Кодирование интернациональных доменных имен (IDNA)
(rfc3490, http://www.ietf.org/rfc/rfc3490.txt)

1. Интернациональные имена задаются в UNICODE. В существующей инфраструктуре DNS используется ASCII (7 бит).
2. Существующую инфраструктуру DNS менять нельзя! Поэтому интернациональные имена должны быть преобразованы.
3. Для того, чтобы отличать преобразованные имена от обычных будет использоваться специальный префикс (ACE label, ACE = ASCII Compatible Encoding). Для IDN префиксом задано = xn--
4. Необходимо обеспечить две функции – toASCII и toUNICODE.
5. Ошибки в преобразовании допустимы только для функции toASCII. При возникновении ошибки трансляция недопустима.
6. Строки должны быть определённым образом обработаны, чтобы исключить появление недопустимых символов в именах доменов (STRIGPREP, http://www.ietf.org/rfc/rfc3454.txt)

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

Исходное (интернациональное имя) - при-вет.рф.
Преобразованное имя (DNA) - xn--p-e-gdd2a4b0a.xn--p1ai.

Кодирование происходит по доменам независимо.

U+043F | U+0070 | U+0438 | U+002D | U+0432 | U+0065 | U+0442

п р и - в е т

Для каждого домена:

1. Задается префикс. = **xn--**
2. Символы из таблицы ASCII копируются в результирующую строку в той последовательности, в которой они появлялись в исходной строке. Затем ставится символ разделитель (-, hyphen, 0x2D) = **p-e-**
3. Остальные символы кодируются по алгоритму BOOTSTRING. = **gdd2a4b0a**

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

Основная идея – преобразование чисел, записанных в позиционных систем счисления, в десятичное значение.

$$437_8 = 7 + 3*8^1 + 4*8^2 = 287_{10}$$

веса цифр задаются - w(0) = 1, w(1) = base, w(2) = w(1) * base, w(3) = w(2) * base и т.д.

Проблема – последовательная запись цифр (без символов разделителей) в исходной системе счисления приводит к путанице.
Направление увеличения весов (Little-endian или Big-endian).

Например – $734251\ldots_8$

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

Решение – зададим алгоритм определения окончания цифры с помощью функции порога t(j), которая будет использована и для определения весов:

$$w(0) = 1, \quad w(j) = w(j-1) * (base - t(j-1))$$

$$t(j) = base * (j + 1) - bias,$$

$$0 <= \mathbf{t}_{\mathbf{min}} <= t(j) <= \mathbf{t}_{\mathbf{max}} <= base$$

Число завершается тогда, когда его очередная цифра становится меньше, чем t(j).

Bias – это способ определения количества цифр в числе.

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

$734251\ldots_8$ (используем обратный порядок записи цифр, base = 8, bias = 13)

$$t(j) = 2, 3, 5, 5, 5 , 5 \ldots$$

$$w(j) = 1, 6, 30, 90, 270\ldots$$

В строке два числа

$$734_8 = 145_{10} \qquad 251_8 = 62_{10}$$

Перевод = 7 * 1 + 3 * 6 + 4 * 30 = 145

Обратно (итерации) =

(t + (q – t) mod (base – t)), q = (q – t) div (base – t)

145 > 2, 2 + (145 – 2) mod (8 – 2) = 7, q = 23

23 > 6, 3 + (23 – 3) mod (8 – 3) = 3, q = 4

4 < 5, 4 = 4

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

U+043F | U+0070 | U+0438 | U+002D | U+0432 | U+0065 | U+0442

п р и - в е т

1. Кодируются расстояния (в терминах UNICODE) между символами с учетом их месторасположения в исходной строке.
2. Символы кодируются по мере возрастания их кодов.
3. Базовым считается символ с кодом 0x80 (128_{10}).
4. Местоположение задается двумя координатами одностадийной машины (n, i), где n – количество полных просмотров строки (с текущей длинной), i – смещение относительно начала строки.

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

Для оптимизации длины строки, для каждого следующего символа пересчитывается bias (с целью предсказания количества символов в результирующей строке).

1. Масштабируем текущее расстояние, чтобы исключить переполнение кодирования:

   Delta = delta / (2 или damp (если первое деление));

2. Увеличиваем расстояние в предположении, что следующее расстояние будет больше:

   Delta = delta + (delta div количество обработанных символов)

3. Считаем во сколько раз расстояние выше порогового значения

   While delta > ((base – tmin) * tmax) div 2

   do let delta = delta div (base – tmin)

4. Считаем новое значение коэффициента

   Bias = (base * результат п.3) + (((base – tmin + 1) * delta) div (delta + skew))

В нашем случае используется две системы счисления:

- UNICODE – целое число без знака (16 разрядов, диапазон – 0 до 65536). Каждое число – это один интернациональный символ.
- IDN – 36-ричная система (значения от 0 до 35). Числа отображаются на допустимые символы ASCII: 0-25 -> a-z (A-Z), 26-39 -> 0-9. Символ – (тире) в системе кодирование не используется, так как считается разделителем полей.

