

COLLABORATORS			
	TITLE : GNU Libidn API Refere	ence Manual	
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		July 22, 2021	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1	GNU	J Libidn API Reference Manual	1
	1.1	idna.h	2
	1.2	stringprep.h	15
	1.3	punycode.h	38
	1.4	pr29.h	44
	1.5	$tld.h \ \dots $	48
	1.6	idn-free.h	57
2	Inde	·x	59

List of Figures

1.1	Components of Libidn	 2

Chapter 1

GNU Libidn API Reference Manual

GNU Libidn is a fully documented implementation of the Stringprep, Punycode and IDNA specifications. Libidn's purpose is to encode and decode internationalized domain name strings. There are native C, C# and Java libraries.

The C library contains a generic Stringprep implementation. Profiles for Nameprep, iSCSI, SASL, XMPP and Kerberos V5 are included. Punycode and ASCII Compatible Encoding (ACE) via IDNA are supported. A mechanism to define Top-Level Domain (TLD) specific validation tables, and to compare strings against those tables, is included. Default tables for some TLDs are also included.

The Stringprep API consists of two main functions, one for converting data from the system's native representation into UTF-8, and one function to perform the Stringprep processing. Adding a new Stringprep profile for your application within the API is straightforward. The Punycode API consists of one encoding function and one decoding function. The IDNA API consists of the ToASCII and ToUnicode functions, as well as an high-level interface for converting entire domain names to and from the ACE encoded form. The TLD API consists of one set of functions to extract the TLD name from a domain string, one set of functions to locate the proper TLD table to use based on the TLD name, and core functions to validate a string against a TLD table, and some utility wrappers to perform all the steps in one call.

The library is used by, e.g., GNU SASL and Shishi to process user names and passwords. Libidn can be built into GNU Libc to enable a new system-wide getaddrinfo flag for IDN processing.

Libidn is developed for the GNU/Linux system, but runs on over 20 Unix platforms (including Solaris, IRIX, AIX, and Tru64) and Windows. The library is written in C and (parts of) the API is also accessible from C++, Emacs Lisp, Python and Java. A native Java and C# port is included.

Also included is a command line tool, several self tests, code examples, and more.

The internal layout of the library, and how your application interact with the various parts of the library, are shown in Figure 1.1.

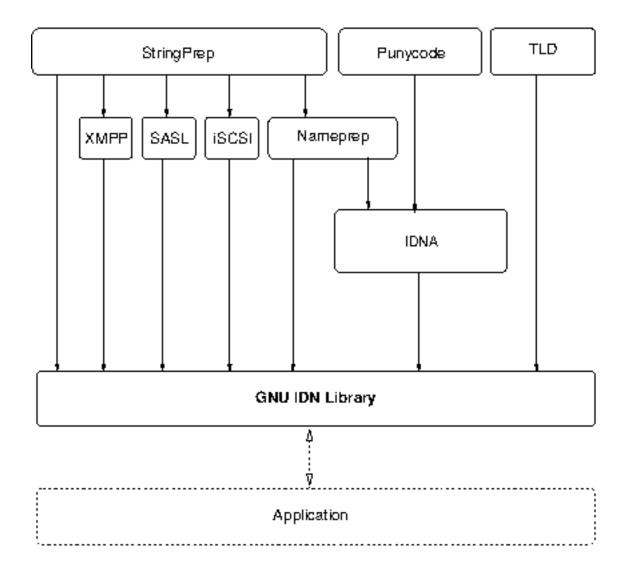


Figure 1.1: Components of Libidn

1.1 idna.h

idna.h — IDNA-related functions

Functions

const char *	idna_strerror ()
int	idna_to_ascii_4i ()
int	idna_to_unicode_44i ()
int	idna_to_ascii_4z ()
int	idna_to_ascii_8z ()
int int	idna_to_ascii_lz ()
int	idna_to_unicode_4z4z ()
int	idna_to_unicode_8z4z ()
int	idna_to_unicode_8z8z ()
int	idna_to_unicode_8zlz ()
int	idna_to_unicode_lzlz ()

Types and Values

#define	IDNAPI
enum	Idna_rc
enum	Idna_flags
#define	IDNA ACE PREFIX

Description

IDNA-related functions.

Functions

idna strerror ()

```
const char~*
idna_strerror (Idna_rc rc);
```

Convert a return code integer to a text string. This string can be used to output a diagnostic message to the user.

IDNA_SUCCESS: Successful operation. This value is guaranteed to always be zero, the remaining ones are only guaranteed to hold non-zero values, for logical comparison purposes. IDNA_STRINGPREP_ERROR: Error during string preparation. IDNA_PUNYCODE_ERROR: Error during punycode operation. IDNA_CONTAINS_NON_LDH: For IDNA_USE_STD3_ASCII_RU indicate that the string contains non-LDH ASCII characters. IDNA_CONTAINS_MINUS: For IDNA_USE_STD3_ASCII_RULES, indicate that the string contains a leading or trailing hyphen-minus (U+002D). IDNA_INVALID_LENGTH: The final output string is not within the (inclusive) range 1 to 63 characters. IDNA_NO_ACE_PREFIX: The string does not contain the ACE prefix (for ToUnicode). IDNA_ROUNDTRIP_VERIFY_ERROR: The ToASCII operation on output string does not equal the input. IDNA_CONTAINS_ACE_PREFIX: The input contains the ACE prefix (for ToASCII). IDNA_ICONV_ERROR: Character encoding conversion error. IDNA_MALLOC_ERROR: Could not allocate buffer (this is typically a fatal error). IDNA_DLOPEN_ERROR: Could not dlopen the libcidn DSO (only used internally in libc).

Parameters

rc an Idna_rc return code.

Returns

Returns a pointer to a statically allocated string containing a description of the error with the return code rc.

idna_to_ascii_4i ()

The ToASCII operation takes a sequence of Unicode code points that make up one domain label and transforms it into a sequence of code points in the ASCII range (0..7F). If ToASCII succeeds, the original sequence and the resulting sequence are equivalent labels.

It is important to note that the ToASCII operation can fail. ToASCII fails if any step of it fails. If any step of the ToASCII operation fails on any label in a domain name, that domain name MUST NOT be used as an internationalized domain name. The method for deadling with this failure is application-specific.

The inputs to ToASCII are a sequence of code points, the AllowUnassigned flag, and the UseSTD3ASCIIRules flag. The output of ToASCII is either a sequence of ASCII code points or a failure condition.

ToASCII never alters a sequence of code points that are all in the ASCII range to begin with (although it could fail). Applying the ToASCII operation multiple times has exactly the same effect as applying it just once.

Parameters

in	input array with unicode
III	code points.
inlen	length of input array with
men	unicode code points.
	output zero terminated
out	string that must have room
out	for at least 63 characters
	plus the terminating zero.
	an Idna_flags value, e.g.,
flags	IDNA_ALLOW_UNASSIGNED
nags	or
	IDNA_USE_STD3_ASCII_RULES.

Returns

Returns 0 on success, or an Idna_rc error code.

idna to unicode 44i ()

The ToUnicode operation takes a sequence of Unicode code points that make up one domain label and returns a sequence of Unicode code points. If the input sequence is a label in ACE form, then the result is an equivalent internationalized label that is not in ACE form, otherwise the original sequence is returned unaltered.

ToUnicode never fails. If any step fails, then the original input sequence is returned immediately in that step.

The Punycode decoder can never output more code points than it inputs, but Nameprep can, and therefore ToUnicode can. Note that the number of octets needed to represent a sequence of code points depends on the particular character encoding used.

The inputs to ToUnicode are a sequence of code points, the AllowUnassigned flag, and the UseSTD3ASCIIRules flag. The output of ToUnicode is always a sequence of Unicode code points.

in	input array with unicode code points.	
inlen	length of input array with unicode code points.	
out	output array with unicode code points.	

	on input, maximum size of
	output array with unicode
outlen	code points, on exit, actual
	size of output array with
	unicode code points.
	an Idna_flags value, e.g.,
flags	IDNA_ALLOW_UNASSIGNED
nags	or
	IDNA_USE_STD3_ASCII_RULES.

