

**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](mailto:ldns-team@nlnetlabs.nl) or in our bugzilla at <http://www.nlnetlabs.nl/bugs/index.html>

## COPYRIGHT

Copyright (c) 2004 - 2006 NLnet Labs.

Licensed under the BSD License. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

## 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.