Network Working Group
Request for Comments: 3402
Obsoletes: 2915, 2168
Category: Standards Track

M. Mealling
Verisign

October 2002

Система DDDS. Часть 2 - Алгоритм

Dynamic Delegation Discovery System (DDDS)

Part Two: The Algorithm

Статус документа

Этот документ содержит спецификацию стандартного протокола, предложенного сообществу Internet, и служит приглашением к дискуссии в целях развития. Текущее состояние стандартизации и статус описанного здесь протокола можно узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Аннотация

Этот документ описывает алгоритм DDDS[1]) для применения динамически отыскиваемых правил преобразования строк по отношению уникальным строкам приложений. Четко определенные правила преобразования будут отражать передачу полномочий управления информацией, связанной со строкой. Этот документ является частью серии документов, полностью указанной в Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS (RFC 3401). Важно отметить, что чтение и понимание каждого документа этой серии невозможно без прочтения остальных документов.

Оглавление

1. Введение

Система DDDS используется для реализации простого связывания строк с данными, обеспечивающего поддержку динамически настраиваемых систем передачи полномочий (делегирования). DDDS отображает некие уникальные строки на данные, хранящиеся в DDDS Database[2], путем итеративного применения правил преобразования строк, пока не будут достигнуты условия завершения.

Этот документ описывает общий алгоритм DDDS, а не какие-либо частные алгоритмы или сценарии использования. Вся серия документов указана в Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS (RFC 3401) [1]. Важно отметить, что чтение и понимание каждого документа этой серии невозможно без прочтения остальных документов.

История DDDS включает процесс, начавшийся с работы группы Uniform Resource Name. Когда были изначально сформулированы имена URN[3] [6], было принято решение определить полномочный сервер для URN так, чтобы (конструктивно) там не содержалось информации о местоположении в сети. Система может использовать базу данных с правилами, которые будут применяться к URN для отыскания информации о конкретных синтаксических блоках[4]. Эта система изначально получила название RDS[5] [7] и применялась только по отношению к URN.

---

[1]Dynamic Delegation Discovery System – динамическая система детектирования передачи полномочий.
[2]База данных DDDS.
[3]Uniform Resource Names – единообразные имена ресурсов.
[4]В оригинале - “chunks of syntax”. *Прим. перев.*
[5]Resolver Discovery Service – служба обнаружения преобразователей (имен).

С течением времени множество систем начало использовать такой же алгоритм и инфраструктуру для других, не связанных с URN, систем (см. примеры использования DDDS в главе 6). Это привело к необходимости изменения и разъяснения некоторых основополагающих допущений. Этот набор документов служит обновлением для исходных спецификаций URN, позволяющим разрабатывать новые приложения и базы правил стандартным способом.

Этот документ отменяет RFC 2168 [11] и RFC 2915 [9], а также обновляет RFC 2276 [7].

2. Терминология

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

Application Unique String – Уникальная строка Приложения

Строка, являющаяся начальными данными для приложения DDDS. Лексическая структура этой строки должна заключать в себе уникальный путь делегирования, который анализируется и трассируется повторяющимся выбором и применением правил перезаписи (Rewrite Rule).

Rewrite Rule – правило перезаписи

Правило, которое применяется к Application Unique String для создания нового ключа выбора нового правила перезаписи из базы правил или результирующей строки, которая возвращается вызвавшему приложению. Часто используется просто термин Правило (Rule).

First Well Known Rule – первое общеизвестное правило

Это правило перезаписи, которое определяется приложением и реально не хранится в базе правил (Rule Database). Это праило используется для создания первого корректного ключа.

Terminal Rule – конечное правило

Правило перезаписи, при использовании которого возвращается строка конечного результата процесса DDDS, а не другой ключ базы данных.

Application - Приложение

Набор протоколов и спецификаций, которые задают реальные значения для различных обобщенных частей алгоритма DDDS. Приложение должно определять синтаксис и семантику Application Unique String, First Well Known Rule и одной или множества баз данных, которые корректны для этого Приложения.

Rule Database – База правил

Любое хранилище правил, позволяющее по уникальному ключу идентифицировать набор правил, задающих этап передачи полномочий используемый для конкретного значения ключа (Key).