Returns Idna_rc error condition, but it must only be used for debugging purposes. The output buffer is always guaranteed to contain the correct data according to the specification (sans malloc induced errors). NB! This means that you normally ignore the return code from this function, as checking it means breaking the standard.

idna_to_ascii_4z()

Convert UCS-4 domain name to ASCII string. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

Parameters

innut	zero terminated input	
input	Unicode string.	
output	pointer to newly allocated	
Output	output string.	
	an Idna_flags value, e.g.,	
flogs	IDNA_ALLOW_UNASSIGN	ED
flags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns

Returns IDNA_SUCCESS on success, or error code.

idna_to_ascii_8z()

Convert UTF-8 domain name to ASCII string. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

input	zero terminated input	
mput	UTF-8 string.	
output	pointer to newly allocated	
output	output string.	
	an Idna_flags value, e.g.,	
flags	IDNA_ALLOW_UNASSIGN	ED
nags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns IDNA_SUCCESS on success, or error code.

idna_to_ascii_lz ()

Convert domain name in the locale's encoding to ASCII string. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

Parameters

	zero terminated input string	
input	encoded in the current	
	locale's character set.	
output	pointer to newly allocated	
Output	output string.	
	an Idna_flags value, e.g.,	
flags	IDNA_ALLOW_UNASSIGN	ED
nags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns

Returns **IDNA_SUCCESS** on success, or error code.

idna_to_unicode_4z4z ()

Convert possibly ACE encoded domain name in UCS-4 format into a UCS-4 string. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

input	zero-terminated Unicode	
mput	string.	
output	pointer to newly allocated	
output	output Unicode string.	
	an Idna_flags value, e.g.,	
flogs	IDNA_ALLOW_UNASSIGN	ED
flags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns IDNA_SUCCESS on success, or error code.

idna_to_unicode_8z4z ()

Convert possibly ACE encoded domain name in UTF-8 format into a UCS-4 string. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

Parameters

innut	zero-terminated UTF-8	
input	string.	
output	pointer to newly allocated	
output	output Unicode string.	
	an Idna_flags value, e.g.,	
flags	IDNA_ALLOW_UNASSIGN	ED
flags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns

Returns **IDNA_SUCCESS** on success, or error code.

idna_to_unicode_8z8z ()

Convert possibly ACE encoded domain name in UTF-8 format into a UTF-8 string. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

input	zero-terminated UTF-8	
input	string.	

output	pointer to newly allocated	
output	output UTF-8 string.	
	an Idna_flags value, e.g.,	
flogs	IDNA_ALLOW_UNASSIGN	ED
flags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns **IDNA_SUCCESS** on success, or error code.

idna_to_unicode_8zlz ()

Convert possibly ACE encoded domain name in UTF-8 format into a string encoded in the current locale's character set. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

Parameters

inpu	input	zero-terminated UTF-8	
	прис	string.	
		pointer to newly allocated	
output	output	output string encoded in the	
	Juiput	current locale's character	
		set.	
fla		an Idna_flags value, e.g.,	
	flags	IDNA_ALLOW_UNASSIGN	ED
	iags	or	
		IDNA_USE_STD3_ASCII_R	ULES.

Returns

Returns **IDNA_SUCCESS** on success, or error code.

idna_to_unicode_lzlz ()

Convert possibly ACE encoded domain name in the locale's character set into a string encoded in the current locale's character set. The domain name may contain several labels, separated by dots. The output buffer must be deallocated by the caller.

	zero-terminated string	
input	encoded in the current	
	locale's character set.	
	pointer to newly allocated	
output	output string encoded in the	
output	current locale's character	
	set.	
	an Idna_flags value, e.g.,	
flags	IDNA_ALLOW_UNASSIGN	ED
nags	or	
	IDNA_USE_STD3_ASCII_R	ULES.

Returns IDNA_SUCCESS on success, or error code.

Types and Values

IDNAPI

Symbol holding shared library API visibility decorator.

This is used internally by the library header file and should never be used or modified by the application.

 $https://www.gnu.org/software/gnulib/manual/html_node/Exported-Symbols-of-Shared-Libraries.html\\$

enum Idna_rc

Enumerated return codes of idna_to_ascii_4i(), idna_to_unicode_44i() functions (and functions derived from those functions). The value 0 is guaranteed to always correspond to success.

Members

	Successful
	op-
	et-
	a-
	tion.
	This
	value
	is
	guar-
	an-
	teed
	to
	al-
	ways
	be
	zero,
	the
	re-
	main-
	ing
IDNA_SUCCESS	ones
	are
	only
	guar-
	an-
	teed
	to
	hold
	non-
	zero
	val-
	ues,
	for
	log-
	i-
	cal
	com-
	par-
	i-
	søn
	pur-
	poses.
	Error
	dur-
	ing
IDNA_STRINGPREP_ERROR	ing string
IDIM_GIMINOI KEI _EKKOK	Sumg
	prepa-
	ra- tion.
	tion.

	Error
	dur-
	ing
	pun-
	y
IDNA_PUNYCODE_ERROR	
	code
	op-
	er-
	a-
	tion.
	For
	IDNA_USE_STD3_ASCII_RULES,
	in-
	di-
	cate
	that
	the
	string
IDNA_CONTAINS_NON_LDH	
	con-
	tains
	non-
	LDH
	ASCII
	char-
	a¢-
	ters.
	Same
	a\$
	IDNA_CONTAINS_NON_LDH
	for
	com-
	pat-
	i-
IDNA_CONTAINS_LDH	bil-
	ity
	with
	typo
	in
	ear-
	lier
	ver-
	sions.

IDNA_CONTAINS_MINUS	For IDNA_USE_STD3_ASCII_RULES, in- di- cate that the string con- tains a lead- ing or trail- ing hyphen- minus (U+002D).
IDNA_INVALID_LENGTH	The fil- nal out- put string is not within the (in- clu- sive) range 1 to 63 charac- ters.
IDNA_NO_ACE_PREFIX	The string does not contain the ACE pre-fix (for ToUni-code).

IDNA_ROUNDTRIP_VERIFY_ERROR	The ToASCII op- er- a- tion on out- put string does not equal the in- put.
IDNA_CONTAINS_ACE_PREFIX	The in- put con- tains the ACE pre- fix (for ToASCII).
IDNA_ICONV_ERROR	Character en- cod- ing con- ver- sion er- rør.
IDNA_MALLOC_ERROR	Could not al- lo- cate buffer (this is typ- i- cally a fa- tal er- ror).

	Could
	not
	dlopen
	the
	lib-
	cidn
IDMA DI ODEM EDDOD	DSO
IDNA_DLOPEN_ERROR	(only
	used
	in-
	ter-
	nally
	in
	libc).

enum Idna_flags

Flags to pass to idna_to_ascii_4i(), idna_to_unicode_44i() etc.

Members

IDNA_ALLOW_UNASSIGNED IDNA_ALLOW_UNASSIGNED IDNA_ALLOW_UNASSIGNED IDNA_UNASSIGNED IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES		Don't
IDNA_ALLOW_UNASSIGNED IDNA_ALLOW_UNASSIGNED ing unas- signed Uni- code code code points. Validate strings ac- cord- ing to STD3 rules (i,e., nor- mal host name		
IDNA_ALLOW_UNASSIGNED contain- ing unas- signed Uni- code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		ject
IDNA_ALLOW_UNASSIGNED tain- ing unas- signed Uni- code code points. Validate strings ac- cord- ing to IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES tain- ing unas- signed Uni- code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		strings
IDNA_ALLOW_UNASSIGNED ing unas- signed Uni- code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		con-
unas- signed Uni- code code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		tain-
unas- signed Uni- code code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name	IDNA_ALLOW_UNASSIGNED	ing
Uni- code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		
code code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		signed
code points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		Uni-
points. Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		code
Validate strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		
strings ac- cord- ing to STD3 rules (i.e., nor- mal host name		points.
IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3 IDNA_USE_STD3		
IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3_ASCII_RULES IDNA_USE_STD3 IDNA_USE_STD3		strings
IDNA_USE_STD3_ASCII_RULES STD3 rules (i.e., nor- mal host name		ac-
IDNA_USE_STD3_ASCII_RULES to STD3 rules (i.e., nor- mal host name		cord-
IDNA_USE_STD3_ASCII_RULES to STD3 rules (i.e., nor- mal host name		ing
rules (i.e., nor- mal host name		
(i.e., nor- mal host name	IDMA LISE STD2 ASCII DIJI ES	
nor- mal host name	IDNA_USE_STDS_ASCII_RULES	rules
mal host name		(i.e.,
host name		nor-
name		mal
		host
rules).		name
		rules).

IDNA_ACE_PREFIX

```
# define IDNA_ACE_PREFIX "xn--"
```

The IANA allocated prefix to use for IDNA. "xn--"

1.2 stringprep.h

stringprep.h — Stringprep-related functions

Functions

#define	stringprep_nameprep()
#define	stringprep_nameprep_no_unassigned()
#define	stringprep_plain()
#define	stringprep_kerberos5()
#define	stringprep_xmpp_nodeprep()
#define	stringprep_xmpp_resourceprep()
#define	stringprep_iscsi()
int	stringprep_4i ()
int	stringprep_4zi ()
int	stringprep ()
int	stringprep_profile ()
const char *	stringprep_strerror ()
const char *	stringprep_check_version ()
int	stringprep_unichar_to_utf8 ()
uint32_t	stringprep_utf8_to_unichar ()
uint32_t *	stringprep_utf8_to_ucs4 ()
char *	stringprep_ucs4_to_utf8 ()
char *	stringprep_utf8_nfkc_normalize ()
uint32_t *	stringprep_ucs4_nfkc_normalize ()
const char *	stringprep_locale_charset ()
char *	stringprep_convert ()
char *	stringprep_locale_to_utf8 ()
char *	stringprep_utf8_to_locale ()

Types and Values

#define	IDNAPI
#define	STRINGPREP_VERSION
enum	Stringprep_rc
enum	Stringprep_profile_flags
enum	Stringprep_profile_steps
#define	STRINGPREP_MAX_MAP_CHARS
struct	Stringprep_table_element
struct	Stringprep_table
typedef	Stringprep_profile
struct	Stringprep_profiles

Description

Stringprep-related functions.

