании URI в сети Internet. Синтаксис URI определяет грамматику, которая является надмножеством всех корректных URI, позволяющую реализациям разбирать основные компоненты ссылки URI без наличия информации о конкретных требованиях схемы для каждого из возможных идентификаторов. Спецификация не определяет порождающую грамматику для URI — такая задача выполняется индивидуальными спецификациями для каждой схемы URI.

Оглавление

[1]Uniform Resource Identifier.

1. Введение

Идентификаторы URI обеспечивают простой и расширяемый способ указания ресурсов. Эта спецификация синтаксиса и семантики URI основана на концепциях, введенных глобальной информационной инициативой WWW[1], где использование таких идентификаторов началось с 1990-х и было описано в документе Universal Resource Identifiers in WWW [RFC1630]. Синтаксис разработан в соответствии с документами Functional Recommendations for Internet Resource Locators [RFC1736] и Functional Requirements for Uniform Resource Names [RFC1737].

Этот документ отменяет [RFC2396], который объединил в себе документы Uniform Resource Locators [RFC1738] и Relative Uniform Resource Locators [RFC1808] для определения единого базового синтаксиса URI. Документ отменяет также [RFC2732], где был введен синтаксис для адресов IPv6. Он исключает часть RFC 1738, определяющую специфический синтаксис для отдельных схем URI — эти части были обновлены в отдельных документах. Процесс регистрации новых схем URI определен отдельно в [BCP35]. Рекомендации для разработчиков новых схем URI можно найти в [RFC2718]. Все существенные отличия от RFC 2396 отмечены в Приложении D.

В этом документе термины «символ» (character) и «кодированный набор символов» (coded character set) используются в соответствии с определением [BCP19], а термин «кодировка символов» (character encoding) в соответствии с определением [BCP19] для «набора символов» (charset).

1.1. Обзор URI

Идентификаторы URI имеют перечисленные ниже особенности.

Унификация (Uniform)

Унификация обеспечивает несколько преимуществ. Можно применять разные типы идентификаторов ресурсов в одном контексте даже при использовании различных механизмов доступа к этим ресурсам. Обеспечивается единая смысловая интерпретация общих семантических соглашений для разнотипных идентификаторов ресурсов. Можно добавлять новые типы идентификаторов без нарушения использования имеющихся типов. Можно использовать одни идентификаторы в разных контекстах, что позволяет новым приложениям или протоколам воспользоваться существующими и широко распространенными наборами идентификаторов.

Ресурс (Resource)

Данная спецификация не ограничивает сферу того, что может быть отнесено к ресурсам. Термин «ресурс» используется в самом общем смысле, обозначая нечто, идентифицируемое с помощью URI. Примерами могут быть электронные документы, изображения, источники информации определенного назначения (например, текущая погода в Лос-Анжелесе), службы (например, шлюз HTTP-SMS) и наборы других ресурсов. Ресурс не обязательно

[1] World Wide Web - «всемирная паутина».

доступен через Internet — например, люди, корпорации, книги в библиотеке могут выступать в качестве ресурсов. Подобно этому, ресурсами могут быть и абстрактные категории типа операторов и операндов в математических выражениях, типы отношений (например, родитель или работник), численные значения (например, 0, 1, ∞).

Идентификатор (Identifier)

Идентификатор олицетворяет информацию, требуемую для того, чтобы отличить идентифицируемое от всего остального в области идентификации. Использование термина «идентификация» относится к цели отличить ресурс от всех других ресурсов, независимо от способа достижения этой цели (например, по имени, адресу или контексту). Не следует считать, что идентификатор определяет или олицетворяет то, на что он указывает, хотя в некоторых случаях это может наблюдаться. Не следует предполагать, что система, использующая URI, получит доступ к тому или иному ресурсу — во многих случаях URI используются лишь для обозначения ресурсов, никак не способствуя доступу к ним. Не следует также считать, что идентифицируемый ресурс является «точечным» по своей природе (например, ресурс может оказаться именованным множеством или отображением, которое время от времени меняется).

URI является идентификатором, состоящим из последовательности символов, соответствующей синтаксическим правилам <URI> в разделе 3. Он обеспечивает однотипную идентификацию ресурсов через определяемый раздельно набор схем именования (параграф 3.1). Способы выполнения, назначения и разрешения определяются в спецификациях конкретных схем.

Данная спецификация не задает каких-либо ограничений на природу ресурсов, причины, по которым приложения могут обращаться к ресурсам, или типы систем, которые могут применять URI для идентификации ресурсов. Спецификация не требует, чтобы URI, идентифицирующие определенные ресурсы, были неизменными во времени, хотя это и служит одной из общих целей схем URI. Тем не менее, данная спецификация не препятствует приложениям вводить свои ограничения на типы используемых ресурсов или выбирать подмножества URI с желаемыми для приложения характеристиками.

URI имеют глобальную область действия и интерпретируются независимо от контекста, хотя результат интерпретации может быть связан с контекстом конкретного конечного пользователя. Например, http://localhost/ имеет одинаковую интерпретацию для любого пользователя этой ссылки, хотя сетевой интерфейс, соответствующий имени localhost может различаться для каждого пользователя — интерпретация не зависит от доступа. Однако выполняемые на основе такой ссылки действия происходят в контексте конечного пользователя, что предполагает использование для уникальных в глобальном масштабе ресурсов использование URI, который отличает этот ресурс от всех прочих. Идентификаторы URI, относящиеся к локальному контексту конечного пользователя следует применять только в том случае, когда сам контекст определяет аспекты ресурса (например, справочная система ссылается на определенный файл в компьютере конечного пользователя типа file:///etc/hosts).

1.1.1. Базовый синтаксис

Каждый идентификатор URI начинается с имени схемы, как описано в параграфе 3.1, которое указывает спецификацию для выделения идентификаторов в рамках этой схемы. Таким образом, синтаксис URI обеспечивает федеративную, расширяемую систему именования, где спецификация каждой схемы может дополнительно ограничивать синтаксис и семантику используемых в этой схеме идентификаторов.

Данная спецификация определяет синтаксические элементы URI используемые во всех или многих схемах URI. Таким образом, определяется синтаксис и семантика, требуемые для реализации независимых от схем механизмов разбора ссылок URI, благодаря чему зависящую от схемы обработку URI можно отложить до того момента, когда потребуется зависящая от схемы семантика URI. Аналогично, протоколы и форматы, использующие ссылки URI, могут использовать эту спецификацию, как определение синтаксиса, разрешенного для всех URI, включая и схемы, которые еще не определены. Это отвязывает развитие схем идентификации от развития протоколов, форматов данных и реализаций, использующих URI.

Анализатор базового синтаксиса URI может разобрать любую ссылку URI на основные компоненты. После определения схемы для компонент может быть применен специфический для этой схемы разбор. Иными словами, базовый синтаксис URI является надмножеством синтаксиса всех схем URI.

1.1.2. Примеры

Ниже приведены примеры нескольких URI, иллюстрирующие разные схемы URI и варианты компонент базового синтаксиса.

```
ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.ietf.org/rfc/rfc2396.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2
```

1.1.3. URI, URL и URN

URI можно разделить на локацию, имя или оба вместе. Термин URL[1] обозначает подмножество URI, которое в дополнение к идентификации ресурса указывает его «расположение» путем обозначения основного механизма доступа (например, его «расположения» в сети). Термин URN[2] исторически применялся для обозначения обоих типов URI в схеме urn [RFC2141] — уникальных в глобальном масштабе и неизменных даже в случаях прекращения существования ресурса или утраты доступа к нему имен, а также любых других URI со свойствами имени.

Отдельные схемы не нужно классифицировать, как «локатор» или «имя». Экземпляры URI из любой данной схемы могут иметь характеристики имени, локации или обоих сразу (зачастую, в зависимости от действий агентства по

[1]Uniform Resource Locator — унифицированный локатор ресурса.
[2]Uniform Resource Name — унифицированное имя ресурса.

именованию, а не особенностей схемы). В будущих спецификациях и связанных с ними документах следует применять общий термин URI, а не более ограниченные термины URL и URN [RFC3305].

1.2. Внутреннее устройство

1.2.1. Транскрипция

Одним из основных соображений при разработке синтаксиса URI была глобальная транскрипция. URI представляет собой последовательность символов из весьма ограниченного набора — буквы латиницы, цифры и несколько специальных символов. URI можно представить разными способами- например, записать на бумаге, вывести на экран или передать в виде последовательности октетов. Интерпретация URI зависит только от использованных символов, а не от способа представления этих символов в сетевом протоколе.

Цели транскрипции можно показать на простом примере. Представим себе двух коллег — Сема и Кима, сидящих в пабе на международной конференции и обменивающихся между собой идеями. Сэм спрашивает у Кима, где можно получить дополнительную информацию и Ким пишет для него URI исследовательского сайта на салфетке. По возвращении домой Сэм вводит URI с салфетки в свой компьютер и находит информацию, о которой говорил Ким.

Этот пример демонстрирует некоторые соображения, которые следует принимать во внимание:

- URI является последовательностью символов, которые не обязательно представляются последовательностью октетов;
- URI можно получить не только через сеть, поэтому ему следует состоять из символов, которые могут быть введены в компьютер с учетом ограничений, присущих клавиатурам (и другим устройствам ввода) в части языка и кодировок символов;
- URI зачастую приходится запоминать, поэтому для человека важно присутствие в составе URI понятных и знакомых компонент.

Эти соображения не всегда согласуются между собой. Например, достаточно часто наиболее значимая компонента имени в URI будет требовать использования символов, которые не могут быть введены в некоторых системах. Возможность переноса идентификаторов ресурсов из одной среду в другую является более важным свойством, нежели осмысленность отдельных компонент URI.

В локальном или региональном контексте по мере развития технологий пользователи будут получать преимущества от возможности применения более широкого набора символов, включая отсутствующие в данной спецификации. Для представления символов, не входящих в кодировку US-ASCII, в URI возможно %-представление (percent-encoding), описанное в параграфе 2.1, если такое представление разрешает используемая схема и протокольный элемент, в который включается URI. Такие определения должны учитывать кодировку используемых символов для их отображения на октеты до использования %-представления для URI.

1.2.2. Отделение идентификации от взаимодействия

Основным недоразумением, связанным с URI, является представление, что эти идентификаторы используются только для ссылок на доступные ресурсы. URI, сам по себе, лишь обеспечивает идентификацию ресурса, а доступ к такому ресурсу не гарантируется и не предполагается в URI. Вместо этого любая операция, связанная со ссылкой URI, определяется протокольным элементом, атрибутом формата данных или текстом на естественном языке, в котором присутствует этот идентификатор.

На основе URI система может попытаться выполнить по отношению к ресурсу различные операции, которые можно описать словами «доступ», «обновление», «замена», поиск атрибутов». Такие операции определяются протоколами, которые могут использовать URI, а вовсе не данной спецификацией. Однако мы используем некоторые термины общего назначения для базовых операций с URI. Преобразование URI (resolution) представляет собой процесс определения механизма доступа и подходящих параметров, требуемых для разыменования URI. Такое преобразование может потребовать нескольких итераций. Для использования механизма доступа с целью выполнения действия по отношению к указанному URI ресурсу нужно «разыменовать» URI.

Когда URI используются в информационно-поисковых системах для идентификации источника данных, наиболее распространенной формой разыменования URI является «извлечение» - возможность использовать URI для того, чтобы получить представления связанного с идентификатором ресурса. «Представление» является последовательностью октетов в комбинации с метаданными, описывающими эти октеты, которые совместно составляют запись о состоянии ресурса на момент генерации представления. Извлечение достигается с помощью процесса, который может включать использование URI в качестве кэш-ключа для проверки локально кэшированного представления (если оно имеется) и разыменования URI для применения операции извлечения (поиска). В зависимости от используемого для поиска протокола может быть предоставлена дополнительная информация о ресурсе (метаданные) и его связях с другими ресурсами.

Ссылки URI в информационно-поисковых системах рассчитаны на отложенную привязку (late-binding) - результат доступа в общем случае определяется в момент доступа и может существенно меняться в зависимости от времени и других аспектов взаимодействия. Эти ссылки создаются для использования в будущем — то, что идентифицируется, не является неким конкретным результатом, полученным раньше, а представляет некие характеристики, которые предполагаются верными для будущих результатов. В таких случаях ресурс, указанный URI, реально представляет собой единообразие характеристик, наблюдаемых с течением времени, возможно сопровождаемых некоторыми комментариями и заявлениями, сделанными держателем (провайдером) ресурса.

Хотя имена многих схем URI включают протокол, это не предполагает, что использование этих URI обеспечит доступ к ресурсу по этому протоколу. URI зачастую используются просто для идентификации. Даже при использовании URI для поиска представления ресурса доступ может выполняться через шлюзы, прокси, системы кэширования, службы преобразования имен, которые не зависят от протокола, связанного с именем схемы. Преобразование некоторых URI может потребовать использования не одного протокола (например, DNS и HTTP обычно требуются для доступа к http URI, если представление отсутствует в локальном кэше).