Services - службы

Общая база правил может использоваться для связывания различных служб с данной строкой Application Unique String (например, различные функции протокола, разные рабочие характеристики, географическое положение, обратная совместимость и т. п.). Возможными вариантами служб могут быть, например, получение сообщения по факсу/телефону/электронной почте, распределение нагрузки между web-серверами, выбор ближайшего сервера-зеркала, учет отношения цена/производительность и т. п. Эти Службы включаются как часть Правила, чтобы позволить Приложению принять решение о выборе на основе применимости той или иной ветви процесса или иных параметров Сервиса.

Flags - флаги

Большинство Приложений будет требовать для Правил способа сигнализировать Приложению, что некие Правила обеспечивают варианты результата[1] (например, различные форматы вывода, механизмы расширяемости, сигнализация конечного правила и т. п.), тогда, как другие правила этого не делают. Большинство Баз данных определяют поле Flags, которое Приложение может использовать для представления различных значений, выражаемых этими сигналами.

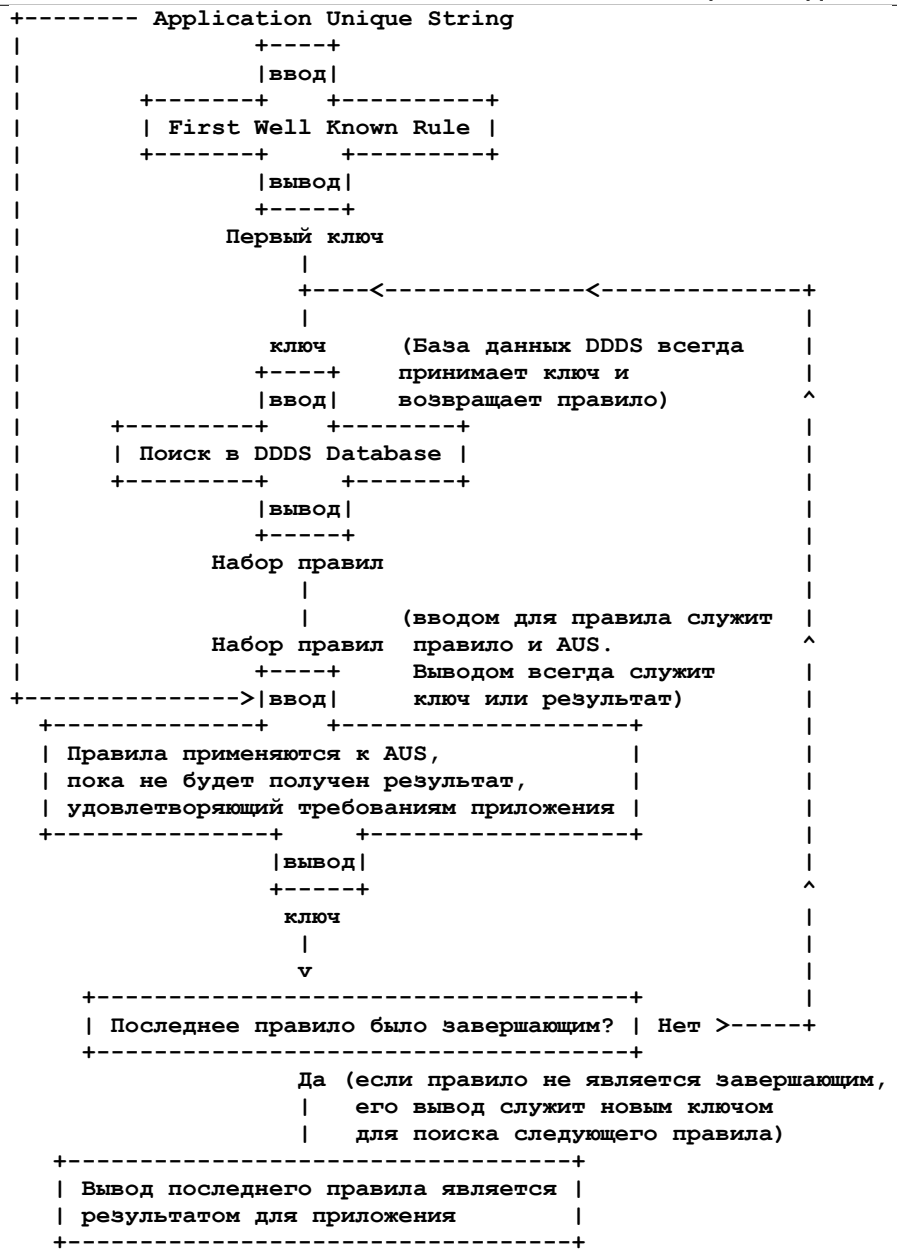
3. Алгоритм

Алгоритм DDDS основан на концепции Правил перезаписи (Rewrite Rule). Эти правила собираются в базу DDDS Rule Database, и доступ к ним обеспечивается по уникальным ключам. Конкретное Правило, будучи примененным к Application Unique String, преобразует эту строку в новый Ключ, который может использоваться для отыскания нового Правила в Базе. Это новое правило применяется к исходной строке Application Unique String и цикл повторяется, пока не будет выполнено условие завершения. Приложению **недопустимо** применять правило для вывода предыдущего правила. Все Правила перезаписи для всех Приложений должны **всегда** применяться точно к той же строке Application Unique String, с которой алгоритм начал работу.

Фундаментальны допущением является то, что строка Application Unique String представляет собой некий тип регулярной лексической структуры, к которой применяются правила. DDDS предполагает, что лексический элемент, используемый для принятия решения о передаче полномочий, содержится в самой строке Application Unique String. DDDS не решает задачи, когда для принятия решения о делегировании требуется информация, содержащаяся за пределами строки AUS и Правила (время сутом, финансовые транзакции и т. п..).

Схема работы алгоритма показана на рисунке.

[1] В оригинале - “ particular outcome”. *Прим. перев.*