Functions

stringprep_nameprep()

```
#define stringprep_nameprep(in, maxlen)
```

Prepare the input UTF-8 string according to the nameprep profile. The AllowUnassigned flag is true, use stringprep_nameprep_no_unass if you want a false AllowUnassigned. Returns 0 iff successful, or an error code.

Parameters

in	input/ouput array with	
	string to prepare.	
moxlan	maximum length of	
maxlen	input/output array.	

stringprep_nameprep_no_unassigned()

```
#define stringprep_nameprep_no_unassigned(in, maxlen)
```

Prepare the input UTF-8 string according to the nameprep profile. The AllowUnassigned flag is false, use stringprep_nameprep() for true AllowUnassigned. Returns 0 iff successful, or an error code.

Parameters

in	input/ouput array with	
III	string to prepare.	
maxlen	maximum length of	
maxicii	input/output array.	

stringprep_plain()

```
#define stringprep_plain(in, maxlen)
```

Prepare the input UTF-8 string according to the draft SASL ANONYMOUS profile. Returns 0 iff successful, or an error code.

Parameters

in	input/ouput array with	
111	string to prepare.	
moxlan	maximum length of	
maxlen	input/output array.	

stringprep_kerberos5()

```
#define stringprep_kerberos5(in, maxlen)
```

Prepare the input UTF-8 string according to the draft Kerberos 5 node identifier profile. Returns 0 iff successful, or an error code.

in	input/ouput array with	
	string to prepare.	
maxlen	maximum length of	
maxien	input/output array.	

stringprep_xmpp_nodeprep()

```
#define stringprep_xmpp_nodeprep(in, maxlen)
```

Prepare the input UTF-8 string according to the draft XMPP node identifier profile. Returns 0 iff successful, or an error code.

Parameters

in	input/ouput array with	
III	string to prepare.	
maylan	maximum length of	
maxlen	input/output array.	

stringprep_xmpp_resourceprep()

```
#define stringprep_xmpp_resourceprep(in, maxlen)
```

Prepare the input UTF-8 string according to the draft XMPP resource identifier profile. Returns 0 iff successful, or an error code.

Parameters

in	input/ouput array with	
ın	string to prepare.	
maylan	maximum length of	
maxlen	input/output array.	

stringprep_iscsi()

```
#define stringprep_iscsi(in, maxlen)
```

Prepare the input UTF-8 string according to the draft iSCSI stringprep profile. Returns 0 iff successful, or an error code.

Parameters

in	input/ouput array with string to prepare.	
movlan	maximum length of	
maxlen	input/output array.	

stringprep_4i()

Prepare the input UCS-4 string according to the stringprep profile, and write back the result to the input string.

The input is not required to be zero terminated (ucs4 [len] = 0). The output will not be zero terminated unless ucs4 [len] = 0. Instead, see stringprep_4zi() if your input is zero terminated or if you want the output to be.

Since the stringprep operation can expand the string, maxucs4len indicate how large the buffer holding the string is. This function will not read or write to code points outside that size.

The flags are one of Stringprep_profile_flags values, or 0.

The *profile* contain the Stringprep_profile instructions to perform. Your application can define new profiles, possibly re-using the generic stringprep tables that always will be part of the library, or use one of the currently supported profiles.

Parameters

ucs4	input/output array with	
ucs4	string to prepare.	
	on input, length of input	
	array with Unicode code	
len	points, on exit, length of	
	output array with Unicode	
	code points.	
maxucs4len	maximum length of	
maxues4ien	input/output array.	
flags	a Stringprep_profile_flags	
nags	value, or 0.	
profile	pointer to	
prome	Stringprep_profile to use.	

Returns

Returns STRINGPREP_OK iff successful, or an Stringprep_rc error code.

stringprep_4zi ()

Prepare the input zero terminated UCS-4 string according to the stringprep profile, and write back the result to the input string.

Since the stringprep operation can expand the string, maxucs41en indicate how large the buffer holding the string is. This function will not read or write to code points outside that size.

The flags are one of Stringprep_profile_flags values, or 0.

The profile contain the Stringprep_profile instructions to perform. Your application can define new profiles, possibly re-using the generic stringprep tables that always will be part of the library, or use one of the currently supported profiles.

ucs4	input/output array with zero	
ucs4	terminated string to prepare.	
maxucs4len	maximum length of	
maxues+ien	input/output array.	
flags	a Stringprep_profile_flags	
nags	value, or 0.	
profile	pointer to	
prome	Stringprep_profile to use.	

Returns STRINGPREP_OK iff successful, or an Stringprep_rc error code.

stringprep ()

Prepare the input zero terminated UTF-8 string according to the stringprep profile, and write back the result to the input string.

Note that you must convert strings entered in the systems locale into UTF-8 before using this function, see stringprep_locale_to_utf8().

Since the stringprep operation can expand the string, maxlen indicate how large the buffer holding the string is. This function will not read or write to characters outside that size.

The flags are one of Stringprep_profile_flags values, or 0.

The *profile* contain the Stringprep_profile instructions to perform. Your application can define new profiles, possibly re-using the generic stringprep tables that always will be part of the library, or use one of the currently supported profiles.

Parameters

in	input/ouput array with	
III	string to prepare.	
maxlen	maximum length of	
maxicii	input/output array.	
flags	a Stringprep_profile_flags	
nags	value, or 0.	
profile	pointer to	
proffic	Stringprep_profile to use.	

Returns

Returns STRINGPREP_OK iff successful, or an error code.

stringprep_profile ()

Prepare the input zero terminated UTF-8 string according to the stringprep profile, and return the result in a newly allocated variable.

Note that you must convert strings entered in the systems locale into UTF-8 before using this function, see stringprep locale to utf8().

The output out variable must be deallocated by the caller.

The *flags* are one of Stringprep_profile_flags values, or 0.

The profile specifies the name of the stringprep profile to use. It must be one of the internally supported stringprep profiles.

in	input array with UTF-8	
	string to prepare.	
out	output variable with pointer	
out	to newly allocate string.	
profile	name of stringprep profile	
prome	to use.	
flags	a Stringprep_profile_flags	
nags	value, or 0.	

Returns **STRINGPREP_OK** iff successful, or an error code.

stringprep_strerror()

```
const char~*
stringprep_strerror (Stringprep_rc rc);
```

Convert a return code integer to a text string. This string can be used to output a diagnostic message to the user.

STRINGPREP_OK: Successful operation. This value is guaranteed to always be zero, the remaining ones are only guaranteed to hold non-zero values, for logical comparison purposes. STRINGPREP_CONTAINS_UNASSIGNED: String contain unassigned Unicode code points, which is forbidden by the profile. STRINGPREP_CONTAINS_PROHIBITED: String contain code points prohibited by the profile. STRINGPREP_BIDI_BOTH_L_AND_RAL: String contain code points with conflicting bidirection category. STRINGPREP_BIDI_LEADTRAIL_NOT_RAL: Leading and trailing character in string not of proper bidirectional category. STRINGPREP_BIDI_CONTAINS_PROHIBITED: Contains prohibited code points detected by bidirectional code. STRINGPREP_TOO_SMALL_BUFFER: Buffer handed to function was too small. This usually indicate a problem in the calling application. STRINGPREP_PROFILE_ERROR: The stringprep profile was inconsistent. This usually indicate an internal error in the library. STRINGPREP_FLAG_ERROR: The supplied flag conflicted with profile. This usually indicate a problem in the calling application. STRINGPREP_UNKNOWN_PROFILE: The supplied profile name was not known to the library. STRINGPREP_ICONV_ERROR: Character encoding conversion error. STRINGPREP_NFKC_FAILED: The Unicode NFKC operation failed. This usually indicate an internal error in the library. STRINGPREP_MALLOC_ERROR: The malloc() was out of memory. This is usually a fatal error.

Parameters

rc a Stringprep_rc return code.

Returns

Returns a pointer to a statically allocated string containing a description of the error with the return code rc.

stringprep_check_version ()

```
const char~*
stringprep_check_version (const char *req_version);
```

Check that the version of the library is at minimum the requested one and return the version string; return NULL if the condition is not satisfied. If a NULL is passed to this function, no check is done, but the version string is simply returned.

See STRINGPREP VERSION for a suitable req_version string.

req_version Required or NU	ed version number, LL.
------------------------------	---------------------------

Version string of run-time library, or NULL if the run-time library does not meet the required version number.

stringprep_unichar_to_utf8 ()

Converts a single character to UTF-8.

Parameters

c	a ISO10646 character code	
	output buffer, must have at	
	least 6 bytes of space. If	
. d. C	NULL, the length will be	
outbuf	computed and returned and	
	nothing will be written to	
	outbuf.	

Returns

number of bytes written.

stringprep_utf8_to_unichar ()

```
uint32_t
stringprep_utf8_to_unichar (const char *p);
```

Converts a sequence of bytes encoded as UTF-8 to a Unicode character. If p does not point to a valid UTF-8 encoded character, results are undefined.