1.2.3. Иерархические идентификаторы

Синтаксис URI организован иерархически с компонентами, перечисленными в порядке убывания значимости слева направо. Для некоторых схем URI видимая иерархия ограничивается самой схемой — все, что указано после разграничителя компонент схемы (:) считается непрозрачным для обработки URI. Другие схемы URI деляют иерархия явной и видимой для алгоритмов разбора общего назначения.

Базовый синтаксис использует символы / (дробная черта), ? (знак вопроса) и # (знак номера) для разграничения компонент, которые значимы для иерархической интерпретации идентификатора средствами разбора общего назначения. В дополнение к улучшению читаемости таких идентификаторов за счет согласованного использования знакомого синтаксиса это унифицированное представление иерархии в схемах именования позволяет создавать независимые от схемы ссылки относительно такой иерархии.

Зачастую организуется группа или «дерево» документов, служащих общей цели, где большинство ссылок URI указывает на ресурсы внутри этого дерева. Аналогично, документы, расположенные на конкретном сайте гораздо чаще будут ссылаться на ресурсы этого сайта, нежели на ресурсы других. Относительные ссылки URI позволяют обеспечить частичную независимость дерева документов от места его расположения и схемы доступа. Например, один и тот же набор гипертекстовых документов может быть доступен и переносим с помощью схем file, http и ftp, если в документах другие документы указаны относительными ссылками. Кроме того, такое дерево документом можно целиком перенести в другое место, не меняя относительных ссылок.

Относительные ссылки (параграф 4.2) указывают на ресурс путем описания разницы в иерархическом пространстве имен между контекстом ссылки и целевым URI. Алгоритм преобразования ссылок, описанный в разделе 5, определяет способ преобразования таких ссылок в целевые URI. Поскольку относительные ссылки могут применяться только в контексте иерархических URI, разработчикам новых схем URI следует использовать синтаксис, согласующийся с базовыми иерархическими компонентами, если нет важный причин для отказа от использования в схеме относительных ссылок.

> Примечание. В предыдущих спецификациях использовались термины partial URI и relative URI для обозначения относительных ссылок URI. Поскольку некоторых такая терминология вводила в заблуждение и они считали относительные URI неким подмножеством URI, а не методом указания ссылок URI, в данной спецификации используется термин «относительные ссылки» (relative reference).

Все ссылки URI разбираются анализатором базового синтаксиса. Однако, поскольку иерархическая обработка не оказывает влияния на абсолютные URI, используемые в ссылках, не содержащих сегментов с точками (сегменты полного пути с символами «.» и «..», как описано в параграфе 3.3), спецификации схем URI могут определять «непрозрачные» идентификаторы для запрета использования символов дробной черты, вопросительного знака и URI вида «scheme:.» и «scheme:..».

1.3. Синтаксические обозначения

В этой спецификации применяются обозначения ABNF[1] [RFC2234], включая основные элементы синтаксиса ABNF, используемые в данной спецификации, - ALPHA (буквы), CR (возврат каретки), DIGIT (десятичные цифры), DQUOTE (двойные кавычки), HEXDIG (шестнадцатеричные цифры), LF (перевод строки) и SP (пробел). Полный синтаксис URI собран в Приложении A.

2. Символы

Синтаксис URI предоставляет метод идентификации ресурсов с помощью последовательности символов. Символы URI, в свою очередь, часто кодируются в форме октетов для передачи или представления. Эта спецификация не задает какого-либо конкретного способа отображения между символами URI и октетами, служащими или передачи этих символов. Когда URI включается в элемент протокола, кодировка символов определяется этим протоколом и без такого определения в URI предполагается такая же кодировка символов, как в окружающем тексте.

Нотация ABNF определяет терминальные значения, как неотрицательные целые числа (код — codepoint) на базе кодировки US-ASCII [ASCII]. Поскольку URI является последовательностью символов, нужно обратить эту связь для понимания синтаксиса URI. Следовательно, целые числа, используемые ABNF должны быть отображены обратно на соответствующие символы US-ASCII для завершения синтаксических правил.

URI состоят из ограниченного набора символов, включающего цифры, буквы и несколько специальных знаков. Зарезервированное подмножество этих символов может использоваться для разграничения синтаксических компонент внутри URI тогда, как оставшиеся символы (включая незарезервированные и зарезервированные, но не используемые в качестве разграничителей) определяют каждую компоненту идентификатора.

2.1. %-представление

Механизм %-кодирования служит для представления в компонентах идентификаторов октетов данных, для которых нет символов в числе разрешенных или соответствующий октету символ является разделителем компонент или внутри компоненты. Октет с %кодированием представляется тремя символами — сначала следует знак №, а за ним две шестнадцатеричных цифры, определяющие значение октета. Например, "%20" будет представлять двоичный октет 00100000 (ABNF - %x20), которому в кодировке US-ASCII соответствует символ пробела (SP). Применение кодирования и декодирования с использованием % описано в параграфе 2.4.

```
pct-encoded = "%" HEXDIG HEXDIG
```

Заглавные буквы A - F в качестве шестнадцатеричных цифр эквивалентны буквам a - f. Если два URI отличаются только регистром букв в шестнадцатеричных цифрах при использовании %-кодирования, эти идентификаторы являются эквивалентными. Для согласованности при создании и нормализации URI следует использовать заглавные буквы в %-представлении.

[1] Augmented Backus-Naur Form.

2.2. Зарезервированные символы

URI включают компоненты и субкомпоненты, разграниченные символами из «зарезервированного» набора. Эти символы называют «зарезервированными», поскольку они могут (но не обязаны) применяться в качестве разграничителей в базовом синтаксисе, синтаксисе отдельных схем или синтаксисе приложений при разыменовании URI. Если какой-либо из зарезервированных символов требуется включить в состав компоненты URI (не в качестве разграничителя), для него должно использоваться %-кодирование до вставки символа в URI.

```
reserved    = gen-delims / sub-delims
gen-delims  = ":" / "/" / "?" / "#" / "[" / "]" / "@"
sub-delims  = "!" / "$" / "&" / "'" / "(" / ")" / "*" / "+" / "," / ";" / "="
```

Целью резервирования является выделение набора символов разграничения, отличимых от остальных символов URI. Идентификаторы URI, различающиеся заменой зарезервированных символов %-представлениями, не являются эквивалентными. %-кодирование зарезервированных символов и декодирование октетов с %представлениям зависит от интерпретации URI в большинстве приложений. Символы зарезервированного набора защищены от нормализации и, следовательно, могут без опаски применяться в определяемых схемой или производителем алгоритмах разграничения для отделения друг от друга компонент URI.

Отдельное подмножество зарезервированных символов (gen-delims) применяется в качестве базовых разграничителей компонент URI, как описано в разделе 3. Синтаксические правила ABNF для компонент не будет использовать символы reserved и gen-delims напрямую, вместо этого каждое синтаксическое правило указывает разрешенные для компоненты символы (т. е., символы, не служащие разграничителями) и, если в числе таких символов оказываются зарезервированные, они могут использоваться для разграничения субкомпонент внутри компоненты. Данная спецификация определяет лишь наиболее общие субкомпоненты, а спецификации схем URI или синтаксис конкретной реализации алгоритма разыменования URI могут определять дополнительные символы разграничения из числа разрешенных для компоненты.

Создающим URI приложениям следует использовать %-представления октетов, соответствующих символа из числа зарезервированных, если эти символы специально не разрешены схемой URI для представления данных в компонентах. Если в компоненте URI обнаружен зарезервированный символ, для которого не известно правило разделения, этот символ должен интерпретироваться, как октет данных, соответствующий символу US-ASCII.

2.3. Незарезервированные символы

Символы, разрешенные для применения в URI, но не зарезервированные в качестве разграничителей, называют незарезервированными (unreserved). К таким символам относятся строчные и прописные буквы, десятичные цифры, дефис, точка, символ подчеркивания и тильда.

```
unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
```

Идентификаторы URI, различающиеся заменой незарезервированных символов %-представлениями, не являются эквивалентными — они указывают один и тот же ресурс. Однако при реализации сравнения URI нормализация не всегда выполняется до сравнения (см. раздел 6). Для согласованности %-кодирование октетов ALPHA (%41-%5A и %61-%7A), DIGIT (%30-%39), дефиса (%2D), точки (%2E), подчеркивания (%5F) или тильды (%7E) не следует применять при создании URI, а при обнаружении такого представления в URI следует сначала выполнить соответствующее декодирование с помощью нормализатора URI.

2.4. Кодирование и декодирование

В обычных обстоятельствах существует единственный момент, когда для октетов внутри URI используется %-кодирование в процессе создания URI из компонент — момент, когда реализация определяет, какие из зарезервированных символов будут разграничивать компоненты, а какие можно безопасно использовать для данных. После создания URI всегда хранится в %-представлении.

При разыменовании URI компоненты и субкомпонеты, значимые для процесса разыменования конкретной схемы (если таковые есть), должны анализироваться и разделяться до того, как октеты с %-кодированием в этих компонентах будут безопасно декодированы, поскольку в противном случае данные могут быть ошибочно истолкованы, как разграничитель. Единственное исключение применяется для %-кодированных октетов, соответствующих незарезервированным символам, которые можно декодировать в любой момент. Например, октет, соответствующий тильде (~), старые реализации URI часто представляют в форме %7E и последовательность %7E может быть заменена символом ~ без изменения интерпретации.

Поскольку символ % служит индикатором специального представления символов, для него должно применяться %-кодирование %25, если символ нужен в составе URI. Реализациям недопустимо использовать %-кодирование или декодирование одной строки более одного раза, поскольку декодирование уже декодированной строки может приводить к ошибочной интерпретации октета с символом %, как начала нового %-представления и наоборот в случае %-кодирования уже кодированной строки.

2.5. Идентификационные данные

Символы URI обеспечивают идентификацию данных для каждой компоненты URI, выступая в качестве внешнего интерфейса для идентификации между системами. Хотя существование и природа интерфейса создания URI скрыты от клиентов, использующих URI (и, таким образом, находятся за пределами требований интероперабельности, задаваемых данной спецификацией), они зачастую являются источником недоразумений и ошибок при интерпретации символов URI. Разработчикам следует принимать во внимание наличие множества кодировок символов в процессе создания и передачи URI — локальная кодировка имен и данных, кодировка публичного интерфейса, кодировка символов URI, кодировка формата данных и протокольная кодировка.

Локальные имена типа имен системных файлов сохраняются в локальной кодировке символов. Создающие URI приложения (например, серверы-источники) обычно используют локальную кодировку в качестве основы для создания осмысленных имен. Производитель URI преобразует локальную кодировку в одну из подходящих для публичного интерфейса и потом преобразует созданное представление в ограниченный набор символов URI (зарезервированные и незарезервированные символы, %-представление). Эти символы, в свою очередь, кодируются в октеты, которые

используются форматом представления данных (например, кодировка документа), а данные зачастую передаются с использованием кодирования в протоколах Internet.

Для большинства систем незарезервированные символы в компонентах URI интерпретируются, как октеты данных, соответствующие представлению данного символа в кодировке US-ASCII. Потребители URI считают, что буква X соответствует октету 01011000 и даже в тех случаях, когда подобное допущение некорректно, оно не наносит вредя. Системы с внутренними идентификаторами в другой кодировке (например, EBCDIC) обычно выполняют трансляцию символов в текстовых идентификаторах в кодировку UTF-8 [STD63] (или какое-либо другое надмножество US-ASCII) at на внутреннем интерфейсе, обеспечивая таким способом юолее осмысленные идентификаторы, нежели те, которые даст простое %-представление исходных октетов.

В качестве примера рассмотрим информационную службу, которая предоставляет данные, локально хранящиеся на файловой системе с кодировкой EBCDIC, клиентам через Internet с помощью сервера HTTP. Когда автор создает файл с именем Laguna Beach в файловой системе, в http URI для соответствующего ресурса разумно предположить осмысленную строку Laguna%20Beach. Однако, если сервер создает идентификаторы URI, используя чрезмерно упрощенное отображение октетов, в результате будет получен URI со строкой %D3%81%87%A4%95%81@ %C2%85%81%83%88. Внутренний интерфейс преобразования кодировки исправит эту проблему путем трансляции локального имени в надмножество US-ASCII до создания URI. Естественно, для корректной интерпретации входных URI на таком интерфейсе требуется декодирование октетов с %-представлением (например, %20 преобразовать в SP) до того, как будет проведена обратная смена кодировки для получения локального имени.

В некоторых случаях внутренний интерфейс между компонентой URI и идентификационными данными, которые будут представлять ресурс, значительно менее прямой, чем трансляция кодировки. Например, честь URI может отражать запрос отличных от ASCII данных или числовых координат на карте. Подобно этому, схема URI может определять компоненты с дополнительными требованиями к кодированию, которые применяются до формирования компонент и создания URI.