3.1 Компоненты правила

Правило содержит до 4 компонент.

Приоритет

Число, которое определяет приоритет в случае выбора из двух правил, совпадающих в остальном. Это позволяет включать в базу данных правила, которые дают похожие результаты, но один из путей делегирования может быть быстрее, лучше или дешевле, чем другой.

Набор флагов

Флаги служат для задания атрибутов правила, которые определяют, является ли это правило последним из числа применяемых. Последнее правило называется завершающим (terminal rule) и его вывод служит результатом для приложения. Флаги являются уникальными для Приложений. Приложение может указать, что оно использует флаг, определенный другим Приложением, но в этом случае должно использоваться заданное тем Приложением определение. Одно Приложение не может переопределить Флаг, заданный другим Приложением. Это означает, что в будущем может потребоваться реестр флагов, но в настоящее время такое требование отсутствует.

Описание сервиса

Описание сервиса используется для указания семантических атрибутов определенной ветви передачи полномочий. Существует множество случаев, когда две ветви идентичны за исключением того, что одна ведет к результату, обеспечивающему один набор функций, в вторая – к результату с другим набором. Эти функции могут включать эксплуатационные параметры типа распределения нагрузки, разделения трафика по географическим признакам, совместимости со старыми версиями и т. п. Например, два правила могут быть одинаково применимы к конкретному решению о делегировании для строки. Одно правило может вести к заключительному правилу, которое обеспечивает информацию для использования в среде с высоким уровнем доступности, а другое - к архивной службе, которая работает более медленно, но более стабильна во времени.

Выражение для замены

Эта часть правила задает изменение строки и представляет собой комбинацию расширенного регулярного выражения[1] POSIX [8] и строки замены (подобно выражениям для замены в Unix-редакторе sed).

[1]Extended Regular Expression.

3.2 Синтаксис выражения для замены

Набор (наборы) символов, используемый в выражениях для замены, определяется как Приложением, так и Базой данных. Приложение должно определить допустимый набор символов для строки AUS. Спецификация DDDS Database должна определить, какие наборы символов требуются для создания ключей и как представляется само выражение для замены. Приведенные ниже символы имеют смысл лишь при выборе определенного набора символов в базе данных и/или Приложении.