Parameters

n	a pointer to Unicode
P	character encoded as UTF-8

Returns

the resulting character.

stringprep_utf8_to_ucs4 ()

Convert a string from UTF-8 to a 32-bit fixed width representation as UCS-4. The function now performs error checking to verify that the input is valid UTF-8 (before it was documented to not do error checking).

Parameters

str	a UTF-8 encoded string	
	the maximum length of str	
len	to use. If $len < 0$, then the	
	string is nul-terminated.	
	location to store the number	
items_written	of characters in the result,	
	or NULL.	

Returns

a pointer to a newly allocated UCS-4 string. This value must be deallocated by the caller.

stringprep_ucs4_to_utf8 ()

Convert a string from a 32-bit fixed width representation as UCS-4. to UTF-8. The result will be terminated with a 0 byte.

Parameters

str	a UCS-4 encoded string	
	the maximum length of str	
len	to use. If $len < 0$, then the	
len	string is terminated with a 0	
	character.	
	location to store number of	
items_read	characters read read, or	
	NULL.	
	location to store number of	
items_written	bytes written or NULL. The	
	value here stored does not	
	include the trailing 0 byte.	

Returns

a pointer to a newly allocated UTF-8 string. This value must be deallocated by the caller. If an error occurs, NULL will be returned.

stringprep_utf8_nfkc_normalize ()

Converts a string into canonical form, standardizing such issues as whether a character with an accent is represented as a base character and combining accent or as a single precomposed character.

The normalization mode is NFKC (ALL COMPOSE). It standardizes differences that do not affect the text content, such as the above-mentioned accent representation. It standardizes the "compatibility" characters in Unicode, such as SUPERSCRIPT THREE to the standard forms (in this case DIGIT THREE). Formatting information may be lost but for most text operations such characters should be considered the same. It returns a result with composed forms rather than a maximally decomposed form.

Parameters

str	a UTF-8 encoded string.	
len	length of str, in bytes, or -1 if str is nul-terminated.	

Returns

a newly allocated string, that is the NFKC normalized form of str.

stringprep_ucs4_nfkc_normalize ()

Converts a UCS4 string into canonical form, see stringprep_utf8_nfkc_normalize() for more information.

Parameters

str	a Unicode string.	
lan	length of str array, or -1 if	
len	str is nul-terminated.	

Returns

a newly allocated Unicode string, that is the NFKC normalized form of str.

stringprep_locale_charset ()

```
const char~*
stringprep_locale_charset (void);
```

Find out current locale charset. The function respect the CHARSET environment variable, but typically uses nl_langinfo(CODESET) when it is supported. It fall back on "ASCII" if CHARSET isn't set and nl_langinfo isn't supported or return anything.

Note that this function return the application's locale's preferred charset (or thread's locale's preferred charset, if your system support thread-specific locales). It does not return what the system may be using. Thus, if you receive data from external sources you cannot in general use this function to guess what charset it is encoded in. Use stringprep_convert from the external representation into the charset returned by this function, to have data in the locale encoding.

Returns

Return the character set used by the current locale. It will never return NULL, but use "ASCII" as a fallback.

stringprep_convert ()

Convert the string from one character set to another using the system's iconv() function.

Parameters

str	input zero-terminated	
	string.	
to codeset	name of destination	
to_codeset	character set.	
from_codeset	name of origin character	
nom_codeset	set, as used by str.	

Returns

Returns newly allocated zero-terminated string which is str transcoded into to_codeset.

stringprep_locale_to_utf8 ()

```
char~*
stringprep_locale_to_utf8 (const char *str);
```

Convert string encoded in the locale's character set into UTF-8 by using stringprep_convert().

Parameters

str input zero terminated string.

Returns

Returns newly allocated zero-terminated string which is str transcoded into UTF-8.

stringprep_utf8_to_locale ()

```
char~*
stringprep_utf8_to_locale (const char *str);
```

Convert string encoded in UTF-8 into the locale's character set by using stringprep_convert().

Parameters

str input zero terminated string.

Returns

Returns newly allocated zero-terminated string which is str transcoded into the locale's character set.

Types and Values

IDNAPI

#define IDNAPI

Symbol holding shared library API visibility decorator.

This is used internally by the library header file and should never be used or modified by the application.

 $https://www.gnu.org/software/gnulib/manual/html_node/Exported-Symbols-of-Shared-Libraries.html\\$

STRINGPREP_VERSION

define STRINGPREP_VERSION "1.38"

String defined via CPP denoting the header file version number. Used together with stringprep_check_version() to verify header file and run-time library consistency.

enum Stringprep_rc

Enumerated return codes of stringprep(), stringprep_profile() functions (and macros using those functions). The value 0 is guaranteed to always correspond to success.

Members

	Successful
	op-
	er-
	a
	tion.
	This
	value
	is
	guar-
	an-
	teed
	to
	al-
	ways
	be
	zero,
	the
	re-
	main-
	ing
STRINGPREP_OK	ones
	are
	only
	guar-
	an-
	teed
	to
	hold
	non-
	zero
	val-
	ues,
	for
	log-
	i-
	cal
	com-
	nar-
	par- i-
	son
	pur-
	poses.

String con- tain unas- signed
Uni- code code points, which is for- bid- den
by the pro- file.
String con- tain code points
pro- hib- ited by
the profile. String
con- tain code points
with con- flict- ing bidi-
rec- tion cat- e- gory.