Когда новая схема URI определяет компоненту, представляющую текстовые данные, которые состоят из символов UCS[1] [UCS], данные следует сначала представить в форме октетов, соответствующих кодировке UTF-8 [STD63] и только тогда не соответствующие незарезервированным символам октеты следует преобразовывать в %-представление. Например, символ A будет представляться, как A, символ LATIN CAPITAL LETTER A WITH GRAVE[2] — как %C3%80, а символ KATAKANA LETTER A[3] — как %E3%82%A2.

3. Синтаксические компоненты

Базовый синтаксис URI включает иерархическую последовательность компонент, называемых схемой (scheme), основанием (authority), путем (path), запросом (query) и фрагментом (fragment).

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
            / path-empty
```

Схема и путь являются обязательными компонентами, хотя путь может быть пустым (нет символов). При наличии компоненты authority путь должен начинаться с символа дробной черты (/) или быть пустым. При отсутствии authority путь не может начинаться с двух символов дробной черты (//). Эти ограничения приводят к 5 разным правилам ABNF для пути (gfhfuhfa 3.3), из которых только одно будет соответствовать любой заданной URI ссылке.

Ниже приведены примеры двух URI с отметкой их компонент.

```
   foo://example.com:8042/over/there?name=ferret#nose
   \_/   \______________/\_________/ \_________/ \__/
    |           |            |            |        |
 схема     основание        путь        запрос  фрагмент
    |   _____________________|__
   / \ /                        \
   urn:example:animal:ferret:nose
```

3.1. Схема

Каждый идентификатор URI начинается с имени схемы, которое указывает спецификацию для выделения идентификаторов в рамках этой схемы. Таким образом, синтаксис URI представляет собой федеративную, расширяемую систему именования, в которой спецификация каждой схемы может вносить свои ограничения в синтаксис и семантику применяемых в этой схеме идентификаторов.

Имя схемы состоит из последовательности символов, начинающейся с буквы, за которой может следовать любая комбинация букв, цифр, а также плюсов (+), точек (.) и дефисов (-). Хотя регистр символов в схемах не принимается во внимание, канонической формой является использование строчных букв (нижний регистр) и в документах со спецификациями схем должны использоваться строчные буквы. Реализациям следует воспринимать прописные буквы в качестве эквивалентов соответствующих строчных (например, HTTP и http) для повышения отказоустойчивости, но для совместимости следует выдавать только строчные буквы.

```
scheme = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
```

Спецификации конкретных схем не включены в этот документ. Процесс регистрации новых схем URI определен в [BCP35]. Реестр схем поддерживает отображение между именами и спецификациями схем. Рекомендации для разработчиков новых схем URI можно найти в [RFC2718]. Спецификация схемы URI должна определять свой собственный синтаксис так, чтобы все соответствующие ему строки соответствовали также грамматике <absolute-URI>, описанной в параграфе 4.3.

---

[1] Universal Character Set — универсальный набор символов.
[2] Символ À. *Прим. перев.*
[3] Японский иероглиф 了. *Прим. перев.*

В случае представления URI с нарушением ограничений для одной или нескольких схем процессам преобразования соответствующих схем следует помечать ссылку, как ошибочную, вместо игнорирования неиспользуемых частей — это позволит снизить число эквивалентных URI и поможет обнаружить неприемлемое использование базового синтаксиса, которое может показывать, что URI был создан для того, чтобы ввести пользователя в заблуждение (параграф 7.6).

3.2. Основание

Многие схемы включают иерархический элемент для «органа именования», которому передаются полномочия по управлению пространством имен, определяемым оставшейся частью URI. Этот «орган» (основание), в свою очередь, может делегировать полномочия кому-либо еще. Базовый синтаксис обеспечивает общие способы различать основания на основе зарегистрированных имен или адресов серверов вкупе с дополнительной информацией о пользователе и порте.

Компоненте authority предшествуют символы //, а завершается она одиночным символом /, знаком вопроса (?), символом номера (#) или просто окончанием URI.

```
authority = [ userinfo "@" ] host [ ":" port ]
```

При создании и нормализации URI следует опускать разграничитель «:», отделяющий хост от порта, если компонента порта пустая. Некоторые схемы не позволяют использовать субкомпоненты порта и/или информации о пользователе.

Если URI содержит компоненту authority, путь должен начинаться с символа / или быть пустым. Анализаторы без проверки (которые просто делят ссылку URI на основные компоненты) зачастую игнорируют структуру субкомпонент основания, трактуя, как цельную строку между все, что находится между символами // и первым завершающим разделителем до тех пор, пока URI не будет разыменован.

3.2.1. Информация о пользователе

Субкомпонента userinfo может состоять из имени пользователя и, опционально, обусловленной схемой информации для получения доступа к ресурсу. При наличии информации о пользователя за ней следует символ @, отделяющий ее от хоста.

```
userinfo = *( unreserved / pct-encoded / sub-delims / ":" )
```

Использование формата user:password в поле userinfo устарело. Приложениям не следует отображать в форме открытого текста какие-либо данные после символа двоеточия (:) в субкомпоненте userinfo, если это не пустая строка (указывающая отсутствие пароля). Приложения могут игнорировать или отвергать такие данные, когда они получены, как часть ссылки, и следует отвергать эти данные, если они не зашифрованы. Передача аутентификационной информации в открытом виде приводит к риску безопасности практически в любом варианте применения.

Приложениям, которые отображают URI для обратной связи с пользователем (например, графические браузеры гипертекста), следует выводить userinfo так, чтобы эту информацию можно было отличить от остальной части URI. Такое отображение поможет пользователю в тех случаях, когда userinfo используется для ввода в заблуждение, будучи представленной подобно доверенному доменному имени (см. параграф 7.6).

3.2.2. Хост

Субкомпонента host в authority идентифицируется IP-литералом в квадратных скобках, адресом IPv4 с разделением полей точками или зарегистрированным именем. Регистр символов в субкомпоненте host не принимается во внимание. Наличие субкомпоненты host в URI не подразумевает, что схема требует доступа к указанному хосту Internet. Во многих случаях host-синтаксис применяется лишь с целью воспользоваться существующим процессом регистрации, реализованным в DNS, которые обеспечивает уникальные в глобальном масштабе имена без издержек на развертывание дополнительного реестра имен. Однако такое использование не достается даром — владелец указанного имени может смениться с течением времени по причинам, не предусмотренным создателем URI. В других случаях данные компоненты host идентифицируют зарегистрированное имя, которое никак не связано с хостом Internet. Термин host используется для правила ABNF потому, что он чаще всего используется для указания хоста, но такое применение не является единственным.

```
host = IP-literal / IPv4address / reg-name
```

Синтаксическое правило для хоста неоднозначно, поскольку оно не полностью различает IPv4address и reg-name. Для устранения этой неоднозначности применяется алгоритм «первого соответствия» (first-match-wins) — если host соответствует правилу для IPv4address, эту субкомпоненту следует считать адресом IPv4, а не зарегистрированным именем (reg-name). Хотя регистр символов в host не принимается во внимание, создателям и нормализаторам следует использовать символы нижнего регистра (строчные буквы) для имен и шестнадцатеричных адресов в целях однородности, а заглавные буквы применять только для %-представления.

Хост, идентифицируемый буквальным адресом IP версии 6 [RFC3513] или выше, выделяется квадратными скобками по обе стороны IP-литерала ([литерал]). Это единственный случай применения квадратных скобок в синтаксисе URI. В предположении будущих (еще не определенных) форматов IP-литералов реализации могут использовать необязательный флаг версии для явной индикации такого формата вместо его эвристической идентификации.

```
IP-literal = "[" ( IPv6address / IPvFuture  ) "]"
IPvFuture  = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
```

Флаг версии не указывает номер версии IP, а показывает будущие версии формата литерала. По этой причине реализациям недопустимо устанавливать флаг версии для имеющихся литеральных адресов IPv4 и IPv6, описанных ниже. Если URI содержит IP-литерал, начинающийся с v (регистр не имеет значения),что показывает наличие флага версии, и разыменовывается приложением, которое не знает значения этого флага версии, этому приложению следует вернуть подходящий сигнал ошибки (address mechanism not supported — неподдерживаемая адресация).

Хост, идентифицируемый литеральным адресом IPv6, указывается в квадратных скобках без предшествующего флага версии. Представленный здесь формат ABNF представляет собой трансляцию текстового определения адреса IPv6 из [RFC3513]. Этот синтаксис не поддерживает идентификаторы областей действия адресов IPv6. 128-битовый адрес IPv6 делится на восемь 16-битовых частей. Каждая из частей представляется числовым 16-ричным значением (без учета регистра букв), содержащим от 1 до 4 символов (нули в начале опускаются). Восемь представленных таким способом частей указываются от старшей к младшей с разделением двоеточиями. Две младшие части могут быть представлены

в текстовом формате IPv4. Одна или несколько последовательных 16-битовых частей со значением могут быть опушены (пусты) с сохранением лишь двоеточий в качестве разделеителей опущенных частей.

```
IPv6address =                            6( h16 ":" ) ls32
            /                         "::" 5( h16 ":" ) ls32
            / [               h16 ] "::" 4( h16 ":" ) ls32
            / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
            / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
            / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
            / [ *4( h16 ":" ) h16 ] "::"              ls32
            / [ *5( h16 ":" ) h16 ] "::"              h16
            / [ *6( h16 ":" ) h16 ] "::"

ls32        = ( h16 ":" h16 ) / IPv4address
                              ; младшие 32 бита адреса

h16         = 1*4HEXDIG
            ; 16 битов адреса представлены в 16-ричном формате
```

Хост, идентифицируемый литеральным адресом IPv4, представляется в нотации с разделением точками (последовательность из 4 десятичных значений от 0 до 255, разделенных точками), как описано в [RFC1123] со ссылкой на [RFC0952]. Отметим, что на некоторых платформах могут использоваться иные варианты обозначения, как описано в параграфе 7.4, но представленная выше форма является единственно разрешенной данной грамматикой.

```
IPv4address = dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet   = DIGIT                 ; 0-9
            / %x31-39 DIGIT         ; 10-99
            / "1" 2DIGIT            ; 100-199
            / "2" %x30-34 DIGIT     ; 200-249
            / "25" %x30-35          ; 250-255
```

Хост, идентифицируемый зарегистрированным именем, указывается последовательностью символов, обычно предназначенной для поиска в локальном списке или реестре имен, хотя специфическая семантика схемы URI может требовать использования конкретного реестра (или таблицы с фиксированным именем). Наиболее распространен механизм поиска имен в DNS[1]. Зарегистрированные имена, предназначенные для поиска в DNS, используют синтаксис, определенный в параграфе 3.5 [RFC1034] и параграфе 2.1 [RFC1123]. Такие имена состоят из последовательности меток, разделенных точками, и каждая метка начинается и заканчивается буквой или цифров и может включать символы «-». За самой правой меткой полного доменного имени в DNS может следовать одна точка и ее следует включать, если требуется различать полные доменные имена от локальных (внутри некого локального домена).

```
reg-name = *( unreserved / pct-encoded / "-" / ".")[2]
```

Если схема URI предполагает принятую по умолчанию субкомпоненту host, она используется в случае отсутствия субкомпоненты host или при указании пустого имени (нулевой размер). Например схема file для URI определена так, что отсутствие субкомпоненты authority, пустая субкомпонента host и значение localhost указывают на машину локального пользователя, а в схеме http отсутствие authority или пустое значение host не приемлемы.

Данная спецификация не задает конкретной технологии поиска зарегистрированного имени и, следовательно, не ограничивает синтаксис reg-name сверх необходимого для интероперабельности. Решение вопроса проверки соответствия синтаксиса зарегистрированного имени передается операционной системе каждого приложения, выполняющего преобразование URI, и эта операционная система решает, что она будет разрешать для идентификации хоста. Реализация преобразования URI может использовать DNS, таблицы хостов, «желтые страницы», NetInfo, WINS или любую другую систему поиска зарегистрированных имен. Однако для URI с глобальной значимостью требуются глобальные способы именования типа полных доменных имен DNS. Создателям URI следует использовать имена, соответствующие синтаксису DNS, даже в случаях, когда использование DNS не представляется очевидным. Следует также ограничивать размер имен 255 символами.

Синтаксис reg-name разрешает использовать октеты с %-кодированием для того, чтобы представлять зарегистрированные имена с отличными от ASCII символами независимым от технологии преобразования имен способом. Не входящие в ASCII символы должны сначала представляться в кодировке UTF-8 [STD63], а после этого для каждого октета последовательности UTF-8 должно использоваться %-кодирование для представления в качестве символа URI. Приложениям, создающим URI, недопустимо использовать %-кодирование в субкомпоненте host, если оно служит не для последовательности символов UTF-8. Когда зарегистрированное имя с отличными от ASCII символами представляет доменное имя на другом (не английском) языке, предназначенное для преобразования через DNS, имя должно трансформироваться в представление IDNA [RFC3490] до операции поиска. Создателям URI следует представлять такие имена в кодировке IDNA, а не с помощью %-кодирования, если они хотят обеспечить интероперабельность с унаследованными преобразователями URI.

3.2.3. Порт

Субкомпонента port в authority является необязательным номером порта в десятичном формате, следующим за субкомпонентой host и отделенным от нее двоеточием.