Синтаксис Выражения для замены в правилах представляет собой выражение в стиле редактора sed. Сами по себе выражения в стиле sed не подходят для использования в приложениях по ряду причин., следовательно, содержимое поля regexp **должно** соответствовать приведенным ниже правилам.

```
subst-expr   = delim-char  ere  delim-char  repl  delim-char  *flags
delim-char   = "/" / "!" / <любой октет, отсутствующий в полях POS-DIGIT и flags>
                    ; Все вхождения delim_char в subst_expr должны быть одинаковы
ere          = <POSIX Extended Regular Expression>
repl         = *(string / backref)
string       = *(anychar / escapeddelim)
anychar      = <любой символ, отличный от delim-char>
escapeddelim = "\" delim-char
backref      = "\" POS-DIGIT
flags        = "i"
POS-DIGIT    = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
```

Применение выражения для замены к Строке **должно** давать в результате ключ, который соответствует правилам базы данных (естественно, если текущий ключ не является завершающим, при котором возвращается не новый ключ, а окончательный результат). Поскольку имеется возможность некорректно задать регулярное выражение, что будет приводить к созданию неприемлемого ключа, клиентской программе **следует** убедиться, что результат является корректным ключом базы данных, прежде, чем использовать этот ключ.

Выражения backref в repl-части выражения для замены меняются на строку (возможно пустую) символов, заключенных в круглые скобки () в ERE-части выражения для замены. N представляет собой одиночную цифру от 1 до 9, включительно. Эта цифра задает N-ое выражение backref, которое начинается с N-ой скобки (и продолжается до соответствующей скобки). Например, ERE

```
(A(B(C)DE)(F)G)
```

имеет backref-выражения:

```
\1  = ABCDEFG
\2  = BCDE
\3  = C
\4  = F
\5..\9  = ошибка – нет соответствующего субвыражения
```

Флаг i показывает, что соответствие ERE **следует** проверять без учета регистра символов. Более того, любые backref-подстановки **могут** быть нормализованы к нижнему регистру при наличии флага i. Этот флаг имеет смысл только в тех случаях, когда Приложение и База данных определяют набор символов, в котором отказ от учета регистра символов является корректным.

Первый символ выражения для замены следует использовать, как символ-ограничитель компонент этого выражения. В выражении для замены должно быть ровно три символа-ограничителя без предшествующего escape-символа. Поскольку символы-ограничители с предшествующим символом escape будут интерпретироваться как собственно символы (не ограничители), **недопустимо** использовать в качестве ограничителей цифровые символы. При включении цифр будут возникать проблемы в выражениях backref. Кроме того, при наличии флагов в выражении для замены символ флага недопустимо использовать в качестве ограничителя.

3.3 Полный алгоритм

Ниже приводится алгоритм DDDS целиком.

1. Применяется правило First Well Known Rule к строке Application Unique String и в результате возвращается ключ Key.
2. Приложение запрашивает Базу данных для получения упорядоченного списка правил, соответствующих ключу Key[1].
3. Для каждого Правила в списке используется замена с помощью Substitution Expression, применяемая с соблюдением порядка к строке Application Unique String, пока не будет возвращена непустая строка. Положение в списке фиксируется и породившее непустую строку Правило используется для следующего шага. Если на следующем шаге выбранное правило отвергается, происходит возврат к использованию Substitution Expression для оставшейся части списка правил. Если список завершится без возврата корректно соответствия, приложение уведомляется об этом.

[1]В некоторых приложениях и/или базах данных результирующий набор может выражать случай, когда два или более Правила рассматриваются как эквивалентные. Такие Правила трактуются как одно Правило; каждое из возвращенных правил может иметь значение Priority, используемое для выбора предпочтительного Правила из набора эквивалентных. Это позволяет использовать Правило взамен другого при отказе того (fallbacks). Следует отметить, что этот механизм определяет предпочтения (Preference), а не распределение нагрузки. Приложениям следует аккуратно определять разницу.
Базы данных могут иметь (или не иметь) правила, которые определяют когда и как завершается срок корректности записей в базе (даты истечения срока, время жизни и т. п.). Эти механизмы должны применяться во всех случаях. В частности, поскольку завершение срока действия записи в базе может вынудить к поиску нового Правила, которое будет несовместимо с предыдущим Правилом, хотя в алгоритме любые попытки оптимизировать процесс путем возврата к предыдущим ключам и Правилам **должны** гарантировать, что срок действия ранее найденного правила не истек. Если срок действия Правила истек, приложение **должно** начать операции с п. 1.

4. Если Описание сервиса в правиле не соответствует требованиям клиента, происходит возврат к п. 3 и продолжение работы с оставшейся частью списка правил. Если в списке имеется правило, соответствующее требованиям клиента, это Правило используется для следующего шага. В том (и только в том) случае, когда клиент способен обработать Правило и если такая обработка представляется безопасной в соответствии со спецификацией Приложения, клиент может отметить текущее Правило и продолжить выполнение п. 3, как будто правило было отвергнуто. В любом случае результатом данного этапа является одно и только одно Правило.