Leading and trial Heading and trial Heading all trial High char age Heading		
stringprep_BIDL_LEADTRAIL_NOT_RAL STRINGPREP_BIDL_LEADTRAIL_NOT_RAL STRINGPREP_BIDL_LEADTRAIL_NOT_RAL STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_C		Leading
### Thin		
ing character ter in a string string not of proper bidirectional category. STRINGPREP_BIDL_LEADTRAIL_NOT_RAL STRINGPREP_BIDL_CANTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED Geode. Buffer hunded to to function was too small. This usually in- tion category in- tion the category in- tin the categor		
char- ac- ter in string string nbt of proper hidi- rec- tional cat- el- gbry. Contains pro- hib- injed code phints de- tected by bidi- rec- tional cate- el- gbry. STRINGPREP_BIDI_CONTAINS_PROHIBITED Buffer hinded to func- tional code. STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER Call- ing al prob- lem in tit tthe call- ing ap- pi- cate- cate		trail-
char- ac- ter in string string nbt of proper hidi- rec- tional cat- el- gbry. Contains pro- hib- injed code phints de- tected by bidi- rec- tional cate- el- gbry. STRINGPREP_BIDI_CONTAINS_PROHIBITED Buffer hinded to func- tional code. STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER Call- ing al prob- lem in tit tthe call- ing ap- pi- cate- cate		ing
STRINGPREP_BIDI_LEADTRAIL_NOT_RAL STRINGPREP_BIDI_LEADTRAIL_NOT_RAL BY Proper bidi- re- tional cat- cat- c- pory. Contains pro- hib- itied code points de- tected by bidi- re- fibral code. Buffer handed to fine- tion was too small, This usu- ally int- STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER Acte all ing ap- pib- cat- cat- cat- cat- cat- cat- cat- cat		char
STRINGPREP_BIDI_LEADTRAIL_NOT_RAL String not of proper bidding rectional cate elegible. STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED Bullet rected by bidding rectional code. Bullet handed to fine-tional code. STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER Cate a probbiding in the calling appopries		
STRINGPREP_BIDI_LEADTRAIL_NOT_RAI. string mot of proper bidirectional called c		
STRINGPREP_BIDL_LEADTRAIL_NOT_RAIL String npt n		ter
STRINGPREP_BIDL_LEADTRAIL_NOT_RAIL String npt of proper bidi- rec- ional cat- cat-		in
STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_		
of proper bidi- pr	STRINGPREP_BIDI_LEADTRAIL_NOT_RAL	
proper bidi- rec- tional cat- e- gory. Contains pro- hib- ited code points de- tected by by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER d- cate a prob- lem in the call- ing ap- pii- ca-		not
bidirectional cate epgory. Contains prohibited code points detected by bidirectional code. STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED Example 1		of
bidirectional cate epgory. Contains prohibited code points detected by bidirectional code. STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED Example 1		proper
rectional cate cpt cpm. cate cpt cpm. Contains prohib- lited code points de- lected by bidi- rectional code. Buffer handed to fine- lite inc to small. This lite lite lite lite lite lite lite lite		bidi-
tional cat- ei ej gory. Contains pro- hib- ited code points de- teted by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- ppli- cai-		
cat- e gory. Contains pro- hib- ited code points STRINGPREP_BIDL_CONTAINS_PROHIBITED STRINGPREP_BIDL_CONTAINS_PROHIBITED Buffer handed to func- tional code. Buffer handed to func- tion was too shaall. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER Cate a prob- lem in the cate cate cate cate a prob- lem in the cate cate cate cate cate a prob- lem in the cate cate cate cate cate cate cate cat		
c- gbry. Contains Pro- hib- itged code points de- tected by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pii- cal-		
gbry. Contains pro- hib- itted code points de- tected by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- Cate a prob- lem in the catl- in the catl- in the catl- in gap- pli- ca-		cat-
gbry. Contains pro- hib- itted code points de- tected by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- Cate a prob- lem in the catl- in the catl- in the catl- in gap- pli- ca-		e e e e e e e e e e e e e e e e e e e
Contains pro- hib- ited code points de- tected by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem if the call- ing ap- pli- ca-		
STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_		gory.
STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED SUBJECT SUBJECT STRINGPREP_TOO_SMALL_BUFFER		Contains
STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED STRINGPREP_BIDI_CONTAINS_PROHIBITED SUBJECT SUBJECT STRINGPREP_TOO_SMALL_BUFFER		pro-
STRINGPREP_BIDI_CONTAINS_PROHIBITED aby bidirect tional code. Buffer hinded to finnc- tion was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER approblem in the call- ing ap- pli- ca- care ap- pli- ca- care ap- pli- ca- care ap- pli- ca- care		
STRINGPREP_BIDI_CONTAINS_PROHIBITED detected by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally iri- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pil- ca-		
STRINGPREP_BIDI_CONTAINS_PROHIBITED points de- tected by bidi- rec- tipnal code. Buffer handed to func- tipn was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pji- ca-		
STRINGPREP_BIDI_CONTAINS_PROHIBITED detected by bidi- rec- tional code. Buffer handed to func- tion was too shall, This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca- ca- ca- ca- ca- ca- ca- ca- ca- ca		
STRINGPREP_BIDI_CONTAINS_PROHIBITED detected by bidi- rec- tional code. Buffer handed to func- tion was too small, This usu- ally in- di- cate a prob- lem in the cate lem in the the cate lem in the		points
tected by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- stringprep_TOO_SMALL_BUFFER STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-	STRINGPREP RIDI CONTAINS PROHIBITED	de-
by bidi- rec- tional code. Buffer handed to func- tion was too small. This usu- ally in- stringprep_too_small_buffer di- cate a prob- lem in the call- ing ap- pli- ca-	STRINGTREE _BID1_CONTRING_TROTILDITED	
bidirectional code. Buffer handed to function was too small. This usu-ally in-discrete discrete a problem in the calling application.		
rectional code. Buffer handed to func- tion was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pli- ca-		
rectional code. Buffer handed to func- tion was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pli- ca-		bidi-
tional code. Buffer handed to func- tion was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pli- ca-		
code. Buffer handed to func- tion was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER d- cate a prob- lem in the call- ing ap- pli- ca-		
Buffer handed to function was too small. This usu-ally in-di-cate a problem in the call-ing applicate of the cate a ppilicate.		
handed to func- tion was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pil- ca-		
to function was too small. This usu-ally in-di-cate a problem in the call-ing ap-pli-ca-		Buffer
to function was too small. This usu-ally in-di-cate a problem in the call-ing ap-pli-ca-		handed
function was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pli- ca-		
tion was too small. This usu- ally in- di- cate a prob- lem in the call- ing ap- pli- ca-		
was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pii- ca-		
was too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pii- ca-		tion
too small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-		
small. This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-		
This usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-		
usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- Ca-		small.
usu- ally in- STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- Ca-		This
STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-		
STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-		
STRINGPREP_TOO_SMALL_BUFFER di- cate a prob- lem in the call- ing ap- pli- ca-		
cate a prob- lem in the call- ing ap- pli- ca-		
cate a prob- lem in the call- ing ap- pli- ca-	STRINGPREP TOO SMALL BUFFER	di-
a problem in the calling applicate the plicate calling applicate calling applicate.		
prob- lem in the call- ing ap- pli- ca-		
lem in the call- ing ap- pli- ca-		
lem in the call- ing ap- pli- ca-		prob-
in the call- ing ap- pli- ca-		
the call- ing ap- pli- ca-		
call- ing ap- pli- ca-		
ing ap- pli- ca-		
ing ap- pli- ca-		call-
ap- pli- ca-		
pli- ca-		
ca-		ap- 1.
ca-		pli-
tion.		
upn.		tion
		upii.

	The
	string-
	prep
	pro-
	file
	was
	in-
	con-
	sis-
	tent.
	This
	uşu-
STRINGPREP_PROFILE_ERROR	ally
	in-
	di-
	cate
	an
	in-
	ter-
	nal
	er-
	ror
	in
	the
	li-
	brary.
	The
	sup-
	plied
	flag
	con-
	flicted
	with
	pro-
STRINGPREP_FLAG_ERROR	file.
	This
	usu-
	ally
	in-
	di-
	cate
	a
	prob-
	lem
	in
	the
	call-
	ing
	ap-
	pli-
	Ca-
	tion.

	The
	sup-
	plied
	pro-
	file
	name
STRINGPREP_UNKNOWN_PROFILE	was
	not
	known
	to
	the
	li-
	brary.
	Character
	en-
	cod-
	ing
STRINGPREP_ICONV_ERROR	con-
	ver-
	sion
	er-
	ror.
	The
	Uni-
	code
	NFKC
	op-
	er-
	a-
	tion
STRINGPREP_NFKC_FAILED	failed.
	This
	usu-
	ally
	in-
	di-
	cate
	an
	in-
	ter-
	nal
	er-
	ror
	in
	the
	li-
	117
	brary.
	1 1 2

STRINGPREP_MALLOC_ERROR

The mal-loc() was out of memory. This is usually a fatal error.

enum Stringprep_profile_flags

Stringprep profile flags.

Members

Disable the NFKC normalization, as well as se-lect-ing the non-NFKC case folding tables. Usually the profile STRINGPREP_NO_NFKC specifies BIDI and NFKC settings, and applications should not override it unless in speclal situ a tions.

	Disable the BIDI step. Usu- ally the pro- file spec- i-
STRINGPREP_NO_BIDI	set- tings, and ap- pli- ca- tions should not over- ride it un-
	less in spe- cial sit- u- a- tions.

Make the library return with an error if string STRINGPREP_NO_UNASSIGNED contains unassigned chara¢ters a¢cording to profile.

enum Stringprep_profile_steps

Various steps in the stringprep algorithm. You really want to study the source code to understand this one. Only useful if you want to add another profile.

STRINGPREP_NFKC	The NFKC
	step.
	The
STRINGPREP_BIDI	BIDI
	step.
	The
STRINGPREP_MAP_TABLE	MAP
	step.
	The
STRINGPREP_UNASSIGNED_TABLE	Unas-
STRINGTREE _OTT SOIGHED_ITEDEE	signed
	step.
	The
	Pro-
STRINGPREP_PROHIBIT_TABLE	hib-
	ited
	step.
	The
STRINGPREP_BIDI_PROHIBIT_TABLE	BIDI-
oranioral publication in the control of the control and the control of the contro	Prohibited
	step.

STRINGPREP_BIDI_RAL_TABLE	The BIDI- RAL step.
STRINGPREP_BIDI_L_TABLE	The BIDI- L step.

STRINGPREP_MAX_MAP_CHARS

```
# define STRINGPREP_MAX_MAP_CHARS 4
```

Maximum number of code points that can replace a single code point, during stringprep mapping.

struct Stringprep_table_element

```
struct Stringprep_table_element {
   uint32_t start;
   uint32_t end;
   uint32_t map[STRINGPREP_MAX_MAP_CHARS];
};
```

Stringprep profile table element.

Members

uint32_t end; point. ending code- point, 0 if only one char- ac- ter. codepoints to map start into, NULL if end ig end ig not		starting
uint32_t end; ending code- point, 0 if only one char- ac- ter. codepoints to map start into, NULL if end is not	<pre>uint32_t start;</pre>	code-
<pre>uint32_t end;</pre> code- point, 0 if only one char- ac- ter. codepoints to map start into, NULL if end is not		
uint32_t end; point, 0 if only one char- ac- ter. codepoints to map start into, NULL if end is not		
uint32_t end; 0 if only one char- ac- ter. codepoints to map start into, NULL if end is not		
uint32_t end; if only one char- ac- ter. codepoints to map start into, NULL if end is not		point,
uint32_t end; only one char- ac- ter. codepoints to map start into, NULL if end is not		
only one char- ac- ter. codepoints to map start into, NULL if end is not	uint32 t and	
char- ac- ter. codepoints to map start into, NULL if end is not	umoz_t end,	only
acter. codepoints to map start into, NULL if end is not		one
ter. codepoints to map start into, NULL if end is not		char-
codepoints to map start into, NULL if end is not		a¢-
to map start into, NULL if end is not		
<pre>uint32_t map[STRINGPREP_MAX_MAP_CHARS];</pre> map start into, NULL if end is not		codepoints
uint32_t map[STRINGPREP_MAX_MAP_CHARS]; start into, NULL if end is not		to
uint32_t map[STRINGPREP_MAX_MAP_CHARS]; into, NULL if end is not		map
uint32_t map[STRINGPREP_MAX_MAP_CHARS]; NULL if end is not		
if end is not		
end is not	<pre>uint32_t map[STRINGPREP_MAX_MAP_CHARS];</pre>	
is not		if
not		
		is
		not
0		0

struct Stringprep_table

```
struct Stringprep_table {
   Stringprep_profile_steps operation;
   Stringprep_profile_flags flags;
   const Stringprep_table_element *table;
   size_t table_size;
};
```

Stringprep profile table.