```
port = *DIGIT
```

Схема может определять используемый по умолчанию порт. Например, схема http по умолчанию предполагает порт 80, соответствующий зарезервированному порт TCP. Тип порта, обозначенного номером (например, TCP, UDP, SCTP), определяется схемой URI. Создателям и нормализаторам URI следует опускать компоненту port и разграничитель «:», если номер порта не указан или совпадает с используемым в схеме по умолчанию.

---

[1] Domain Name System — система доменных имен.

[2] В оригинале была допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=4942. *Прим. перев.*

3.3. Путь

Компонента path содержит данные (обычно, организованные иерархически), которые вместе с неиерархическими данными компоненты query (параграф 3.4) служат для идентификации ресурса в области действия схемы URI и полномочий именования (если они есть). Путь завершается первым символом «?», «#» или просто концом URI.

Если URI содержит компоненту authority, путь может быть пустым или начинаться с символа «/». Если URI не включает authority, путь не может начинаться с символов «//». Кроме того, ссылка URI (параграф 4.1) может указывать относительный путь и в этом случае первый сегмент пути не может включать символ «:». ABNF задает пять раздельных правил для устранения неоднозначности пути, из которых только одно будет соответствовать подстроке пути в данной ссылке URI. Для обозначения подстрок URI при разборе правил используется базовый термин «компонента пути» (path component).

```
path          = path-abempty    ; начинается с / или пуст
              / path-absolute   ; начинается с /, но не с //
              / path-noscheme   ; начинается с сегмента без двоеточия
              / path-rootless   ; начинается с сегмента
              / path-empty      ; 0 символов

path-abempty  = *( "/" segment )
path-absolute = "/" [ segment-nz *( "/" segment ) ]
path-noscheme = segment-nz-nc *( "/" segment )
path-rootless = segment-nz *( "/" segment )
path-empty    = ""[1]

segment       = *pchar
segment-nz    = 1*pchar
segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
              ; сегмент ненулевого размера без двоеточий (:)

pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"
```

Путь состоит из последовательности сегментов (path segment), разделенных символами «/». Путь всегда определяется для URI, хотя он может быть пустым (нулевого размера). Использование символ дробной черты для индикации иерархии требуется лишь в тех случаях, кода URI будет применяться в качестве контекста для относительных ссылок. Например URI <mailto:fred@example.com> имеет компоненту пути fred@example.com, а <foo://info.example.com?fred> имеет пустой путь.

Сегменты пути «.» и «..», называемые также сегментами из точек (dot-segment), определяются для относительных ссылок в иерархии имен. Они предназначены для использования в начале пути относительных ссылок (параграф 4.2) для индикации положения в иерархическом дереве имен. Это похоже на роль этих символов в структуре каталогов некоторых операционных систем, где одна точка указывает текущий каталог, а две — родительский. Однако, в отличие от файловых систем сегменты с точками интерпретируются только внутри иерархии пути URI и удаляются в процессе преобразования (параграф 5.2).

За исключением сегментов с точками сегменты пути с точки зрения базового синтаксиса являются «непрозрачными». Создающие URI приложения зачастую применяют зарезервированные символы, дозволенные в сегментах, для разграничения специфических для схемы или обработчика субкомпонент. Например, зарезервированные символы «;» и «=» часто служат для разделения параметров и их значений в данном сегменте. Зарезервированный символ запятой «,» часто применяется с аналогичными целями. Например, создатель URI может использовать сегмент вида «name;v=1.1» для индикации ссылки на версию 1.1 для «name», а другой может использовать для этих же целей сегмент вида «name,1.1». Типы параметров могут определяться семантикой конкретной схемы, но в большинстве случаев синтаксис параметра зависит от конкретной реализации алгоритма разыменования URI.

3.4. Запрос

Компонента query содержит неиерархические данные, которые, наряду с данными компоненты path (параграф 3.3), служат для идентификации ресурса в зоне действия схемы URI и полномочий именования (если они есть). Компонента запроса указывается первым символом «?» и завершается символом «#» или концом URI.

```
query       = *( pchar / "/" / "?" )
```

Символы «/» и «?» могут представлять данные в компоненте query. Следует помнить, что некоторые устаревшие реализации с ошибками могут некорректно обрабатывать такие данные при их использовании в качестве базовых URI для относительных ссылок (параграф 5.1), видимо по причине неспособности отделить путь от запроса при просмотре разделителей иерархии. Однако, поскольку компоненты query часто применяются для передачи идентифицирующих данных в форме gfh «ключ=значение» (key=value) и зачастую значение является ссылкой на другой URI, для большей применимости иногда лучше избегать %-представления этих символов.

3.5. Фрагмент

Компонента идентификатора фрагмента URI позволяет опосредованно указывать вторичный ресурс путем ссылки на основной и указания дополнительных идентификационных данных. Идентифицируемым вторичным ресурсом может быть некая часть или подмножество основного ресурса или некий другой ресурс, описанный данным представлением. Компонента идентификатора фрагмент указывается символом «#» и завершается в конце URI.

```
fragment = *( pchar / "/" / "?" )
```

Смысл идентификатора фрагмента определяется множеством представлений, которое может быть результатом операций поиска на первичном ресурсе. Формат и преобразование фрагмента зависят, следовательно, от типа среду [RFC2046] потенциально найденного представления, хотя само этот поиск выполняется только в случае разыменования URI. Если такого представления не существует, семантика фрагмента считается неизвестной и,

[1] В оригинале была допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=2033. *Прим. перев.*

фактически, неограниченной. Семантика идентификатора фрагмента не зависит от схемы URI и, в силу этого, не может быть переопределена в спецификации схемы.

Отдельные типы сред могут задавать свои ограничения и структуры в рамках синтаксиса идентификатора фрагмента для задания различных типов подмножеств, представления или внешних ссылок, которые могут быть идентифицированы этим типом среду в качестве вторичных ресурсов. Если основной ресурс имеет множество представлений, как часто бывает с ресурсами, чье представление выбирается на основе атрибутов поискового запроса (иными словами, согласования содержимого), тогда все идентифицируемое фрагментом должно быть согласовано в рамках всех таких представлений. Каждому представлению следует определять фрагмент так, чтобы он соответствовал одному и тому же вторичному ресурсу, независимо от способа представления, или просто оставлять фрагмент не определенным (например, не найден).

Как и для всех URI, использование компоненты идентификатора фрагмента не подразумевает реальных поисковых действий. URI с идентификатором фрагмента может применяться для ссылки на вторичный ресурс без каких-либо предположений о доступности или фактах доступа к основному ресурсу.

Идентификаторы фрагментов играют важную роль в информационно-поисковых системах в качестве основного варианта косвенных ссылок на клиентской стороне, что позволяет автору конкретно указать аспекты существующего ресурса, предоставляемые его владельцем лишь опосредованно. Таким образом, идентификаторы фрагментов не используются в специфической для схем обработке URI и отделяются от остальной части URI до разыменования, а идентификационные данные в самом фрагменте разыменовываются только пользовательским агентом, независимо от схемы URI. Хотя такая раздельная обработка зачастую воспринимается как потеря информации, особенно в случаях перенаправления ссылок когда ресурс со временем перемещается, это также позволяет предотвратить возможность запрета авторам указывать селективные ссылки внутри ресурса со стороны держателей информации. Косвенные ссылки также обеспечивают дополнительную гибкость и расширяемость для систем, использующих URI, поскольку определить и развернуть новый тип среды проще, чем новую схему идентификации.

Символы «/» и «?» разрешены для представления данных в идентификаторах фрагментов. Однако следует помнить, что некоторые устаревшие реализации с ошибками могут некорректно обрабатывать такие данные при их использовании в качестве базовых URI для относительных ссылок (параграф 5.1).

4. Применение

Когда приложение указывает ссылку на URI, это не всегда является полной формой URI, определенной правилами синтаксиса. В целях экономии места и использования преимуществ иерархической локализации многие элементы протоколов Internet и форматов типа среды допускают сокращение URI, тогда как другие ограничивают синтаксис конкретной формой URI. В этой спецификации определены наиболее общие формы синтаксиса ссылок, поскольку они оказывают влияние на базовый синтаксис и сами зависят от него, требуя однотипного разбора для обеспечения согласованности интерпретаций.

4.1. Ссылка URI

Ссылки URI (URI-reference) служат для обозначения наиболее распространенного применения идентификаторов ресурсов.

```
URI-reference = URI / relative-ref
```

URI-reference представляет собой URI или относительную ссылку. Если префикс ссылки URI не соответствует синтаксису с указанием схемы, отделенной двоеточием, это говорит о том, что URI-reference является относительной ссылкой.

Ссылка URI обычно сначала разбирается на 5 компонент URI в соответствии с порядком их указания и типом (относительная или абсолютная) ссылки. После этого каждая компонента разбирается на соответствующие части и выполняется их проверка. ABNF для ссылок URI вместе с правилом «первого соответствия» (first-match-wins) вполне достаточно для проверяющего базовый синтаксис анализатора. Читателям, знакомым с регулярными выражениями, следует обратиться к Приложению B, где приведен пример разбора URI-reference без проверки корректности, который будет принимать произвольную строку и выделять из нее компоненты URI.

4.2. Относительная ссылка

Относительные ссылки используют преимущества иерархического синтаксиса (параграф 1.2.3) для выражения ссылок URI относительно пространства имен другого иерархического URI.

```
relative-ref  = relative-part [ "?" query ] [ "#" fragment ]
relative-part = "//" authority path-abempty
                  / path-absolute
                  / path-noscheme
                  / path-empty
```

Идентификатор URI, указываемый относительной ссылкой, который называют также целевым URI, получается с помощью алгоритма преобразования ссылок, описанного в разделе 5.

Относительные ссылки, начинающиеся с символов //, называют сетевыми путями и используются такие ссылки достаточно редко. Относительные ссылки, начинающиеся с одного символа /, называют абсолютными путями. Относительные ссылки без символа дробной черты в начале называют относительными путями (relative-path reference).

Сегменты пути с символом двоеточия (например, this:that) не могут использоваться в качестве первого сегмента относительного пути, поскольку будет возникать путаница с именами схем. Перед такими сегментами должен присутствовать сегмент с точкой (например, «./this:that») для указания ссылки по относительному пути.

4.3. Абсолютные URI

Некоторые протокольные элементы разрешают только абсолютную форму URI без идентификаторов фрагментов. Например, в базовых URI для использования с относительными ссылками синтаксическое правило absolute-URI не разрешает фрагменты.

```
absolute-URI  = scheme ":" hier-part [ "?" query ]
```

Спецификации схем URI должны определять свой синтаксис так, чтобы все строки, соответствующие этому синтаксису, соответствовали и грамматике <absolute-URI>. Спецификации схем не будут определять синтаксис идентификаторов фрагментов независимо от их применимости к определяемым этой схемой ресурсам, поскольку идентификация фрагментов ортогональна определению схемы. Однако в спецификации схем рекомендуется включать широкий набор примеров, показывающих использование URI этой схемы с идентификаторами фрагментов, когда это применимо.

4.4. Ссылка на тот же документ

Когда ссылка URI указывает на URI, который за исключением его фрагментов (если они есть) идентичен базовому URI (параграф 5.1), это называется ссылкой на тот же документ. Наиболее частым примером таких ссылок являются относительные ссылки, которые пусты или включают лишь символ #, за которым следует идентификатор фрагмента.

При разыменовании ссылки на тот же документ с целью получения документа субъект этой ссылки размещается в том же самом элементе (презентации, документе, сообщении) и, следовательно, после разыменование ссылки не требуется новых поисковых действий.

Нормализация базовых и целевых URI до их сравнения (как описано в параграфах 6.2.2 и 6.2.3) разрешена, но редко выполняется на практике. Нормализация может повышать число ссылок на тот же документ, что позволяет получить преимущества для некоторых систем кэширования. По этой причине авторам ссылок не следует полагать, что слабо различающиеся, но эквивалентные ссылки URI будут (или не будут) интерпретироваться любым приложением как ссылки на тот же документ.

4.5. Ссылки по суффиксам

Синтаксис URI разработан для однозначного указания на ресурсы с возможностью расширения за счет схем URI. Однако использование URI для идентификации стало повсеместным и традиционные средства массовой информации (телевидение, радио, газеты, рекламные щиты и т. п.) все шире применяют в качестве ссылок суффиксы URI, содержащие только компоненты authority и path типа