5. Если Flags в Правиле показывает, что данное правило **не является** завершающим (NOT Terminal), происходит возврат к п. 2 с использованием полученного результата в качестве нового Ключа.

6. Приложение уведомляется о завершении процесса и ему передаются из Правила значения Flags и Services вместе с выводом последнего выражения Substitution Expression.

4. Спецификация Приложения

Для того, чтобы описанный алгоритм стал используемым, нужно написать спецификации приложения и по крайней мере одной базы данных. Для задания спецификации приложения требуются перечисленные ниже компоненты.

Application Unique String – Уникальная Строка Приложения

Это единственная строка, к которой применяются правила перезаписи. Строка должна иметь регулярную структуру и быть уникальной в приложении так, чтобы все, кто применяет Правила из одной Базы данных, получали в результате одинаковые Ключи. Например, приложение URI Resolution в качестве Application Unique String определяет URI.

Никаким приложениям не дозволено определять Application Unique String так, чтобы Ключ, полученный правилом перезаписи, трактовался бы как Application Unique String для ввода в новое правило, поскольку это ведет к правилам перезаписи в стиле sendmail, которые отличаются склонностью к ошибкам. Единственное исключение из этого правила возможно в случае, когда Приложение определяет некий флаг или состояние, в котором правила для приложения подавляются и принимается новое Приложение DDDS или некий произвольный набор правил. Один из таких случаев можно найти в приложении URI Resolution, которое определяет флаг p, указывающий, что следующий шаг зависит от протокола (protocol specific) и, таким образом, выходит за рамки DDDS.

First Well Known Rule – Первое Общеизвестное Правило

Это первое правило, которое применяется к Application Unique String и создает первый Ключ. Для выражения может использоваться такая же форма, какая принята для Правил, или несколько более сложная форма. Например, приложение URI Resolution может задавать, что это правило возвращает в качестве первого Ключа последовательность символов URI вплоть (но не включая) до первого двоеточия (схема URI).

Valid Databases – корректные Базы данных

Приложение может определить, какая из Баз данных является корректной. Для каждой базы данных Приложение должно определить, как вывод правила First Well Known Rule (первый Ключ) может быть обращен во что-либо, подходящее для этой Базы данных. Например, приложение URI Resolution может использовать в качестве Базы данных систему DNS[1]. Операция обращения первого Ключа во что-либо, подходящее для базы данных, будет преобразованием в некое корректное с точки зрения DNS имя. В дополнение для каждой Базы данных, которую определяет Приложение, оно должно задать допустимые наборы символов, которые будут порождать корректные Ключи. В примере URI Resolution набором символов URI является 7-битовый код ASCII, который согласуется с 8-битовым ограничением для символов в файлах зон DNS.

Expected Output – ожидаемый результат

Приложение должно определить ожидаемый результат Завершающего Правила. Например, приложение URI Resolution направлено на поиск серверов, которые содержат полномочные данные о данном URI. Таким образом, результатом завершающего правила будет информация (хосты, порты, протоколы и т. п.), которая будет служить для контакта с полномочным сервером.

В прошлом были некоторые недоразумения, касающиеся распределения нагрузки и использования DDDS Priority. Приложениям следует понимать, что Priority данного правила является лишь способом показать, что одно правило “быстрее, лучше, дешевле” другого. Если приложению требуется некий метод, позволяющий клиенту распределять нагрузку между серверами (например, взвешенный случайный выбор и т. п.), это следует реализовать за пределами алгоритма DDDS. Например, Приложение, использующее DNS Database может трактовать запись SRV, как способ обозначения того, что определенная служба действительно поддерживается несколькими скооперированными хостами. Распределение нагрузки будет осуществляться между хостами, которые идентичны с точки зрения DDDS в части путей передачи полномочий, которые имеют некий общий набор свойств или административный домен.

5. Спецификация Базы данных

В дополнение к сказанному каждое Приложение должно иметь по крайней мере одну Базу данных, в которой отыскиваются Правила. Важно отметить, что данная База может использоваться и другими Приложениями. В таких случаях каждое правило должно использовать некую комбинацию своих служб (Service) и/или выражений для замены, соответствующую только той строке Application Unique Strings, для которой правило корректно.