Members

Stringprep_profile_steps operation;	a String- prep_profile_steps value
Stringprep_profile_flags flags;	a String- prep_profile_flags value
<pre>const Stringprep_table_element *table;</pre>	zero- terminated ar- ray of String- prep_table_element el- e- ments.
<pre>size_t table_size;</pre>	size of table , to speed up search- ing.

Stringprep_profile

```
typedef struct Stringprep_table Stringprep_profile;
```

Stringprep profile table.

struct Stringprep_profiles

```
struct Stringprep_profiles {
   const char *name;
   const Stringprep_profile *tables;
};
```

Element structure

Members

	name of
const char *name;	string-
114	prep
	pro-
	file.
	zero-
	terminated
	ar-
	ray
const Stringgroup grafile */ -1-1	of
<pre>const Stringprep_profile *tables;</pre>	String-
	prep_profile
	el-
	e e
	ments.

1.3 punycode.h

punycode.h — Punycode-related functions

Functions

const char *	punycode_strerror ()
int	punycode_encode ()
int	punycode decode ()

Types and Values

	#define	IDNAPI
-	enum	Punycode_status
	typedef	punycode_uint

Description

Punycode-related functions.

Functions

punycode_strerror ()

```
const char~*
punycode_strerror (Punycode_status rc);
```

Convert a return code integer to a text string. This string can be used to output a diagnostic message to the user.

PUNYCODE_SUCCESS: Successful operation. This value is guaranteed to always be zero, the remaining ones are only guaranteed to hold non-zero values, for logical comparison purposes. PUNYCODE_BAD_INPUT: Input is invalid. PUNYCODE_BIG_OUTPUT: Output would exceed the space provided. PUNYCODE_OVERFLOW: Input needs wider integers to process.

Parameters

rc an Punycode_status return code.

Returns

Returns a pointer to a statically allocated string containing a description of the error with the return code rc.

punycode_encode ()

Converts a sequence of code points (presumed to be Unicode code points) to Punycode.

	The number of code points
innut lonath	in the input array and the
input_length	number of flags in the
	case_flags array.
	An array of code points.
	They are presumed to be
	Unicode code points, but
	that is not strictly
	REQUIRED. The array
	contains code points, not
	code units. UTF-16 uses
	code units D800 through
inant	DFFF to refer to code
input	points 1000010FFFF. The
	code points D800DFFF do
	not occur in any valid
	Unicode string. The code
	points that can occur in
	Unicode strings (0D7FF
	and E00010FFFF) are also
	called Unicode scalar
	values.

case_flags	A NULL pointer or an array of boolean values parallel to the <i>input</i> array. Nonzero (true, flagged) suggests that the corresponding Unicode character be forced to uppercase after being decoded (if possible), and zero (false, unflagged) suggests that it be forced to lowercase (if possible). ASCII code points (07F) are encoded literally, except that ASCII letters are forced to uppercase or	
	lowercase according to the corresponding case flags. If case_flags is a NULL pointer then ASCII letters are left as they are, and other code points are treated as unflagged.	
output_length	The caller passes in the maximum number of ASCII code points that it can receive. On successful return it will contain the number of ASCII code points actually output.	
output	An array of ASCII code points. It is *not* null-terminated; it will contain zeros if and only if the input contains zeros. (Of course the caller can leave room for a terminator and add one if needed.)	

Returns

The return value can be any of the Punycode_status values defined above except PUNYCODE_BAD_INPUT. If not PUNYCODE_SUCCESS, then output_size and output might contain garbage.

punycode_decode ()

Converts Punycode to a sequence of code points (presumed to be Unicode code points).

input_length	The number of ASCII code	
mput_iengui	points in the input array.	
input	An array of ASCII code	
mput	points (07F).	
	The caller passes in the	
	maximum number of code	
	points that it can receive	
	into the output array	
	(which is also the	
	maximum number of flags	
	that it can receive into the	
	case_flags array, if	
	case_flags is not a	
	NULL pointer). On	
	successful return it will	
	contain the number of code	
	points actually output	
	(which is also the number	
output_length	of flags actually output, if	
	case_flags is not a null	
	pointer). The decoder will	
	never need to output more	
	code points than the number	
	of ASCII code points in the	
	input, because of the way	
	the encoding is defined.	
	The number of code points	
	output cannot exceed the	
	maximum possible value of	
	a punycode_uint, even if the	
	supplied output_length	
	is greater than that.	
	An array of code points like	
output	the input argument of	
1	punycode_encode() (see	
	above).	
	A NULL pointer (if the	
	flags are not needed by the	
	caller) or an array of	
	boolean values parallel to	
	the output array. Nonzero	
	(true, flagged) suggests that	
	the corresponding Unicode	
	character be forced to	
	uppercase by the caller (if	
case_flags	possible), and zero (false,	
	unflagged) suggests that it	
	be forced to lowercase (if	
	possible). ASCII code	
	points (07F) are output	
	already in the proper case,	
	but their flags will be set	
	appropriately so that	
	applying the flags would be	
	harmless.	

Returns

The return value can be any of the Punycode_status values defined above. If not PUNYCODE_SUCCESS, then <code>output_length</code> , <code>output</code> , and <code>case_flags</code> might contain garbage.

Types and Values

IDNAPI

#define

IDNAPI

Symbol holding shared library API visibility decorator.

This is used internally by the library header file and should never be used or modified by the application.

https://www.gnu.org/software/gnulib/manual/html_node/Exported-Symbols-of-Shared-Libraries.html

enum Punycode_status

Enumerated return codes of punycode_encode() and punycode_decode(). The value 0 is guaranteed to always correspond to success.

	Successful
	op-
	er-
	a-
	tion.
	This
	value
	value
	is
	guar-
	an-
	teed
	to
	al-
	ways
	be
	zero,
	the
	re-
	main-
	ing
	ones
PUNYCODE_SUCCESS	
	are
	only
	guar-
	an-
	teed
	to
	hold
	non-
	zero
	val-
	ues,
	for
	log-
	i-
	cal
	com-
	par-
	i-
	søn
	pur-
	poses.
	Input
	is
PUNYCODE_BAD_INPUT	18
	in
	valid.
	Output
	would
	ex-
	ceed
PUNYCODE_BIG_OUTPUT	the
	space
	pro-
	vided.

PUNYCODE_OVERFLOW

PUNYCODE_OVERFLOW

tegers
to
process.

punycode_uint

```
typedef uint32_t punycode_uint;
```

Unicode code point data type, this is always a 32 bit unsigned integer.

1.4 pr29.h

pr29.h — PR29-related functions

Functions

const char *	pr29_strerror ()
int	pr29_4 ()
int	pr29_4z ()
int	pr29_8z ()

Types and Values

#define	IDNAPI
enum	Pr29_rc

Description

PR29-related functions.

Functions

pr29_strerror()

```
const char~*
pr29_strerror (Pr29_rc rc);
```

Convert a return code integer to a text string. This string can be used to output a diagnostic message to the user.

PR29_SUCCESS: Successful operation. This value is guaranteed to always be zero, the remaining ones are only guaranteed to hold non-zero values, for logical comparison purposes. PR29_PROBLEM: A problem sequence was encountered. PR29_STRINGPREP_ERROR: The character set conversion failed (only for pr29_8z()).

rc an Pr29_rc return code.

Returns

Returns a pointer to a statically allocated string containing a description of the error with the return code rc.

pr29_4 ()

Check the input to see if it may be normalized into different strings by different NFKC implementations, due to an anomaly in the NFKC specifications.

Parameters

in	input array with unicode code points.	
len	length of input array with unicode code points.	

Returns

Returns the Pr29_rc value PR29_SUCCESS on success, and PR29_PROBLEM if the input sequence is a "problem sequence" (i.e., may be normalized into different strings by different implementations).

pr29_4z()

```
int
pr29_4z (const uint32_t *in);
```

Check the input to see if it may be normalized into different strings by different NFKC implementations, due to an anomaly in the NFKC specifications.

Parameters

in	zero terminated array of
111	Unicode code points.

Returns

Returns the Pr29_rc value PR29_SUCCESS on success, and PR29_PROBLEM if the input sequence is a "problem sequence" (i.e., may be normalized into different strings by different implementations).

pr29_8z ()

```
int
pr29_8z (const char *in);
```

Check the input to see if it may be normalized into different strings by different NFKC implementations, due to an anomaly in the NFKC specifications.

Parameters

in	zero terminated input
III	UTF-8 string.

Returns

Returns the Pr29_rc value PR29_SUCCESS on success, and PR29_PROBLEM if the input sequence is a "problem sequence" (i.e., may be normalized into different strings by different implementations), or PR29_STRINGPREP_ERROR if there was a problem converting the string from UTF-8 to UCS-4.

Types and Values

IDNAPI

#define IDNAPI

Symbol holding shared library API visibility decorator.