```
www.w3.org/Addressing/
```

или просто DNS-имя, которое им принадлежит. Такие ссылки предназначены для непосредственного восприятия человек без участия машин и предполагают, что контекстной эвристики достаточно для дополнения URI (например, большинство зарегистрированных имен, начинающихся с www, очевидно имеют в URI префикс http://). Хотя не существует стандартного набора эвристических методов устранения неоднозначности суффиксов URI, многие реализации клиентов позволяют пользователю вводить такие ссылки и эвристическими методами преобразуют их.

И хотя использование ссылок по суффиксам стало повсеместным, его следует по возможности избегать и никогда не следует применять в ситуациях, где предполагается долгосрочная ссылка. Упомянутая выше эвристика может меняться с течением времени (в частности, при распространении новых схем URI) и зачастую будет неприменимой в ином контексте. Кроме того, это может вызывать проблемы безопасности, аналогичные описанным в [RFC1535].

Поскольку суффиксы URI используют такой же синтаксис, как ссылки с относительными путями, их нельзя использовать в контексте, где предполагается наличие относительных ссылок. Поэтому ссылки по суффиксам ограничены ситуациями, где не определено базовых URI, типа диалоговых окон или анонсов off-line.

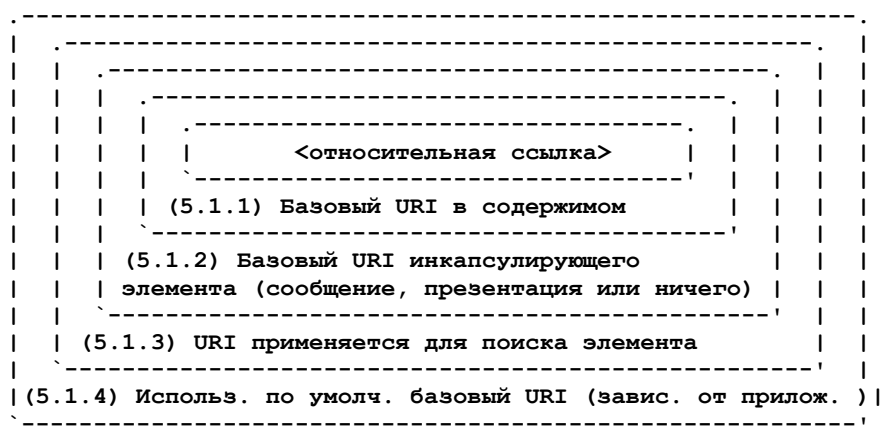
5. Преобразование ссылки

В этом разделе описан процесс преобразования ссылок URI в контексте, позволяющем использовать относительные ссылки так, что в результате получается строка, соответствующая синтаксическим правилам <URI> из раздела 3.

5.1. Организация базового URI

Термин «относительный» предполагает наличие «базового URI», от которого «начинается» относительная ссылка. Как и ссылки, включающие только фрагменты (параграф 4.4), относительные ссылки применимы только при известном базовом URI. Базовый URI нужно организовать до разбора ссылки URI, которая может быть относительной. Для базового URI требуется соответствие синтаксическому правилу <absolute-URI> (параграф 4.3). Если базовый URI создается из ссылки URI, эта ссылка должна быть преобразована в абсолютную форму с исключением всех фрагментов до использования в качестве базового URI.

Базовый URI для ссылки можно организовать одним из четырех способов, описанных ниже в порядке предпочтения. Порядок предпочтения можно рассматривать в терминах уровней, где самый внутренний вариант базового URI является наиболее предпочтительным (см. рисунок ниже).



5.1.1. Базовые URI внутри содержимого

В некоторых типах сред базовые URI для относительных ссылок могут встраиваться непосредственно в содержимое так, что они могут быть легко получены анализатором. Это может быть полезно для описательных документов типа оглавлений, которые могут передаваться другим с использованием протоколов, отличающихся от применяемых в обычном контексте поиска (например, по электронной почте или в новостях USENET).

Методы встраивания базовых URI для каждого типа среды выходят за рамки этой спецификации. В тех случаях, когда это возможно, здесь описывается подходящий синтаксис путем указания формата данных, связанного с типом среды.

5.1.2. Базовый URI из инкапсулирующего элемента

Если нет встроенного базового URI, он определяется контекстом поиска представления. Для вложенных документов (например, в сообщении или архиве), таким контекстом служит инкапсулирующий элемент. Таким образом, по умолчанию базовым URI для представления служит базовый URI инкапсулирующего элемента.

Механизм встраивания базовых URI в контейнеры MIME (например, сообщение или элемент со множеством частей) определен MHTML [RFC2557]. Протоколы, которые не используют синтаксис заголовков MIME, но позволяют включать в сообщения те или иные метаданные с тегами, могут задавать свой синтаксис для определения базового URI.

5.1.3. Базовый URI из поискового URI

Если нет вложенного базового URI и представление не инкапсулировано в некий другой объект, тогда, если URI был использован для получения представления, этот идентификатор URI следует считать базовым URI. Отметим, что если поиск представления привел к перенаправлению, в качестве базового следует использовать последний URI (по которому было получено реальное представление).

5.1.4. Используемый по умолчанию базовый URI

Если не применимо ни одно из описанных выше условий, базовый идентификатор URI определяется контекстом приложения. Поскольку такое определение очевидным образом зависит от приложения, невозможность использовать какой-либо из описанных выше методов может приводить к разной интерпретации одного и того же содержимого в разнотипных приложениях.

Отправитель представления с относительными ссылками отвечает за возможность организации базовых URI для таких ссылок. Как и ссылки, содержащие только идентификаторы фрагментов, относительные ссылки можно надежно применять лишь в ситуациях, где возможно определить базовый URI.

5.2. Преобразование относительных ссылок

В этом разделе описан алгоритм преобразования ссылки URI, которая может быть задана относительно данного базового URI, в компоненты цели. Эти компоненты потом можно собрать в целевой URI, как описано в параграфе 5.3. Этот алгоритм обеспечивает детерминированные результаты и может применяться для тестирования вывода других реализаций. Приложения могут использовать свои алгоритмы преобразования относительных ссылок, если они дают на выходе результат, совпадающих с выходом приведенного здесь алгоритма.

5.2.1. Предварительный разбор базового URI

Базовый организуется URI (Base) в соответствии с описанной в параграфе 5.1 процедурой и разбирается на пять основных компонент, описанных в разделе 3. Отметим, что обязательной для базового URI является лишь компонента scheme, а остальные компоненты могут быть пустыми или не определенными. Компонента считается не определенной, если соответствующий разграничитель отсутствует в ссылке URI. Компонента path определена всегда, но может быть пустой.

Описанная в параграфах 6.2.2 и 6.2.3 нормализация базового URI является необязательной. Ссылка URI должна быть преобразована в целевой URI до выполнения нормализации.

5.2.2. Преобразование ссылок

Для каждой ссылки URI (R) приведенный ниже псевдокод описывает преобразование R в целевой URI (T).

```
-- Ссылка URI разбирается на пять компонент URI
-- (R.scheme, R.authority, R.path, R.query, R.fragment) = parse(R);

-- Нестрогий анализатор может игнорировать схему в ссылке,
-- если она идентична схеме в базовом URI.
--
if ((not strict) and (R.scheme == Base.scheme)) then
   undefine(R.scheme);
endif;

if defined(R.scheme) then
   T.scheme    = R.scheme;
   T.authority = R.authority;
   T.path      = remove_dot_segments(R.path);
   T.query     = R.query;
else
   if defined(R.authority) then
      T.authority = R.authority;
      T.path      = remove_dot_segments(R.path);
      T.query     = R.query;
   else
      if (R.path == "") then
```

```
            T.path = Base.path;
            if defined(R.query) then
               T.query = R.query;
            else
               T.query = Base.query;
            endif;
         else
            if (R.path starts-with "/") then
               T.path = remove_dot_segments(R.path);
            else
               T.path = merge(Base, R);[1]
               T.path = remove_dot_segments(T.path);
            endif;
            T.query = R.query;
         endif;
         T.authority = Base.authority;
      endif;
      T.scheme = Base.scheme;
   endif;

   T.fragment = R.fragment;
```

5.2.3. Слияние путей

Приведенный выше псевдокод иллюстрирует процедуру «слияния» ссылки по относительному пути с путем из базового URI. Это достигается следующим образом:

- если базовый URI имеет определенную компоненту authority и пустой путь или путь в базовом URI заканчивается символами «/..», возвращается строка, состоящая из базового пути, дополненного символом «/», которая затем объединяется с путем в ссылке,[2]
- в остальных случаях возвращается строка, состоящая компоненты path в ссылке, добавленной в конец пути базового URI, из которого исключен последний сегмент (т. е., все символы после самого правого символа «/» или весь путь базового URI, если в нем нет символов «/»).

5.2.4. Удаление сегментов с точками

Псевдокод также включает процедуру «удаления сегментов с точками» для интерпретации и удаления сегментов «.» и «..» из пути в ссылке. Это делается после извлечения пути из ссылки (независимо от того, является ли путь относительным) для удаления всех неприемлемых или избыточных сегментов с точками до формирования целевого URI. Этот процесс можно реализовать множеством способов и ниже описан простой путь с двумя буферами для строк.

1. Входной буфер инициализируется добавленными сейчас компонентами пути, а выходной — пустой строкой.
2. Пока входной буфер не пуст, выполняется приведенный ниже цикл:
   A. если входной буфер начинается с префикса «../» или «./», этот префикс удаляется из буфера;
   B. если входной буфер начинается с префикса «/./» или «/.», где «.» представляет собой полный сегмент пути, этот префикс заменяется во входном буфере префиксом «/»;
   C. если входной буфер начинается с префикса «/../» или «/..», где «..» представляет собой полный сегмент пути, этот префикс заменяется во входном буфере префиксом «/», а из выходного буфера удаляется последний сегмент с предшествующим ему символом «/»;
   D. если входной буфер содержит только «.» или «..», он опустошается;
   E. первый сегмент пути из входного буфера переносится в конец выходного буфера с включением начального символа «/» (если он есть) и всеми последующими символами вплоть до следующего символа «/» (не включая его) или до конца входного буфера.
3. Содержимое выходного буфера возвращается в качестве результата remove_dot_segments.

Отметим, что сегменты с точками предназначены для использования в ссылках URI, где нужно указать идентификатор относительно иерархии имен в базовом URI. Алгоритм remove_dot_segments учитывает эту иерархию, удаляя лишние сегменты с точками вместо их трактовки как ошибок или сохранения, ведущего к ошибочной интерпретации реализациями разыменования.

Ниже показано по шагам применение описанного алгоритма для двух примеров слияния путей.

Этап	*Выходной буфер*	*Входной буфер*
1		`/a/b/c/./../../g`
2E	`/a`	`/b/c/./../../g`
2E	`/a/b`	`/c/./../../g`
2E	`/a/b/c`	`/./../../g`
2B	`/a/b/c`	`/../../g`
2C	`/a/b`	`/../g`

[1] В оригинале была допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=2933. *Прим. перев.*
[2] В оригинале была допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=4789. *Прим. перев.*

2C	/a	/g
2E	/a/g	

Этап	*Выходной буфер*	*Входной буфер*
1		mid/content=5/../6
2E	mid	/content=5/../6
2E	mid/content=5	/../6
2C	mid	/6
2E	mid/6	

Некоторые приложения могут счесть более эффективной реализацию алгоритма remove_dot_segments с использованием двухсегментного стека вместо строк.

Примечание. Помните, что некоторые устаревшие реализации с ошибками могут сталкиваться с отказами при попытке выделить компоненту ссылки query из компоненты path для слияния базового и ссылочного путей, что приводит к проблемам совместимости для запросов с компонентой query, содержащей «/../» или «/./».

5.3. Сборка компонент

Разобранные компоненты URI могут быть скомпонованы заново для получения соответствующей строки ссылки URI. Ниже приведен псевдокод для такой сборки.

