NAME

Net::LibIDN − Perl bindings for GNU Libidn

SYNOPSIS

```
use Net::LibIDN ':all';

idn_to_ascii("Räksmörgås.Josefßon.ORG") eq
  idn_to_ascii(idn_to_unicode("xn--rksmrgs-5wao1o.josefsson.org"));

idn_prep_name("LibÜDN") eq "libüdn";

idn_punycode_encode("kistenmöhre") eq
  idn_punycode_encode(idn_punycode_decode("kistenmhre-kcb"));

my $errpos;
tld_check("mèrle.se", $errpos) eq undef;
  $errpos == 1;

tld_get("mainbase.mars") eq "mars";

my $hashref = Net::LibIDN::tld_get_table("de");

print "$hashref->{version}\n";
foreach (@{$hashref->{valid}})
{
  print "Unicode range from ".$_->{start}." to ".$_->{end}."\n";
}
```

DESCRIPTION

Provides bindings for GNU Libidn, a C library for handling Internationalized Domain Names according to IDNA (RFC 3490), in a way very much inspired by Turbo Fredriksson's PHP-IDN.

Functions

Net::LibIDN::idn_to_ascii(*$clear_hostname*, [*$charset*, [*$flags*]]);

Converts *$clear_hostname* which might contain characters outside the range allowed in DNS names, to IDNA ACE. If *$charset* is specified, treats string as being encoded in it, otherwise assumes it is ISO−8859−1 encoded. If flag **IDNA_ALLOW_UNASSIGNED** is set in *$flags*, accepts also unassigned Unicode characters, if **IDNA_USE_STD3_ASCII_RULES** is set, accepts only ASCII LDH characters (letter-digit-hyphen). Flags can be combined with ||. Returns result of conversion or **undef** on error.

Net::LibIDN::idn_to_unicode(*$idn_hostname*, [*$charset*, [*$flags*]]);

Converts ASCII *$idn_hostname*, which might be IDNA ACE encoded, into the decoded form in *$charset* or ISO−8859−1. Flags are interpreted as above. Returns result of conversion or **undef** on error.

Net::LibIDN::idn_punycode_encode(*$string*, [*$charset*]);

Encodes *$string* into "punycode" (RFC 3492). If *$charset* is present, treats *$string* as being in *$charset*, otherwise uses ISO−8859−1. Returns result of conversion or **undef** on error.

Net::LibIDN::idn_punycode_decode(*$string*, [*$charset*]);

Decodes *$string* from "punycode" (RFC 3492). If *$charset* is present, result is converted to *$charset*, otherwise it is converted to ISO−8859−1. Returns result of conversion or **undef** on error.

Net::LibIDN::idn_prep_name(*$string*, [*$charset*]);
Net::LibIDN::idn_prep_kerberos5(*$string*, [*$charset*]);
Net::LibIDN::idn_prep_node(*$string*, [*$charset*]);

Net::LibIDN::idn_prep_resource(*$string*, [*$charset*]);
Net::LibIDN::idn_prep_plain(*$string*, [*$charset*]);
Net::LibIDN::idn_prep_trace(*$string*, [*$charset*]);
Net::LibIDN::idn_prep_sasl(*$string*, [*$charset*]);
Net::LibIDN::idn_prep_iscsi(*$string*, [*$charset*]);

Performs "stringprep" (RFC 3454) on $string according to the named profile (e.g. *_name –> "nameprep" (RFC 3491)). If *$charset* is present, converts from and to this charset before and after the operation respectively. Returns result string, or **undef** on error.

Net::LibIDN::tdl_check(*$string*, *$errpos*, [*$charset*, [*$tld*]]);

Checks whether or not *$string* conforms to the restrictions on the sets of valid characters defined by TLD authorities around the World. Treats *$string* as a hostname if *$tld* is not present, determining the TLD from the hostname. If *$tld* is present, uses the restrictions defined by the parties responsible for TLD *$tld*. *$charset* may be used to specify the character set the *$string* is in. Should an invalid character be detected, returns 0 and the 0–based position of the offending character in *$errpos*. In case of other failure conditions, *$errpos* is not touched, and **undef** is returned. Should *$string* conform to the TLD restrictions, 1 is returned.

Net::LibIDN::tld_get(*$hostname*);

Returns top level domain of *$hostname*, or **undef** if an error occurs or if no top level domain was found.

Net::LibIDN::tld_get_table(*$tld*);

Retrieves a hash reference with the TLD restriction info of given TLD *$tld*, or **undef** if *$tld* is not found. The hash ref contains the following fields:

- *$h–>{name}* ... name of TLD
- *$h–>{version}* ... version string of this restriction table
- *$h–>{nvalid}* ... number of Unicode intervals
- *$h–>{valid}* ... [{*start* => number, *end* => number}, ...] ... Unicode intervals

Limitations

There is currently no support for Perl's unicode capabilities (man perlunicode). All input strings are assumed to be octet strings, all output strings are generated as octet strings. Thus, if you require Perl's unicode features, you will have to convert your strings manually. For example:

```
use Encode;

use Data::Dumper;

print Dumper(Net::LibIDN::idn_to_unicode('xn—uro–j50a.com', 'utf–8'));

print Dumper(decode('utf–8', Net::LibIDN::idn_to_unicode('xn—uro–j50a.com', 'utf–8')));
```

AUTHOR

Thomas Jacob, http://internet24.de

SEE ALSO

perl (1), RFC 3454, RFC 3490–3492, http://www.gnu.org/software/libidn.