This is used internally by the library header file and should never be used or modified by the application.

https://www.gnu.org/software/gnulib/manual/html_node/Exported-Symbols-of-Shared-Libraries.html

enum Pr29_rc

Enumerated return codes for pr29_4(), pr29_4z(), pr29_8z(). The value 0 is guaranteed to always correspond to success.

	d c.
	Successful
	р-
	r-
a	†
ti	ion.
T	his
V	alue
is	
g	uar-
	n-
l te	eed
to	o
a	.1-
, w	vays
b	p e
	ero,
	he
	e-
n	nain-
i	ng
	nes
	re
0	phly
	uar-
a	n-
te	eed
to	
h	old
	on-
	ero
	al-
	es,
f	or
	og-
i-	<u> </u>
	al
	om-
P i-	par-
	on
l P	our-
P	oses.
A	1 vah
$\mid \stackrel{p}{\underset{\scriptscriptstyle 1}{\scriptstyle 1}}$	rob-
	em
	e-
	uence
	vas
	n-
	oun-
te	ered.

PR29_STRINGPREP_ERROR

```
The char-
ac-
ter set con-
ver-
sion failed (only for pr29_8z()).
```

1.5 tld.h

tld.h — TLD-related functions

Functions

const char *	tld_strerror ()
int	tld_get_4 ()
int	tld_get_4z ()
int	tld_get_z ()
const Tld_table *	tld_get_table ()
const Tld_table *	tld_default_table ()
int int int	tld_check_4t ()
int	tld_check_4tz ()
int	tld_check_4 ()
int	tld_check_4z ()
int	tld_check_8z ()
int	tld check lz ()

Types and Values

#define	IDNAPI
struct	Tld_table_element
struct	Tld_table
enum	Tld_rc

Description

TLD-related functions.

Functions

tld_strerror ()

```
const char~*
tld_strerror (Tld_rc rc);
```

Convert a return code integer to a text string. This string can be used to output a diagnostic message to the user.

TLD_SUCCESS: Successful operation. This value is guaranteed to always be zero, the remaining ones are only guaranteed to hold non-zero values, for logical comparison purposes. TLD_INVALID: Invalid character found. TLD_NODATA: No input data was provided. TLD_MALLOC_ERROR: Error during memory allocation. TLD_ICONV_ERROR: Character encoding conversion error. TLD_NO_TLD: No top-level domain found in domain string.

Parameters

rc tld return code

Returns

Returns a pointer to a statically allocated string containing a description of the error with the return code rc.

tld_get_4 ()

Isolate the top-level domain of in and return it as an ASCII string in out.

Parameters

	Array of unicode code	
in	points to process. Does not	
	need to be zero terminated.	
inlen	Number of unicode code	
inien	points.	
out	Zero terminated ascii result	
Out	string pointer.	

Returns

Return TLD_SUCCESS on success, or the corresponding Tld_rc error code otherwise.

tld_get_4z()

Isolate the top-level domain of in and return it as an ASCII string in out.

	Zero terminated array of	
in	unicode code points to	
	process.	
out	Zero terminated ascii result	
out	string pointer.	

Returns

Return TLD_SUCCESS on success, or the corresponding Tld_rc error code otherwise.

tld_get_z ()

Isolate the top-level domain of in and return it as an ASCII string in out. The input string in may be UTF-8, ISO-8859-1 or any ASCII compatible character encoding.

Parameters

i	Zero terminated character	
111	array to process.	
out	Zero terminated ascii result	
	string pointer.	

Returns

Return TLD_SUCCESS on success, or the corresponding Tld_rc error code otherwise.

tld_get_table ()

Get the TLD table for a named TLD by searching through the given TLD table array.

Parameters

tld	TLD name (e.g. "com") as zero terminated ASCII byte	
	string.	
	Zero terminated array of	
tables	Tld_table info-structures	
	for TLDs.	

Returns

Return structure corresponding to TLD tld by going thru tables, or return NULL if no such structure is found.

tld_default_table ()

Get the TLD table for a named TLD, using the internal defaults, possibly overridden by the (optional) supplied tables.

Parameters

	TLD name (e.g. "com") as	
tld	zero terminated ASCII byte	
	string.	
	Additional zero terminated	
	array of Tld_table	
overrides	info-structures for TLDs, or	
	NULL to only use library	
	default tables.	

Returns

Return structure corresponding to TLD tld_str , first looking through overrides then thru built-in list, or NULL if no such structure found.

tld_check_4t()

Test each of the code points in in for whether or not they are allowed by the data structure in tld, return the position of the first character for which this is not the case in errpos.

Parameters

	Array of unicode code	
in	points to process. Does not	
	need to be zero terminated.	
inlen	Number of unicode code	
IIIIeII	points.	
arross	Position of offending	
errpos	character is returned here.	
tld	A Tld_table data structure	
	representing the restrictions	
	for which the input should	
	be tested.	

Returns

Returns the Tld_rc value TLD_SUCCESS if all code points are valid or when tld is null, TLD_INVALID if a character is not allowed, or additional error codes on general failure conditions.

tld_check_4tz ()

Test each of the code points in in for whether or not they are allowed by the data structure in tld, return the position of the first character for which this is not the case in errpos.

Parameters

in	Zero terminated array of	
	unicode code points to	
	process.	
Arrnos	Position of offending	
errpos	character is returned here.	
	A Tld_table data structure	
tld	representing the restrictions	
	for which the input should	
	be tested.	

Returns

Returns the $\overline{\text{Tld}}_{\text{rc}}$ value $\overline{\text{TLD}}_{\text{SUCCESS}}$ if all code points are valid or when t1d is null, $\overline{\text{TLD}}_{\text{INVALID}}$ if a character is not allowed, or additional error codes on general failure conditions.

tld_check_4 ()

Test each of the code points in *in* for whether or not they are allowed by the information in *overrides* or by the built-in TLD restriction data. When data for the same TLD is available both internally and in *overrides*, the information in *overrides* takes precedence. If several entries for a specific TLD are found, the first one is used. If *overrides* is NULL, only the built-in information is used. The position of the first offending character is returned in *errpos*.

Parameters

	Array of unicode code	
in	points to process. Does not	
	need to be zero terminated.	
inlen	Number of unicode code	
Illeli	points.	
arrnos	Position of offending	
errpos	character is returned here.	
	A Tld_table array of	
	additional domain	
overrides	restriction structures that	
	complement and supersede	
	the built-in information.	

Returns

Returns the Tld_rc value TLD_SUCCESS if all code points are valid or when tld is null, TLD_INVALID if a character is not allowed, or additional error codes on general failure conditions.

tld_check_4z ()

```
int
tld_check_4z (const uint32_t *in,
```

```
size_t *errpos,
const Tld_table **overrides);
```

Test each of the code points in <code>in</code> for whether or not they are allowed by the information in <code>overrides</code> or by the built-in TLD restriction data. When data for the same TLD is available both internally and in <code>overrides</code>, the information in <code>overrides</code> takes precedence. If several entries for a specific TLD are found, the first one is used. If <code>overrides</code> is <code>NULL</code>, only the built-in information is used. The position of the first offending character is returned in <code>errpos</code>.

Parameters

in	Zero-terminated array of	
	unicode code points to	
	process.	
errnos	Position of offending	
errpos	character is returned here.	
	A Tld_table array of	
overrides	additional domain	
	restriction structures that	
	complement and supersede	
	the built-in information.	

Returns

Returns the Tld_rc value TLD_SUCCESS if all code points are valid or when tld is null, TLD_INVALID if a character is not allowed, or additional error codes on general failure conditions.

tld_check_8z()

Test each of the characters in <code>in</code> for whether or not they are allowed by the information in <code>overrides</code> or by the built-in TLD restriction data. When data for the same TLD is available both internally and in <code>overrides</code>, the information in <code>overrides</code> takes precedence. If several entries for a specific TLD are found, the first one is used. If <code>overrides</code> is <code>NULL</code>, only the built-in information is used. The position of the first offending character is returned in <code>errpos</code>. Note that the error position refers to the decoded character offset rather than the byte position in the string.

in	Zero-terminated UTF8	
	string to process.	
arrnos	Position of offending	
errpos	character is returned here.	
	A Tld_table array of	
	additional domain	
overrides	restriction structures that	
	complement and supersede	
	the built-in information.	
	'	•

Returns

Returns the Tld_rc value TLD_SUCCESS if all characters are valid or when tld is null, TLD_INVALID if a character is not allowed, or additional error codes on general failure conditions.

tld_check_lz()

Test each of the characters in <code>in</code> for whether or not they are allowed by the information in <code>overrides</code> or by the built-in TLD restriction data. When data for the same TLD is available both internally and in <code>overrides</code>, the information in <code>overrides</code> takes precedence. If several entries for a specific TLD are found, the first one is used. If <code>overrides</code> is <code>NULL</code>, only the built-in information is used. The position of the first offending character is returned in <code>errpos</code>. Note that the error position refers to the decoded character offset rather than the byte position in the string.

Parameters

in	Zero-terminated string in	
	the current locales encoding	
	to process.	
Arrnos	Position of offending	
errpos	character is returned here.	
	A Tld_table array of	
	additional domain	
overrides	restriction structures that	
	complement and supersede	
	the built-in information.	

Returns

Returns the Tld_rc value TLD_SUCCESS if all characters are valid or when t1d is null, TLD_INVALID if a character is not allowed, or additional error codes on general failure conditions.