```
result = ""

if defined(scheme) then
   добавить scheme в конце result;
   добавить ":" в конце result;
endif;

if defined(authority) then
   добавить "//" в конце result;
   добавить authority в конце result;
endif;

добавить path в конце result;

if defined(query) then
   добавить "?" в конце result;
   добавить query в конце result;
endif;

if defined(fragment) then
   добавить "#" в конце result;
   добавить fragment в конце result;
endif;

вернуть result;
```

Следует отметить, что мы стараемся сохранить различие между компонентами, которые не определены (это означает, что разделитель для компоненты не присутствует в ссылке), и пустыми компонентами (сепаратор для компоненты присутствует, но сразу за ним следует сепаратор другой компоненты или ссылка заканчивается).

5.4. Пример преобразования ссылки

В представлении с четко определенным базовым URI

```
http://a/b/c/d;p?q
```

относительная преобразуется в целевой идентификатор URI, как показано в двух следующих параграфах.

5.4.1. Нормальные примеры

```
"g:h"           =  "g:h"
"g"             =  "http://a/b/c/g"
"./g"           =  "http://a/b/c/g"
"g/"            =  "http://a/b/c/g/"
"/g"            =  "http://a/g"
"//g"           =  "http://g"
"?y"            =  "http://a/b/c/d;p?y"
"g?y"           =  "http://a/b/c/g?y"
"#s"            =  "http://a/b/c/d;p?q#s"
"g#s"           =  "http://a/b/c/g#s"
"g?y#s"         =  "http://a/b/c/g?y#s"
";x"            =  "http://a/b/c/;x"
"g;x"           =  "http://a/b/c/g;x"
"g;x?y#s"       =  "http://a/b/c/g;x?y#s"
""              =  "http://a/b/c/d;p?q"
```

```
      "."             =  "http://a/b/c/"
      "./"            =  "http://a/b/c/"
      ".."            =  "http://a/b/"
      "../"           =  "http://a/b/"
      "../g"          =  "http://a/b/g"
      "../.."         =  "http://a/"
      "../../"        =  "http://a/"
      "../../g"       =  "http://a/g"
```

5.4.2. Аномальные примеры

Хотя появление приведенных ниже аномальных случаев на практике не очевидно, всем анализаторам URI следует быть готовыми к таким аномалиям. Во всех приведенных ниже примерах используется та же база, что и раньше.

Анализаторы должны с осторожностью обрабатывать идентификаторы с числом сегментов «..» в ссылке с относительным путем, превышающим число уровней иерархии в пути базового URI. Отметим, что синтаксис «..» не может применяться для изменения компоненты authority в URI.

```
      "../../../g"    =  "http://a/../g"
      "../../../../g" =  "http://a/../../g"[1]
```

Анализаторы должны также удалять сегменты с точками «.» и «..», когда они являются полными компонентами пути, но не частью сегмента.

```
      "/./g"          =  "http://a/g"
      "/../g"         =  "http://a/g"
      "g."            =  "http://a/b/c/g."
      ".g"            =  "http://a/b/c/.g"
      "g.."           =  "http://a/b/c/g.."
      "..g"           =  "http://a/b/c/..g"
```

Менее очевидны слкчаи, когда в относительной ссылке используются ненужные или бессмысленные полные сегменты пути «.» и «..».

```
      "./../g"        =  "http://a/b/g"
      "./g/."         =  "http://a/b/c/g/"
      "g/./h"         =  "http://a/b/c/g/h"
      "g/../h"        =  "http://a/b/c/h"
      "g;x=1/./y"     =  "http://a/b/c/g;x=1/y"
      "g;x=1/../y"    =  "http://a/b/c/y"
```

Некоторые приложения сталкиваются с отказом при попытке отделить в ссылке запрос и/или фрагмент от компоненты path до слияния с базовым путем и удаления сегментов с точками. Эта ошибка встречается достаточно редко, поскольку типичное использование фрагментов никогда не включает символ иерархии (/), а компонента query обычно не применяется в относительных ссылках.

```
      "g?y/./x"       =  "http://a/b/c/g?y/./x"
      "g?y/../x"      =  "http://a/b/c/g?y/../x"
      "g#s/./x"       =  "http://a/b/c/g#s/./x"
      "g#s/../x"      =  "http://a/b/c/g#s/../x"
```

Некоторые анализаторы позволяют включать имя схему в относительные ссылки, Если оно совпадает с именем схемы в базовом URI. Это было признано «лезейкой» в предшествующих спецификациях частичных URI [RFC1630]. Следует избегать этого, но поддержка допускается в целях совместимости со старыми версиями.

```
      "http:g"        =  "http:g"         ; для строгого разбора
                      /  "http://a/b/c/g" ; для совместимости с ранними версиями
```

6. Нормализация и сравнение

Одной из наиболее распространенных операций с URI является простое сравнение — проверка эквивалентности двух URI без их использования для доступа к соответствующим ресурсам. Сравнение выполняется всякий раз при обращении к кэшу откликов, проверке браузером своей истории для подсвечивания ссылок или обработке анализатором XML тегов в пространстве имен. Поисковые и индексирующие машины зачастую используют расширенную нормализацию для сужения пространства поиска и снижения числа дублирующих операций при запросах и пространства для хранения откликов.

Сравнение URI выполняется с той или иной конкретной целью. Протоколы или реализации, сравнивающие URI с разными целями, зачастую по разному организованы для достижения компромисса в части прилагаемых для снижения числа идентификаторов-псевдонимов усилий. В этом разделе описаны разные методы, которые могут применяться для сравнения URI, компромиссы при выборе метода сравнения, а также приложения, использующие эти методы.

6.1. Эквивалентность

Поскольку URI служат для идентификации ресурсов, по-видимому два идентификатора следует считать эквивалентными, если они указывают на один и тот же ресурс. Однако такое определение эквивалентности не совсем практично, поскольку реализация не может сравнить два ресурса, пока у нее нет полной информации или контроля над ними. По этой причине определение эквивалентности разных URI основано на сравнении строк, возможно с добавлением неких правил, обеспечиваемых определением схемы URI. Для обозначения результата такого сравнения используются термины «разные» (different) и «эквивалентные» (equivalent), но может существовать множество зависимых от приложений вариантов эквивалентности.

Можно определить эквивалентность двух URI, однако сравнения строк не достаточно для того, чтобы утверждать, что два URI указывают на разные ресурсы. Например, владелец двух различающихся доменных имен может отнести тот или иной ресурс к обоим доменами, что приведет к наличию двух разных URI для одного ресурса. Следовательно,

[1] В оригинале была допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=4547. *Прим. перев.*

методы сравнения разрабатываются с учетом минимизации ложных различий при полном предотвращении ложных совпадений.

При проверке эквивалентности приложениям не следует напрямую сравнивать относительные ссылки, перед сравнением их следует преобразовать в целевые URI. При сравнении URI для выбора ил предотвращения сетевого действия (типа поиска представления), фрагменты следует исключать из сравнения.

6.2. Порядок сравнения

На практике для проверки эквивалентности URI применяется множество методов. Эти методы ранжируются по трудоемкости обработки и возможности разных расхождений. Как отмечено выше, полностью избежать ложных различий не удается, но снижение их числа значительно повышает трудоемкость и эффективно не для всех приложений.

Если представить ранжирование методов в форме лестницы, последующее обсуждение будет подъемом, начинающимся с простейших методов, дающих сравнительно высокую вероятность ложных различий в направлении методов с большими вычислительными издержками и меньшим риском получить ложные различия.

6.2.1. Простое сравнение строк

Если текстовые строки двух сравниваемых URI идентичны, можно безопасно считать эти URI эквивалентными. Проверка этого типа вносит очень незначительные вычислительные издержки и широко применяется в разных типах приложений (особенно при разборе).

При проверке эквивалентности строк требуются некоторые меры предосторожности. Эту процедуру часто называют побитовым (bit-for-bit) или побайтовым (byte-for-byte) сравнением, что может вводить в заблуждение. Проверка строк на совпадение обычно выполняется в виде попарного сравнеия символов разных строк, начиная с первого и до конца строки или обнаружения различия.

Для такого сравнения символов требуется, чтобы каждая пара символов была представлена в сравнимой форме. Например, если один URI хранится в виде массива байтов с кодировкой EBCDIC, а другой в объекте Java String (UTF-16), побитовое сравнение даст расхождение сразу же. Лучше говорить о посимвольной эквивалентности, а не побайтовой или побитовой. С практической точки зрения посимвольное сравнение следует выполнять путем сравнения кодов символов после перевода обеих строк в одну кодировку.

Ложные различия связаны с созданием и использованием псевдонимов URI. Ненужные псевдонимы можно частично устранить независимо от используемого метода сравнения путем представления ссылок URI в нормализованной форме (т. е., идентичной той форме, которая будет получаться на выходе описанной ниже нормализации).

Протоколы и форматы данных часто ограничиваются в сравнении URI простым сравнением символьных строк, исходя из допущения о том, что люди и программные реализации заинтересованы в том, чтобы предоставлять согласованные ссылки URI или, по крайней мере, делать их достаточно согласованными для того, чтобы нормализация ничего не давала в смысле повышения эффективности.

6.2.2. Нормализация на основе синтаксиса

Для снижения вероятности ложных расхождений реализации могут использовать логику, представленную в этой спецификации. Эта обработка требует несколько больших издержек по сравнению с посимвольным сравнением строк. Например, приложение, использующее такой подход, может считать приведенные ниже два URI эквивалентными.

```
example://a/b/c/%7Bfoo%7D
eXAMPLE://a/./b/../b/%63/%7bfoo%7d
```

Пользовательские агенты Web (например, браузеры) обычно используют этот тип нормализации URI при определении доступности кэшированного отклика. Основанная на синтаксисе нормализация включает такие методы, как нормализация регистра символов и %-представления, а также удаление сегментов с точками.

6.2.2.1. Нормализация регистра

Для всех URI для шестнадцатеричных цифр в триплетах %-кодирования не принимается во внимание (например, %3a и %3A), поэтому их следует нормализовать путем преобразования в верхний регистр символов A-F.

При использовании в URI компонент базового синтаксиса всегда применяются правила эквивалентности синтаксических компонент — в именах схем и хостов регистр символов не имеет значения, поэтому их следует нормализовать путем перевода в нижний регистр. Например, URI <HTTP://www.EXAMPLE.com/> является эквивалентом <http://www.example.com/>. Для других компонент базового синтаксиса считается, что регистр символов имеет значение, если в определении схемы явно не указано иное (см. параграф 6.2.3).

6.2.2.2. Нормализация %-кодирования

Механизм %-кодирования (параграф 2.1) часто служит причиной того, что идентичные URI трактуются, как разные. Кроме описанных выше различий в регистре символов некоторые создатели URI используют %-кодирование без необходимости. Такие URI следует нормализовать путем декодирования октетов с %-представлением, которые соответствуют незарезервированным символам, как описано в параграфе 2.3.

6.2.2.3. Нормализация сегмента пути

Полные сегменты пути «.» и «..» предназначены для использования только в относительных ссылках (параграф 4.1) и удаляются в процессе преобразования ссылки (параграф 5.2). Однако некоторые развернутые реализации некорректно предполагают, что преобразование ссылок не обязательно, если ссылка уже является URI и не удаляют сегменты с точками из путей, которые не являются относительными. Нормализаторам URI следует удалять сегменты с точками, используя описанный в параграфе 5.2.4 алгоритм remove_dot_segments.

6.2.3. Нормализация в зависимости от схемы

Синтаксис и семантика URI меняются от схемы к схеме и описаны в спецификации для каждой схемы. Разработчики могут применять определяемые схемой правила в качестве дополнительной обработки с целью снижения вероятности ложных различий. Например, поскольку схема http использует компоненту authority с принятым по умолчанию значением номера порта 80 и указывает, что пустой путь эквивалентен «/», приведенные ниже URI будут эквивалентны.

```
http://example.com
http://example.com/
http://example.com:/
http://example.com:80/
```

В общем случае URI, использующий базовый синтаксис для authority с пустым путем, следует нормализовать с путем «/». Аналогично, явное указание :port с пустым или принятым по умолчанию номером порта, эквивалентно отсутствию порта и разделителя «:», поэтому его следует удалять при нормализации на основании схемы. Например, второй идентификатор в приведенном выше примере является обычной формой URI для схемы http.

Другой ситуацией, когда нормализация зависит от схемы, является обработка пустой компоненты authority или пустой субкомпоненты host. Для многих схем в спецификациях указано, что пустые значения этих компонент считаются ошибкой, а в других пустые значений считают эквивалентом localhost или хоста конечного пользователя. Когда в схеме определено используемое по умолчанию значение authority и желательна ссылка URI на это значение, ссылку следует нормализовать, указав пустое значение authority в целях единообразия, краткости и поддержки других языков. Однако, если не пуста субкомпонента userinfo или port, хост должен задаваться явно, даже для используемого по умолчанию.

При нормализации не следует удалять разграничители пустых компонент, если это не задано в спецификации схемы. Например, URI «http://example.com/?» нельзя считать эквивалентом любому из приведенных выше. Аналогично, наличие или отсутствие разграничителей внутри субкомпоненты userinfo обычно важно для ее интерпретации. К фрагментам нормализация на основе схемы не применима — два URI, различающиеся лишь наличием в одном суффикса #, считаются разными независимо от схемы.

Некоторые схемы определяют дополнительные субкомпоненты, которые состоят из регистронезависимых данных, и это неявно позволяет нормализаторам преобразовывать эти данные к одном регистру (например, нижнему). Например, схемы URI, которые определяют субкомпоненту пути для включения имени хоста Internet (такие, как схема mailto), предполагают, что регистр символов в субкомпоненте не имеет значения и к ней может применяться нормализация регистра (например, mailto:Joe@Example.COM эквивалентно mailto:Joe@example.com, хотя базовый синтаксис предполагает учет регистра символов в компоненте path).

Возможны и другие, специфические для схемы варианты нормализации.

6.2.4. Нормализация в зависимости от протокола

Значительные усилия по снижению влияния ложный различий зачастую не приемлемы для систем индексации web и поэтому они пользуются более агрессивным сравнением URI. Например, встретив URI вида