Спецификация базы данных должна включать перечисленные ниже компоненты.

General Specification – общая спецификация

База данных должна иметь общую спецификацию, в качестве которой может использоваться ссылка на другой стандарт (SQL, DNS и т. п.) или полностью определенная новая спецификация. Спецификация Базы данных **должна** четко указывать допустимый набор символов, чтобы знать, какие наборы символов используются для представления Ключей и Правил.

Lookup Procedure – процедура поиска

Этот параметр задает способ формулировки и подачи запросов. В случае использования базы данных для других целей (например, DNS) спецификация должна ясно показывать, как формулируются запросы применительно к случаю DDDS. Например, База данных на основе DNS должна указывать используемые типы записей RR[2] и запросов (Query).

[1]Domain Name System – система доменных имен.
[2]Resource Record – запись о ресурсе.

Key Format – формат ключа

Если требуются какие-либо операции для преобразования Ключа во что-либо корректное для базы данных, такие преобразования должны быть четко определены. Например, в случае базы данных DNS Ключи должны создаваться как корректные доменные имена.

Rule Format – формат правила

Спецификация выходного формата для правила.

Rule Insertion Procedure процедура вставки правила

Спецификация вставки Правил в базу данных. Эта спецификация может включать политику, в соответствии с которой Правила могут или не могут добавляться в базу данных.

Rule Collision Avoidance – предотвращение конфликтов между правилами

Поскольку База данных может использоваться множеством Приложений (например, ENUM и URI Resolution), спецификация должна четко описывать способы предотвращения конфликтов между правилами. Обычно для решения этой задачи используются два метода: 1) запрет для одного ключа возможности быть корректным в двух разных Приложениях; 2) если вариант 1 невозможен, создается такое выражение для замены, чтобы регулярное выражение включало достаточную часть Application Unique String для идентификации различия между двумя Приложениями.

6. Примеры

Приведенные ниже примеры служат лишь для учебных целей. Рассмотренные приложения не специфицированы в каких-либо реальных документах.

6.1 Система идентификации автомобильных деталей

В этом примере рассматривается воображаемая система, в которой собраны воедино все производители автомобилей и используется стандартизованная система кодирования различных деталей (болты, гайки, шайбы, инструменты и т. п.), которая применяется в процессе производства и ремонта автомобилей. Проблема такого рода систем заключается в том, что автомобильная промышленность является сильно распределенной и многие детали производятся различными фирмами, расположенными по всему миру. Для того, чтобы найти информацию о какой-либо детали, система должна быть способна выяснить производителя этой детали и контактировать с ним.

Для облегчения задачи идентификационные номера выделяются как часть иерархической системы, в которой первые 5 цифр являются идентификатором производителя. Следующие 3 цифры идентифицируют линейку автомобилей (например, Ford, Toyota). Остальные цифры присваиваются производителями деталей по своему усмотрению.

Автомобильная промышленность приняла решение об использовании DDDS для создания распределенной системы поиска информации, которая маршрутизирует запросы к реальным держателям данных. Отрасль специфицировала базу данных и синтаксис запросов для отыскания правил перезаписи (APIDA Network) и Приложение Auto Parts Identification DDDS Application (APIDA).

Спецификация APIDA будет определять следующее:

- Application Unique String – код детали;
- First Well Known Rule – принимает первые 5 цифр (идентификатор производителя) и использует их в качестве Ключа;
- Valid Databases - APIDA Network.
- Expected Output – информация EDIFAC об интересующей детали.

Спецификация Базы данных APIDA Network будет определять следующее:

- General Specification – сеть поддерживающих EDI баз данных и служб, которые по заданному субномеру детали будут возвращать правила перезаписи в формате XML;
- Lookup Procedure – следует обычным протоколам APIDA Network, запрашивая в сети правило перезаписи для Ключа;
- Key Format – преобразования не требуется;
- Rule Format: см. документацию APIDA Network для XML DTD.
- Rule Insertion Procedure – определяется агентством, которое контролирует каждую часть кодов деталей; например, для получения идентификатора производителя нужно быть членом Auto Parts Manufacturers Association.