Регистр символов задается отдельно!!!

Начальные значения базовых переменных:

base = 36

tmin = 1

tmax = 26

skew = 38

damp = 700

bias = 72

n = 0x80

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

U+043F | U+0070 | U+0438 | U+002D | U+0432 | U+0065 | U+0442

п р и - в е т

Строка результата (уже скопированы символы ASCII) = p-e-
(всего 4 символа, h = 3, delta = 0).

Шаг 1. Кодируется символ ‘**в**’ (U+0432). Расстояние (432_{16}-80_{16}) = 946_{10}
Символов в исходной строке до кодируемого (2 символа, p, -).
Расстояние для кодирования = 0 + 946*(3 + 1) + 2 = 3786.
Кодируем: получается 6, 3, 3 => g, d, d
Смотрим дальше. Delta = 2.

Шаг 2. Кодируется символ ‘**и**’ (U+0438). Расстояние (438_{16}-433_{16}) = 5_{10}
Символов в исходной строке до кодируемого (1 символ, p).
Расстояние для кодирования = 2 + 5*(4 + 1) + 1 = 28.
Кодируем: получается 28, 0 => 2, a
Смотрим дальше. Delta = 4.

Шаг 3. Кодируется символ ‘**п**’ (U+043F). Расстояние ($43F_{16}$-439_{16}) = 6_{10}
Символов в исходной строке до кодируемого (0 символов).
Расстояние для кодирования = 4 + 6*(5 + 1) + 0 = 40.
Кодируем: получается 30, 1 => 4, b
Смотрим дальше. Delta = 6.

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

U+043F | U+0070 | U+0438 | U+002D | U+0432 | U+0065 | U+0442

п р и - в е т

Шаг 4. Кодируется символ ‘**т**’ (U+0442). Расстояние (442_{16}-440_{16}) = 2_{10}
Символов в исходной строке до кодируемого (6 символов).
Расстояние для кодирования = 6 + 2*(6 + 1) + 6 = 26.
Кодируем: получается 26, 0 => 0, a
Смотрим дальше. Delta = 0.

Символы закончились.

Результат кодирования = xn--p-e-gdd2a4b0a

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

Декодируем строку = xn--p-e-gdd2a4b0a

Шаг 1. Исключаем префикс. Копируем все символы до последнего разделителя:
out = p-e

Шаг 2. Декодируем первое расстояние (bias = 72, out = 3, n = 0x80)

t(0) = 1, t(1) = 1, t(2) = 26,
w(0) = 1, w(1) = 35, w(3) = 1225,
‘g’=6, ‘d’ = 3, ‘d’ = 3…. Значение = 3786.
Определяем код символа: n += (значение div (out + 1)) = 1074 (0x0432)
Определяем позицию вставки в строку: i = (значение mod (out + 1)) = 2.
Результат будет = р-**в**е
Адаптируем bias (4).

Шаг 2. Декодируем первое расстояние (bias = 4, out = 4, n = 0x0433)

t(0) = 26, t(1) = 26
w(0) = 1, w(1) = 10
‘2’=28, ‘a’ = 0, ... Значение = 31.
Определяем код символа: n += (значение div (out + 1)) = 1080 (0x0438)
Определяем позицию вставки в строку: i = (значение mod (out + 1)) = 1.
Результат будет = р**и**-ве
Адаптируем bias (10).

Алгоритм BOOTSRING (PUNYCODE) преобразования строк для IDNA (rfc3492, http://www.ietf.org/rfc/rfc3492.txt)

Декодируем строку = xn--p-e-gdd2a4b0a

Шаг 3. Декодируем первое расстояние (bias = 10, out = 5, n = 0x0439)

t(0) = 26, t(1) = 26

w(0) = 1, w(1) = 10,

‘4’=30, ‘b’ = 1 …. Значение = 42.

Определяем код символа: n += (значение div (out + 1)) = 1087 (0x043F)

Определяем позицию вставки в строку: i = (значение mod (out + 1)) = 0.

Результат будет = **п**р-ве

Адаптируем bias (13).

Шаг 4. Декодируем первое расстояние (bias = 13, out = 6, n = 0x0440)

t(0) = 23, t(1) = 26

w(0) = 1, w(1) = 13

‘0’=26, ‘a’ = 0, ... Значение = 27.

Определяем код символа: n += (значение div (out + 1)) = 1090 (0x0442)

Определяем позицию вставки в строку: i = (значение mod (out + 1)) = 6.

Результат будет = при-ве**т**

Адаптируем bias (9).