```
http://example.com/data
```

с перенаправлением на URI, отличающийся только символом / в конце

```
http://example.com/data/
```

они в будущем станут считать эти идентификаторы эквивалентными. Такой подход приемлем лишь в тех случаях, когда эквивалентность четко демонстрируется как результатом доступа к ресурсу, так и общими соглашениями механизма разыменования схемы (в этом случае использование перенаправления HTTP служит для предотвращения проблем с относительными ссылками).

7. Вопросы безопасности

Сами по себе URI не представляют угроз безопасности. Однако, поскольку URI часто используются для предоставления краткого набора инструкций для доступа к сетевым ресурсам, следует с осторожностью интерпретировать данные в URI, чтобы предотвратить нежелательный доступ к данным и избежать включения данных, которые не следует представлять в открытом виде.

7.1. Надежность и согласованность

Нет никакой гарантии, что после использования URI для извлечения данных такая же информация будет получена и в будущем при обращении по тому же URI. Нет даже гарантии того, что информация, получаемая с помощью данного URI, будет похожа на ту, которая была получена в прошлый раз. Синтаксис URI не вносит ограничений для схем или организаций по использованию или поддержке их пространства имен. Такие гарантии могут быть получены лишь от лиц, контролирующих пространство имен и соответствующие ресурсы. Конкретная схема URI может определять дополнительную семантику (например, постоянство имен), если такая семантика требуется для всех «органов именования» в данной схеме.

7.2. Злонамеренные конструкции

Можно создать такой URI, что попытка выполнить кажущуюся безопасной и идемпотентной операцию (например, извлечение представления) будет фактически приводить к выполнению на удаленной стороне вредоносной операции. Такие опасные URI обычно создаются путем указания номера порта, отличающегося от того, который зарезервирован для соответствующего протокола. Клиент помимо своей воли соединяется с сайтом, где работает другая протокольная служба и данные в URI содержат инструкции, которые при интерпретации этим другим протоколом вызывают неожиданное воздействие. Распространенным примером такого злонамеренного применения является использование основанной на протоколе схемы с номером порта 25, для обманной отправки программой непредусмотренного сообщения через сервер SMTP.

Приложениям следует предотвращать разыменование URI, в которых указан номер порта TCP, отличающийся от общепринятого (из диапазона 0 — 1023), если протокол, который будет применяться для разыменования URI, совместим с протоколом, ожидаемом на общепринятом порту. Хотя IANA поддерживает реестр общепринятых портов,

приложениям следует давать пользователю возможность настройки своих ограничений, чтобы это не препятствовало развертыванию новых приложений.

Когда URI включает октеты %представления, которые соответствуют символам разграничения для данного протокола преобразования или разыменования (например, CR и LF для протокола TELNET), такое %-представление недопустимо декодировать до прохождения через этот протокол. Передача %-кодирования, которое может нарушать протокол, менее опасна, нежели интерпретация декодированных октетов, как дополнительных операций или параметров, которые могут приводить к неожиданным и, возможно, небезопасным действиям на удаленной стороне.

7.3. Внутренняя перекодировка

При разыменовании URI разбор зачастую выполняется как пользовательским агентом, так и одним или несколькими серверами. Например, в HTTP типичный пользовательский агент будет разбирать URI на пять основных компонент, обращаться к указанному authority серверу и передавать ему данные в компонентах authority, path и query. Типичный сервер будет принимать эту информацию, разбивать путь на сегменты, а query — на пары key/value и после этого выполнять определяемую приложением обработку для отклика на запрос. В результате общей проблемой безопасности для реализаций серверов, которые обрабатывают URI целиком или по отдельным компонентам, является корректная интерпретация октетов данных, представленных символами и %-кодами в данном URI.

Октеты с %-представлением должны декодироваться в некой точке процесса разыменования. Приложение должно разделить URI на компоненты и субкомпоненты до декодирования октетов, поскольку в противном случае декодированные октеты могут быть ошибочно восприняты в качестве разделителей. Проверки безопасности данных в URI следует выполнять после декодирования октетов. Однако следует отметить, что для кодирования %00 (NUL) может потребоваться специальная обработка и такие символы следует отвергать, если приложение не предполагает получение необработанных (raw) данных внутри компонент идентификатора.

Особую осторожность следует проявлять в случаях, когда процесс интерпретации пути URI включает использование файловой системы или связанных с ней функций. Файловые системы обычно используют особую трактовку специальных символов типа «/», «\», «:», «[« и «]» и специальных имен устройств типа «.», «..», «...», «aux», «lpt» и т. п. В некоторых случаях обычная проверка существования такого имени будет приводить к паузе в работе операционной системы или ненужным вызовам системных функций, что может создавать существенные угрозы в поане отказа служб или непредусмотренного обмена данными. В данной спецификации невозможно перечислить все такие значимые символы и имена устройств. Разработчикам следует определиться с зарезервированными именами и символами для устройств хранения, которые могут быть подключены к их приложениям и соответствующим образом ограничить использование данных, получаемых из компонент URI.

7.4. Редкие форматы адресов IP

Хотя синтаксис URI для IPv4address разрешает использовать только десятичное представление адресов IPv4 с разделением точками, многие реализации, обрабатывающие URI, используют функции своей операционной системы (например, gethostbyname() и inet_aton()) для преобразования символьной строки в реальный адрес IP. К сожалению такие функции зачастую позволяют обрабатывать значительно более широкий набор форматов, нежели указано в параграфе 3.2.2.

Например, многие реализации разрешают применять адреса в форме трех разделенных точками чисел, в котором последняя часть является 16-битовым значением, определяющим два правых байта сетевого адреса (например, сеть класса B). Аналогично форма из двух разделенных точками чисел интерпретируется, как 24 бита, определяющие три правых байта сетевого адреса (класс A), а одно число (без точки) интерпретируется, как 32-битовое значение, указывающее весь адрес. В дополнение к этой путанице некоторые реализации позволяют интерпретировать каждую из разделенных точками частей, как десятичное, восьмеричное или шестнадцатеричное значение в соответствии с принятыми в языке C обозначениями (0x или 0X в начале указывает шестнадцатеричное значение, 0 — восьмеричное, остальные цифры- десятичное).

Эти дополнительные форматы адресов IP не разрешается использовать в синтаксисе URI по причине их разной интерпретации на разных платформах. Однако могут возникать проблемы безопасности, если приложение предпримет попытку фильтровать доступ к ресурсам по адресам IP в форме символьных строк. Если фильтрация нужна, символьное представление адресов следует сначала преобразовать в численное и выполнять фильтрацию по числовым значениям, а не по префиксам или суффиксам символьного представления.

7.5. Конфиденциальная информация

Создателям URI не следует выпускать идентификаторы, включающие имя пользователя или пароль, которые должны держаться в секрете. Текст URI зачастую отображается в браузерах, сохраняется в открытой форме в закладках и записывается в протокольные файлы пользовательского агента и промежуточных приложений (прокси). Включение паролей в компоненту userinfo отменено и его следует трактовать, как ошибку (или просто игнорировать), за исключением тех редких случаев, когда параметр password осознанно сделан открытым.

7.6. Семантические атаки

Поскольку субкомпонента userinfo применяется редко и указывается перед host в компоненте authority, она может служить для создания URI, предназначенных для обмана пользователя (человека), показывая один (доверенный) орган именования (authority) и скрывая «шумом» другое значение authority. Например

```
ftp://cnn.example.com&story=breaking_news@10.0.0.1/top_story.htm
```

может убедить пользователя в том, что идентификатор указывает хост cnn.example.com, тогда как реально это будет 10.0.0.1. Отметим, что такие вводящие в заблуждение субкомпоненты userinfo на практике могут быть значительно длиннее, нежели в приведенном примере.

Обманные URI (типа показанного выше) служат для атак на представление пользователей о значении URI, а не на сами программы. Пользовательские агент могут снизить шансы таких атак, различая компоненты URI при отображении (например, выводя userinfo другим цветом), но это не является панацеей. Дополнительная информация о семантических атаках с использованием URI приведена в работе [Siedzik].

8. Согласование с IANA

Имена схем URI, определенные как <scheme> в параграфе 3.1, образуют пространство зарегистрированных имен, поддерживаемое IANA в соответствии с процедурами [BCP35]. Данный документ не предъявляет каких-либо требований к IANA.

9. Благодарности

Эта спецификация создана на основе RFC 2396 [RFC2396], RFC 1808 [RFC1808] и RFC 1738 [RFC1738], поэтому указанные там благодарности применимы и здесь. It also incorporates the update (with corrections) for IPv6 literals in the host syntax, as defined by Robert M. Hinden, Brian E. Carpenter, and Larry Masinter in [RFC2732]. In addition, contributions by Gisle Aas, Reese Anschultz, Daniel Barclay, Tim Bray, Mike Brown, Rob Cameron, Jeremy Carroll, Dan Connolly, Adam M. Costello, John Cowan, Jason Diamond, Martin Duerst, Stefan Eissing, Clive D.W. Feather, Al Gilman, Tony Hammond, Elliotte Harold, Pat Hayes, Henry Holtzman, Ian B. Jacobs, Michael Kay, John C. Klensin, Graham Klyne, Dan Kohn, Bruce Lilly, Andrew Main, Dave McAlpin, Ira McDonald, Michael Mealling, Ray Merkert, Stephen Pollei, Julian Reschke, Tomas Rokicki, Miles Sabin, Kai Schaetzl, Mark Thomson, Ronald Tschalaer, Norm Walsh, Marc Warne, Stuart Williams, and Henry Zongaro are gratefully acknowledged.

10. Литература

10.1. Нормативные документы

[ASCII] American National Standards Institute, "Coded Character Set -- 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[STD63] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[UCS] International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO/IEC 10646:2003, December 2003.

10.2. Дополнительная литература

[BCP19] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.

[BCP35] Petke, R. and I. King, "Registration Procedures for URL Scheme Names", BCP 35, RFC 2717, November 1999.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts — Application and Support", STD 3, RFC 1123, October 1989.

[RFC1535] Gavron, E., "A Security Problem and Proposed Correction With Widely Deployed DNS Software", RFC 1535, October 1993.

[RFC1630] Berners-Lee, T., "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", RFC 1630, June 1994.

[RFC1736] Kunze, J., "Functional Recommendations for Internet Resource Locators", RFC 1736, February 1995.

[RFC1737] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Resource Names", RFC 1737, December 1994.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[RFC1808] Fielding, R., "Relative Uniform Resource Locators", RFC 1808, June 1995.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and D. Jensen, "HTTP Extensions for Distributed Authoring – WEBDAV", RFC 2518, February 1999.

[RFC2557] Palme, J., Hopmann, A., and N. Shelness, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2557, March 1999.

[RFC2718] Masinter, L., Alvestrand, H., Zigmond, D., and R. Petke, "Guidelines for new URL Schemes", RFC 2718[1], November 1999.

[RFC2732] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.

[RFC3305] Mealling, M. and R. Denenberg, "Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations", RFC 3305, August 2002.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513[2], April 2003.

[1] Документ заменен RFC 4395, а тот — RFC 7595. *Прим. перев.*
[2] Документ заменен RFC 4291. *Прим. перев.*

[Siedzik] Siedzik, R., "Semantic Attacks: What's in a URL?", April 2001, <http://www.giac.org/practical/gsec/Richard_Siedzik_GSEC.pdf>.

Приложение A. ABNF для URI

```
URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part     = "//" authority path-abempty
              / path-absolute
              / path-rootless
              / path-empty

URI-reference = URI / relative-ref

absolute-URI  = scheme ":" hier-part [ "?" query ]

relative-ref  = relative-part [ "?" query ] [ "#" fragment ]

relative-part = "//" authority path-abempty
              / path-absolute
              / path-noscheme
              / path-empty

scheme        = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

authority     = [ userinfo "@" ] host [ ":" port ]
userinfo      = *( unreserved / pct-encoded / sub-delims / ":" )
host          = IP-literal / IPv4address / reg-name
port          = *DIGIT

IP-literal    = "[" ( IPv6address / IPvFuture  ) "]"

IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

IPv6address   =                            6( h16 ":" ) ls32
              /                       "::" 5( h16 ":" ) ls32
              / [               h16 ] "::" 4( h16 ":" ) ls32
              / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
              / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
              / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
              / [ *4( h16 ":" ) h16 ] "::"              ls32
              / [ *5( h16 ":" ) h16 ] "::"              h16
              / [ *6( h16 ":" ) h16 ] "::"

h16           = 1*4HEXDIG
ls32          = ( h16 ":" h16 ) / IPv4address
IPv4address   = dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet     = DIGIT                 ; 0-9
              / %x31-39 DIGIT         ; 10-99
              / "1" 2DIGIT            ; 100-199
              / "2" %x30-34 DIGIT     ; 200-249
              / "25" %x30-35          ; 250-255

reg-name      = *( unreserved / pct-encoded / sub-delims )

path          = path-abempty    ; начинается с / или пуст
              / path-absolute   ; начинается с /, но не с //
              / path-noscheme   ; начинается не с сегмента с двоеточием
              / path-rootless   ; начинается с сегмента
              / path-empty      ; 0 символов

path-abempty  = *( "/" segment )
path-absolute = "/" [ segment-nz *( "/" segment ) ]
path-noscheme = segment-nz-nc *( "/" segment )
path-rootless = segment-nz *( "/" segment )
path-empty    = 0<pchar>

segment       = *pchar
segment-nz    = 1*pchar
segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
              ; non-zero-length segment without any colon ":"

pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"

query         = *( pchar / "/" / "?" )
```

```
   fragment      = *( pchar / "/" / "?" )

   pct-encoded   = "%" HEXDIG HEXDIG

   unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
   reserved      = gen-delims / sub-delims
   gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"
   sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
                 / "*" / "+" / "," / ";" / "="
```

Приложение B. Разбор ссылок URI с регулярными выражениями

Алгоритм выбора первого подходящего (first-match-wins) идентичен «всеядному» методу устранения неоднозначностей, применемому в регулярных выражениях POSIX, поэтому естественным и комфортным будет использование регулярных выражений для анализа компонент URI.

В следующей строке приведено регулярное выражение для разбора корректно сформированной ссылки URI на компоненты.