В качестве иллюстрации рассмотрим деталь с кодом 4747301AB7D. Система будет брать первые 5 цифр 47473 и запрашивать в сети Правило перезаписи (Rewrite Rule). Это правило будет предоставляться базой данных о производителях деталей и позволит производителю передать полномочия далее или напрямую возвратить запрашивающему информацию EDIFAC.

Предположим для примера, что производитель возвращает Правило, которое говорит, что следующие 3 цифры нужно использовать как часть запроса к их сервису для получения нового Правила. Это новое Правило позволит производителю деталей передать полномочия на заводы по производству деталей для каждой линейки автомобилей. В нашем примере цифры 01A означают автомобили Toyota. Правило, которое возвращает производитель передает полномочия далее производителю в Японии. Это правило также означает, что данное Правило является завершающим и, таким образом, результатом последнего запроса будет информация о детали.

6.2 Служба идентификации документов

Этот пример напоминает предыдущий, поскольку документы в этой системе можно рассматривать аналогично автомобильным деталям. Различие состоит в том, что информация о документе хранится близко к его автору (обычно

на его настольном компьютере). Это означает, что число передач полномочий может быть достаточно большим. Кроме того, для того, чтобы избавиться от необходимости поддержки большого плоского пространства авторов, они организованы по компаниям и подразделениям.

Предположим, что строка Application Unique String для данного примера имеет вид:

```
<organization>-<department>-<author>:<project>-<bookcase>-<book>
```

Спецификация приложения может выглядеть примерно так:

- Application Unique String - приведенная выше строка идентификации документа;
- First Well Known Rule – символы до (но не включая) “-” трактуются, как первый Ключ;
- Valid Databases – каталог DIS LDAP;
- Expected Output – запись с сервера LDAP, содержащая библиографические данные о документе в формате XML.

Спецификация базы данных для DIS LDAP Directory будет иметь вид:

- General Specification – база данных использует службу каталогов LDAP; каждый сервер LDAP имеет запись, содержащую Правило перезаписи; правила указывают на другие серверы LDAP, используя схему LDAP URL;
- Lookup Procedure – используя стандартные запросы LDAP, клиент спрашивает у сервера LDAP информацию о ключе;.
- Key Format – преобразование не требуется;
- Rule Format – см. спецификацию LDAP Rewrite Rule;
- Rule Insertion Procedure – см. процедуры, опубликованные стороной, имеющей полномочия для данной ветви дерева DIS; первым разделом (организация) владеет Агентство DIS.

В этом примере первый поиск осуществляется по организации. В результате организация может может возвратить клиенту конечный результат, если база данных этой организации содержит его. В других случаях процесс может включать дополнительные Правила и делегировать запрос вниз по иерархии к владельцу документа.

7. Вопросы безопасности

Этот документ просто определяет алгоритм DDDS и, таким образом, сам по себе не оказывает влияния на безопасность. Когда алгоритм объединяется с Базой данных и Приложением, могут возникать связанные с безопасностью вопросы, одним из которых является возможность организации атак на динамические точки делегирования.

8. Взаимодействие с IANA

Этот документ не создает каких-либо требований к IANA. Спецификации Баз данных и Приложений могут выдвигать такие требования, но они не рассматриваются здесь.

Литература

[1] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS", RFC 3401, October 2002.

[2] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm", RFC 3402, October 2002.

[3] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.

[4] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI) Resolution Application", RFC 3404, October 2002.

[5] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Five: URI.ARPA Assignment Procedures", RFC 3405, October 2002.

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

[7] Sollins, K., "Architectural Principles of Uniform Resource Name Resolution", RFC 2276, January 1998.

[8] The Institute of Electrical and Electronics Engineers, "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1)", IEEE Std 1003.2-1992, ISBN 1-55937-255-9, January 1993.

[9] Mealling, M. and R. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record", RFC 2915, August 2000.

[10] Faltstrom, P., "E.164 number and DNS", RFC 2916, September 2000.

[11] Daniel, R. and M. Mealling, "Resolution of Uniform Resource Identifiers using the Domain Name System", RFC 2168, June 1997.

Адрес автора

Michael Mealling

VeriSign

21345 Ridgetop Circle

Sterling, VA 20166

US

EMail: michael@neonym.net

URI: http://www.verisignlabs.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Полное заявление авторский прав

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.