Types and Values

IDNAPI

```
#define IDNAPI
```

Symbol holding shared library API visibility decorator.

This is used internally by the library header file and should never be used or modified by the application.

https://www.gnu.org/software/gnulib/manual/html_node/Exported-Symbols-of-Shared-Libraries.html

struct Tld_table_element

```
struct Tld_table_element {
    uint32_t start;
    uint32_t end;
};
```

Interval of valid code points in the TLD.

Members

	Start
<pre>uint32_t start;</pre>	ot
	range.
	End
	of
	range,
	end
uint32_t end;	+
	start
	if
	sin-
	gle.

struct Tld_table

```
struct Tld_table {
   const char *name;
   const char *version;
   size_t nvalid;
   const Tld_table_element *valid;
};
```

List valid code points in a TLD.

	TLD
const char *name;	name,
const chai mame,	e.g.,
	e.g., "no".
	Version
	string
const char *version;	from
	TLD
	file.
	Number
	of
olgo t 1 ' d	en-
size_t nvalid;	tries
	in
	data.
	Sorted
	ar-
	ray
	(of
	size
const Tld_table_element *valid;	nvalid
	of
	valid
	code
	points.
	•

enum Tld_rc

Enumerated return codes of the TLD checking functions. The value 0 is guaranteed to always correspond to success.

Successful op- et- ai tion. This value is guar- an- teed to al- ways be zero, the re- main- ing ones arc only guar- an- teed to hold hold hon- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter found,		
Op- ct- at- tijn. This value is guar- an- teed to al- ways be zero, the re- main- iing ones ar- only guar- an- teed to bold non- zero val- to bold non- zero val- teed to		Successful
### ### ##############################		
a+ tion. This value is guar- an- teed to al- ways be zero, tie re- main- iing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid charage TLD_INVALID ac- ter		
tion. This value is guar- an- teed to al- ways be zero, the re- main- ing ones are only guar- an- teed to in ILD_SUCCESS TLD_SUCCESS		
This value is a guarant product of the control of t		
value is guar- ah- teed to al- ways be zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- pair- i- son pur- poses Invalid char- at- ter TLD_INVALID		
is guar- an- teed to al- ways be zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- pur- pises Invalid char- ter		
guar- an- teed to al- ways be zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- pur- pir- j- son pur- poses. Invalid char- are- terd to to hold non- pur- poses. Invalid char- are- terd terd terd terd		
ah- teed to al- ways be zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- pir- i- son pur- poses. Invalid char- ae- ter		
teed to al- ways be zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- par- i- son pur- poses. Invalid char- ac- ter to		
to al- ways be Zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log i- cal compar i- son pur- poses. Invalid charate Invalid Invalid Charate Invalid Inval		
al- ways be zero, the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- sepn pur- poses. Invalid char- ac- ter t		
TLD_SUCCESS Representation of the company of the		
De Zero, the re- main- ing ones are only guar- an- teed to hold non- Zero val- ues, for log- i- cal com- par- i- søn pur- poses. Invalid charar ar- ted to to to to to to to t		al-
zero, the re- re- main- ing ones are only guaran- teed to hold non- zero val- ues, for log- i- cal comparison jur- poses liyualid charan teed to to to to to to to t		ways
TLD_SUCCESS the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- phoses. Invalid char- ac- ter TLD_INVALID		be
TLD_SUCCESS the re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- phoses. Invalid char- ac- ter TLD_INVALID		zero,
TLD_SUCCESS re- main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal compar- i- son pur- poses.		
TLD_SUCCESS main- ing ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- pyses. Invalid char- ac- ter		
Ing Ones are only guaran- teed to hold non- zero values, for log- i- cal com- par- i- son pur- poses.		
TLD_SUCCESS ones are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		
are only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- at-		
only guar- an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- at- teed to hold non- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses.	TLD_SUCCESS	
gliar- ah- teed to hold non- zero val- ues, for log- i- cal com- par- i- sen pur- poses. Ilvalid char- at- ter		
an- teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		
teed to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		
to hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		
hold non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		
non- zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- TLD_INVALID ac- ter		
zero val- ues, for log- i- cal com- par- i- son pur- poses. Invalid char- TLD_INVALID		
values, for log- i- cal compar- i- son pur- poses. Invalid charater TLD_INVALID ac- ter		
ues, for log- i- cal compar- i- son pur- poses. Invalid character TLD_INVALID ac- ter		
for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		val-
for log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		ues,
log- i- cal com- par- i- son pur- poses. Invalid char- ac- ter		for
i- cal com- par- i- son pur- poses. Invalid char- ac- ter		
cal com- par- i- son pur- poses. Invalid char- ac- ter		i-
compar- i- son pur- poses. Invalid char- ac- ter		
par- i- son pur- poses. Invalid char- ac- ter		
son pur- poses. Invalid char- TLD_INVALID ac- ter		
son pur- poses. Invalid char- TLD_INVALID ac- ter		pm
purposes. Invalid char- TLD_INVALID ac- ter		
poses. Invalid char- TLD_INVALID ac- ter		
TLD_INVALID Invalid char- ac- ter		pur-
TLD_INVALID char- ac- ter		poses.
TLD_INVALID ac- ter		
te <mark>r</mark>		
ter found.	TLD_INVALID	
found.		ter
		found.

	No No
	in-
	put
TLD_NODATA	data
122_103/m1	
	was
	pro-
	vided.
	Error
	dur-
	ing
	mem-
TID MALLOC EDDOD	
TLD_MALLOC_ERROR	ory
	al-
	10-
	ca-
	tion.
	Character
	en-
	cod-
	ing
TLD_ICONV_ERROR	con-
122_1001,2401	
	ver-
	sion
	er-
	ror.
	No
	top-
	level
	do-
TLD_NO_TLD	main
ILD_NO_ILD	found
	in
	do-
	main
	string.
	Same
	as
	TLD_NO_TLD
	,
	for
	com-
	pat-
	i-
TLD_NOTLD	
	bil-
	ity
	with
	typo
	typo in
	ear-
	lier
	ver-
	sions.
	1 2/2

1.6 idn-free.h

idn-free.h — Memory deallocation functions

Types and Values

#define IDNAPI

Description

Memory deallocation functions.

Functions

Types and Values

IDNAPI

#define IDNAPI

Symbol holding shared library API visibility decorator.

This is used internally by the library header file and should never be used or modified by the application.

https://www.gnu.org/software/gnulib/manual/html_node/Exported-Symbols-of-Shared-Libraries.html

Chapter 2

Index

I	Stringprep_profile, 37
IDNA_ACE_PREFIX, 14	stringprep_profile, 19
Idna_flags, 14	Stringprep_profile_flags, 32
Idna_rc, 9	Stringprep_profile_steps, 35
idna_strerror, 3	Stringprep_profiles, 37
idna_to_ascii_4i, 3	Stringprep_rc, 26
idna_to_ascii_4z, 5	stringprep_strerror, 21
idna_to_ascii_8z, 5	Stringprep_table, 36
idna_to_ascii_lz, 6	Stringprep_table_element, 36
idna_to_unicode_44i, 4	stringprep_ucs4_nfkc_normalize, 24
idna_to_unicode_4z4z, 6	stringprep_ucs4_to_utf8, 23
idna_to_unicode_8z4z, 7	stringprep_unichar_to_utf8, 22
idna_to_unicode_8z8z, 7	stringprep_utf8_nfkc_normalize, 23
idna_to_unicode_8zlz, 8	stringprep_utf8_to_locale, 25
idna_to_unicode_lzlz, 8	stringprep_utf8_to_ucs4, 22
IDNAPI, 9, 26, 42, 46, 54, 58	stringprep_utf8_to_unichar, 22
	STRINGPREP_VERSION, 26
P	stringprep_xmpp_nodeprep, 17
pr29_4, 45	stringprep_xmpp_resourceprep, 17
pr29_4z, 45	
pr29_8z, 45	T
Pr29_rc, 46	tld_check_4, 52
pr29_strerror, 44	tld_check_4t, 51
punycode_decode, 40	tld_check_4tz, 51
punycode_encode, 39	tld_check_4z, 52
Punycode_status, 42	tld_check_8z, 53
punycode_strerror, 38	tld_check_lz, 54
punycode_uint, 44	tld_default_table, 50
C C	tld_get_4, 49
S	tld_get_4z, 49
stringprep, 19	tld_get_table, 50
stringprep_4i, 17	tld_get_z, 50
stringprep_4zi, 18 stringprep_check_version, 21	Tld_rc, 56
stringprep_cneck_version, 21 stringprep_convert, 25	tld_strerror, 48
stringprep_convert, 25 stringprep_iscsi, 17	Tld_table, 55
stringprep_iscsi, 17 stringprep_kerberos5, 16	Tld_table_element, 54
stringprep_locale_charset, 24	
stringprep_locale_to_utf8, 25	
STRINGPREP_MAX_MAP_CHARS, 36	
stringprep_nameprep, 15	
stringprep_nameprep_no_unassigned, 16	
stringprep_nameprep_no_unassigned, 10 stringprep_plain, 16	