```
/^(([^:\/?#]+):)?(\/\/([^\/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?/[1]
  12            3    4           5       6  7        8 9
```

Числа во второй строке приведены только для удобства понимания — они указывают «опорные точки» для каждого субвыражения (в парных скобках). Значение, соответствующее субвыражению <n> обозначается $<n>. Например, соответствие приведенному выше выражению для

```
http://www.ics.uci.edu/pub/ietf/uri/#Related
```

будут давать следующие субвыражения:

```
$1 = http:
$2 = http
$3 = //www.ics.uci.edu
$4 = www.ics.uci.edu
$5 = /pub/ietf/uri/
$6 = <не определено>
$7 = <не определено>
$8 = #Related
$9 = Related
```

где <не определено> говорит об отсутствии субкомпонеты (например, query в приведенном выше примере). Следовательно, мы можем определить значения пяти компонент как

```
scheme    = $2
authority = $4
path      = $5
query     = $7
fragment  = $9
```

Выполняя операции в обратном направлении, можно восстановить ссылку URI по компонентам с использованием алгоритма, описанного в параграфе 5.3.

Приложение C. Ограничители URI в контексте

URI часто передаются через форматы, которые не обеспечивают четкого контекста для их интерпретации. Например, достаточно часто URI включают в обычный текст — примерами могут служить сообщения электронной почты, новости USENET или печатные документы. В таких случаях важна возможность отделить URI от остального текста и, в частности, от знаков препинания, которые могут быть ошибочно восприняты как часть URI.

На практике URI зачастую обозначают тем или иным способом, но чаще для этого применяются двойные кавычки "http://example.com/", угловые скобки или просто пробелы

```
http://example.com/
```

Используемые в качестве ограничителей символы не являются частью URI.

В некоторых случаях в URI могут добавляться пробельные символы (пробел, перевод строки, табуляция и т. п.) для разбиения длинных URI на несколько строк. Эти пробельные символы следует игнорировать при извлечении URI.

Не следует добавлять пробельные символы после знака дефиса ("-"). Это связано с тем, что некоторые наборные машины и принтеры могут (ошибочно) добавлять дефис в конце строки при ее разрыве, поэтому интерпретаторам URI, сталкивающимся с переводом строки сразу после символа дефиса, следует игнорировать все пробельные символы «вокруг» перевода строки, а также удостовериться, является ли дефис частью URI.

Использование угловых скобок <> для каждого URI настоятельно рекомендуется в качестве ограничителей для ссылок, включающих в себя пробельные символы.

Префикс "URL:" (с пробелом или без пробела в конце) рекомендовали раньше для того, чтобы отличить URI от другого текста в скобках, хотя это не получило широкого распространения и в настоящее время такая рекомендация не применяется.

Для отказоустойчивости приложениям, принимающим набранные пользователем URI, следует пытаться распознать и вырезать ограничители и вложенные пробельные символы.

Например, приведенный ниже текст

[1] В оригинале была допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=2624. *Прим. перев.*

```
Да, Джим, я нашел это по ссылке "http://www.w3.org/Addressing/",
но, возможно, это доступно и по ссылке <ftp://foo.example.
com/rfc/>. Обратите внимание на предупрежденеи <http://www.ics.uci.edu/pub/
ietf/uri/historical.html#WARNING>.
```

содержит три ссылки URI

```
http://www.w3.org/Addressing/
ftp://foo.example.com/rfc/
http://www.ics.uci.edu/pub/ietf/uri/historical.html#WARNING
```

Приложение D. Отличия от RFC 2396

D.1. Дополнения

Введено правило ABNF для URI с целью соответствия общему применению термина «абсолютный URI с необязательным фрагментом».

Литералы IPv6 (и более поздние) были добавлены в список возможных идентификаторов для субкомпоненты host в authority, как описано в [RFC2732], с добавлением скобок [ ] в набор зарезервированных символов и флага версии для возможных будущих литералов IP. Квадратные скобки указаны в качестве зарезервированных символов для компоненты authority и не допускается их использование иначе, как ограничителей литерала IP в субкомпонентах host. Для того, чтобы внести это изменение без смены имеющегося технического определения компонент path, query и fragment, правила для которых были переопределены с прямым указанием разрешенных символов.

Поскольку определение литеральных адресов IPv6 из [RFC2732] было изменено в [RFC3513] определением, не содержащим описания ABNF для IPv6address, было создано новое правило ABNF для IPv6address, которое соответствует текстовому представлению, определенному в параграфе 2.2 [RFC3513]. Аналогично было изменено определение IPv4address для ограничения значений десятичного представления каждого октета диапазоном 0-255.

Раздел 6, посвященный нормализации и сравнению URI, был переписан полностью и расширен с использованием информации от Tim Bray и обсуждений в группе W3C Technical Architecture.

D.2. Изменения

Специальный синтаксис BNF RFC 2396 был заменен ABNF [RFC2234]. Это потребовало замены всех имен правил с символами подчеркивания именами, с символом дефиса. Кроме того, было исключено и упрощено множество правил, чтобы сделать общую грамматику более понятной. Спецификации, ссылающиеся на устаревшие грамматические правила можно понять, выполнив замены, указанные в приведенной ниже таблице.

Устаревшее правило	*Трансляция*
absoluteURI	absolute-URI
relativeURI	relative-part ["?" query]
hier_part	("//" authority path-abempty / path-absolute) ["?" query]
opaque_part	path-rootless ["?" query]
net_path	"//" authority path-abempty
abs_path	path-absolute
rel_path	path-rootless
rel_segment	segment-nz-nc
reg_name	reg-name
server	authority
hostport	host [":" port]
hostname	reg-name
path_segments	path-abempty
param	*<pchar excluding ";">
uric	unreserved / pct-encoded / ";" / "?" / ":" / "@" / "&" / "=" / "+" / "$" / "," / "/"
uric_no_slash	unreserved / pct-encoded / ";" / "?" / ":" / "@" / "&" / "=" / "+" / "$" / "," / "/"
mark	"-" / "_" / "." / "!" / "~" / "*" / "'" / "(" / ")"
escaped	pct-encoded
hex	HEXDIG
alphanum	ALPHA / DIGIT

Использование приведенный в таблице устаревших правил для определения синтаксиса конкретных схем не допускается.

Раздел 2, посвященный символам, был переписан, чтобы объяснить, какие символы, когда и по каким причинам были зарезервированы, даже если эти символа не применяются в базовом синтаксисе в качестве разделителей. Символы, которые обычно не безопасны для декодирования, включая восклицательный знак, (!), звездочку (*), одинарные кавычки (') а также открывающие и закрывающие скобки ((и)) включены в число зарезервированных для прояснения

различия между зарезервированными и незарезервированными символами ответа на наиболее частые вопросы разработчиков схем. Аналогично, был переписан раздел, посвященных %-кодированию и нормализаторы URI получили право декодировать любые %-представления, соответствующие незарезервированным символам. В общем случае термины escaped и unescaped были заменены на %-кодирование и декодирование, соответственно, для устранения путаницы с другими вариантами escape-механизмов.

ABNF для URI и ссылок URI был переработан с целью повышения уровня дружественности к анализаторам LALR и упрощения. В результате была удалена схематическая форма описания синтаксиса вместе с правилами uric, uric_no_slash, opaque_part, net_path, abs_path, rel_path, path_segments, rel_segment и mark. Все упоминания «непрозрачных» (opaque) URI были заменены более качественными описаниями, как компоненты пути могут быть непрозрачными для иерархии. Правило relativeURI было заменено на relative-ref для предотвращения ненужной путаницы с в части принадлежности к подмножеству URI. Неоднозначность при разборе ссылок URI, как URI или relative-ref с двоеточием в первом сегменте была устранена за счет использования пяти раздельных правил соответствия пути.

Идентификатор фрагмента был возвращен в раздел компонент базового синтаксиса для правил URI и relative-ref, хотя по прежнему исключается из absolute-URI. Знак номера (#) был возвращен в набор зарезервированных символов в результате возвращения синтаксиса фрагментов.

Представление ABNF было скорректировано для того, чтобы разрешить пустые компоненты пути. Это разрешает также absolute-URI, не содержащие ничего после «scheme:», как уже используется на практике в пространстве имен «dav:» [RFC2518] и схеме «about:», используемой внутри себя многими реализациями браузеров WWW. Неоднозначность границы между authority и path была устранена за счет использования пяти раздельных правил соответствия для пути.

Управляющие именованием на основе реестров, использующие базовый синтаксис, включены в правило для хоста. Это изменение позволяет современным реализациям, где любое представленное имя просто передается механизму преобразования имен, соответствовать данной спецификации, а также отменяет необходимость повторной спецификации здесь форматов имен DNS. Кроме того, это позволяет включать в компоненту host октеты %-представления, которые нужны для доменных имен на других языках, включаемых в URI, чтобы те могли обрабатываться в естественной для языка кодировке символов на прикладных уровнях выше уровня обработки URI и передаваться в библиотеку IDNA, как зарегистрированные имена с кодировкой символов UTF-8. Правила server, hostport, hostname, domainlabel, toplabel и alphanum были удалены из спецификации.

Алгоритм преобразования относительных ссылок [RFC2396] был переписан в данной версии в целях большей ясности и исправления перечисленных ниже ошибок.

- [RFC2396], параграф 5.2, п. 6a не принимал во внимание базовый URI без path.
- Восстановлено поведение [RFC1808] - в тех случаях, когда ссылка содержит пустой путь и присутствует компонента query, целевой URI наследует компоненту path тз базового URI.
- Определение того, ссылается ли URI на тот же документ, было отделено от разбора URI, что позволило упростить интерфейс обработки URI в приложениях за счет возможности использования внутренней архитектуры развернутых реализаций обработки URI. Определение сейчас основывается на сравнении базового URI после преобразования ссылки в абсолютную, а не по формату самой ссылки. Это изменение может приводить к увеличению числа ссылок на тот же документ в соответствии с данной спецификацией по сравнению с RFC 2396, особенно в случаях применения нормализации для снижения числа псевдонимов. Однако это не изменит состояния существующих ссылок на тот же документ.
- Процедура слияния пути разделена на две — собственно слияние для описания комбинации пути базового URI с путем относительной ссылки и remove_dot_segments для описания процедуры удаления специальных сегментов «.» и «..» из композитного пути. Алгоритм удаления сегментов с точками remove_dot_segments сейчас применяется для всех путей в ссылках URI с целью обеспечения однотипных реализаций и повышения производительности нормализации URI на практике. Это изменение влияет лишь на разбор аномальных ссылок и ссылок с той же схемой, где базовый URI имеет неиерархический путь.

Предметный указатель

Исключен в переводе.

Адреса авторов

Tim Berners-Lee

World Wide Web Consortium

Massachusetts Institute of Technology

77 Massachusetts Avenue

Cambridge, MA 02139

USA

Phone: +1-617-253-5702

Fax: +1-617-258-5999

EMail: timbl@w3.org

URI: http://www.w3.org/People/Berners-Lee/

Roy T. Fielding

Day Software

5251 California Ave., Suite 110

Irvine, CA 92617

USA

Phone: +1-949-679-2960

Fax: +1-949-679-2972

EMail: fielding@gbiv.com

URI: http://roy.gbiv.com/

Larry Masinter

Adobe Systems Incorporated

345 Park Ave

San Jose, CA 95110

USA

Phone: +1-408-536-3024

EMail: LMM@acm.org

URI: http://larry.masinter.net/

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

Полное заявление авторских прав

Интеллектуальная собственность

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the IETF's procedures with respect to rights in IETF Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.