NAME

ldns_dnssec_trust_tree_new, ldns_dnssec_trust_tree_free, ldns_dnssec_trust_tree_depth, ldns_dnssec_derive_trust_tree, ldns_dnssec_trust_tree_contains_keys, ldns_dnssec_trust_tree_print, ldns_dnssec_trust_tree_print_sm, ldns_dnssec_trust_tree_add_parent, ldns_dnssec_derive_trust_tree_normal_rrset, ldns_dnssec_derive_trust_tree_dnskey_rrset, ldns_dnssec_derive_trust_tree_ds_rrset, ldns_dnssec_derive_trust_tree_no_sig - functions for ldns_dnssec_trust_tree

SYNOPSIS

```
#include <stdint.h>
#include <stdbool.h>

#include <ldns/ldns.h>

ldns_dnssec_trust_tree* ldns_dnssec_trust_tree_new(void);

void ldns_dnssec_trust_tree_free(ldns_dnssec_trust_tree *tree);

size_t ldns_dnssec_trust_tree_depth(ldns_dnssec_trust_tree *tree);

ldns_dnssec_trust_tree* ldns_dnssec_derive_trust_tree( ldns_dnssec_data_chain *data_chain, ldns_rr
*rr);

ldns_status ldns_dnssec_trust_tree_contains_keys( ldns_dnssec_trust_tree *tree, ldns_rr_list *keys);

void ldns_dnssec_trust_tree_print(FILE *out, ldns_dnssec_trust_tree *tree, size_t tabs, bool extended);

 ldns_dnssec_trust_tree_print_sm();

ldns_status ldns_dnssec_trust_tree_add_parent(ldns_dnssec_trust_tree *tree, const
ldns_dnssec_trust_tree *parent, const ldns_rr *parent_signature, const ldns_status parent_status);

void ldns_dnssec_derive_trust_tree_normal_rrset( ldns_dnssec_trust_tree *new_tree,
ldns_dnssec_data_chain *data_chain, ldns_rr *cur_sig_rr);

void ldns_dnssec_derive_trust_tree_dnskey_rrset( ldns_dnssec_trust_tree *new_tree,
ldns_dnssec_data_chain *data_chain, ldns_rr *cur_rr, ldns_rr *cur_sig_rr);

void ldns_dnssec_derive_trust_tree_ds_rrset( ldns_dnssec_trust_tree *new_tree,
```

ldns_dnssec_data_chain *data_chain, ldns_rr *cur_rr);

void ldns_dnssec_derive_trust_tree_no_sig(ldns_dnssec_trust_tree *new_tree,
ldns_dnssec_data_chain *data_chain);

DESCRIPTION

ldns_dnssec_trust_tree_new() Creates a new (empty) dnssec_trust_tree structure

Returns ldns_dnssec_trust_tree *

ldns_dnssec_trust_tree_free() Frees the dnssec_trust_tree recursively

There is no deep free; all data in the trust tree consists of pointers to a data_chain

tree: The tree to free

ldns_dnssec_trust_tree_depth() returns the depth of the trust tree

tree: tree to calculate the depth of
Returns The depth of the tree

ldns_dnssec_derive_trust_tree() Generates a dnssec_trust_tree for the given rr from the given data_chain

This does not clone the actual data; Don't free the data_chain before you are done with this tree

***data_chain**: The chain to derive the trust tree from
***rr**: The RR this tree will be about
Returns ldns_dnssec_trust_tree *

ldns_dnssec_trust_tree_contains_keys() Returns OK if there is a trusted path in the tree to one of the DNSKEY or DS RRs in the given list

\param *tree The trust tree so search \param *keys A ldns_rr_list of DNSKEY and DS rrs to look for

Returns LDNS_STATUS_OK if there is a trusted path to one of the keys, or the *first* error encountered if there were no paths

ldns_dnssec_trust_tree_print() Prints the dnssec_trust_tree structure to the given file stream.

If a link status is not LDNS_STATUS_OK; the status and relevant signatures are printed too

***out**: The file stream to print to
tree: The trust tree to print
tabs: Prepend each line with tabs*2 spaces
extended: If true, add little explanation lines to the output

ldns_dnssec_trust_tree_print_sm()

ldns_dnssec_trust_tree_add_parent() Adds a trust tree as a parent for the given trust tree

***tree**: The tree to add the parent to
***parent**: The parent tree to add
***parent_signature**: The RRSIG relevant to this parent/child connection
parent_status: The DNSSEC status for this parent, child and RRSIG
Returns LDNS_STATUS_OK if the addition succeeds, error otherwise

ldns_dnssec_derive_trust_tree_normal_rrset() Sub function for derive_trust_tree that is used for a 'normal' rrset

new_tree: The trust tree that we are building
data_chain: The data chain containing the data for the trust tree
cur_sig_rr: The currently relevant signature

ldns_dnssec_derive_trust_tree_dnskey_rrset() Sub function for derive_trust_tree that is used for DNSKEY rrsets

new_tree: The trust tree that we are building
data_chain: The data chain containing the data for the trust tree
cur_rr: The currently relevant DNSKEY RR
cur_sig_rr: The currently relevant signature

ldns_dnssec_derive_trust_tree_ds_rrset() Sub function for derive_trust_tree that is used for DS rrsets

new_tree: The trust tree that we are building
data_chain: The data chain containing the data for the trust tree
cur_rr: The currently relevant DS RR

ldns_dnssec_derive_trust_tree_no_sig() Sub function for derive_trust_tree that is used when there are no signatures

new_tree: The trust tree that we are building
data_chain: The data chain containing the data for the trust tree

AUTHOR

The ldns team at NLnet Labs.

REPORTING BUGS

Please report bugs to ldns-team@nlnetlabs.nl or in our bugzilla at
http://www.nlnetlabs.nl/bugs/index.html

COPYRIGHT

SEE ALSO

ldns_dnssec_data_chain, *ldns_dnssec_trust_tree*. And **perldoc Net::DNS**, **RFC1034**, **RFC1035**, **RFC4033**, **RFC4034** and **RFC4035**.

REMARKS

This manpage was automatically generated from the ldns source code.