zenoh-c

Release 0.5.0

ADLINK zenoh team

Nov 03, 2020

CONTENTS

The *libzenoh-c* library provides a C client API for the zenoh protocol.

An introduction to zenoh and its concepts is available on zenoh.io.

Note that only the zenoh-net API is available in c at this time.

CHAPTER

ONE

ZENOH-NET API

1.1 Examples

1.1.1 Publish

```
#include <string.h>
#include "zenoh/net.h"

int main(int argc, char **argv) {
    char* value = "value";

    zn_session_t *s = zn_open(zn_config_default());
    zn_write(s, zn_rname("/res/name"), value, strlen(value));
    zn_close(s);

    return 0;
}
```

1.1.2 Subscribe

```
#include <stdio.h>
#include "zenoh/net.h"

void data_handler(const zn_sample_t *sample, const void *arg) {
    printf(">> Received (%.*s, %.*s)\n",
        (int)sample->key.len, sample->key.val,
        (int)sample->value.len, sample->value.val);
}

int main(int argc, char **argv) {
    zn_session_t *s = zn_open(zn_config_default());
    zn_subscriber_t *sub = zn_declare_subscriber(s, zn_rname("/res/name"), zn_subinfo_
↪default(), data_handler, NULL);

    char c = 0;
    while (c != 'q') {
        c = fgetc(stdin);
    }

    zn_undeclare_subscriber(sub);
    zn_close(s);
```

(continues on next page)

(continued from previous page)

```
    return 0;
}
```

1.1.3 Query

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include "zenoh/net.h"

void reply_handler(const zn_source_info_t *info, const zn_sample_t *sample, const␣
↪void *arg) {
    printf(">> Received (%.*s, %.*s)\n",
        (int)sample->key.len, sample->key.val,
        (int)sample->value.len, sample->value.val);
}

int main(int argc, char** argv) {
    zn_session_t *s = zn_open(zn_config_default());
    zn_query(s, zn_rname("/res/name"), "", zn_query_target_default(), zn_query_
↪consolidation_default(), reply_handler, NULL);

    sleep(1);

    zn_close(s);
    return 0;
}
```

1.2 API Reference

1.2.1 Types

1.2.1.1 String

struct z_string_t
A string.

const char ***val**
A pointer to the string.

unsigned int **len**
The length of the string.

z_string_t **z_string_make**(**const** char **s*)
Construct a *z_string_t* from a NULL terminated string. The content of the given string is copied.

Parameters

- **s** – The NULL terminated string.

Returns A new *z_string_t*.

1.2.1.2 Array of Str

struct z_str_array_t
An array of NULL terminated strings.

char ***const** ***val**
A pointer to the array.

unsigned int **len**
The length of the array.

1.2.1.3 Bytes

struct z_bytes_t
An array of bytes.

const unsigned char ***val**
A pointer to the bytes array.

unsigned int **len**
The length of the bytes array.

1.2.1.4 Properties

type zn_properties_t
A map of key/value properties where the key is an `unsigned int` and the value a *z_string_t*. Multiple values are coma separated.

zn_properties_t ***zn_properties_make**()
Return a new empty map of properties.

unsigned int **zn_properties_len**(*zn_properties_t *ps*)
Get the length of the given properties map.

Parameters

- **ps** – A pointer to the properties map.

Returns The length of the given properties map.

zn_properties_t ***zn_properties_insert**(*zn_properties_t *ps*, unsigned long *key*, *z_string_t value*)
Insert a property with a given key to a properties map. If a property with the same key already exists in the properties map, it is replaced.

Parameters

- **ps** – A pointer to the properties map.
- **key** – The key of the property to add.
- **value** – The value of the property to add.

Returns A pointer to the updated properties map.

z_string_t **zn_properties_get**(*zn_properties_t *ps*, unsigned int *key*)
Get the property with the given key from a properties map.

Parameters

- **ps** – A pointer to properties map.
- **key** – The key of the property.

Returns The value of the property with key `key` in properties map `ps`.

void **zn_properties_free**(*zn_properties_t *ps*)
Free a set of properties.

Parameters

- **ps** – A pointer to the properties.

1.2.2 Scouting

1.2.2.1 Types

Possible flags in a whatami bitmask :

const unsigned int **ZN_ROUTER**

const unsigned int **ZN_PEER**

const unsigned int **ZN_CLIENT**

struct zn_hello_t
A hello message returned by a zenoh entity to a scout message sent with *zn_scout()*.

unsigned int **whatami**
The kind of zenoh entity.

z_bytes_t **pid**
The peer id of the scouted entity (empty if absent).

z_str_array_t **locators**
The locators of the scouted entity.

struct zn_hello_array_t
An array of *zn_hello_t* messages.

const *zn_hello_t* ***val**
A pointer to the array.

unsigned int **len**
The length of the array.

1.2.2.2 Functions

zn_hello_array_t **zn_scout**(unsigned int *what*, *zn_properties_t *config*, unsigned long *scout_period*)
Scout for routers and/or peers.

Parameters

- **what** – A whatami bitmask of zenoh entities kind to scout for.
- **config** – A set of properties to configure the scouting.
- **scout_period** – The time that should be spent scouting before returnng the results.

Returns An array of *zn_hello_t* messages.

void **zn_hello_array_free**(*zn_hello_array_t hellos*)
Free an array of *zn_hello_t* messages and it's contained *zn_hello_t* messages recursively.

Parameters

- **strs** – The array of *zn_hello_t* messages to free.

1.2.3 Session

1.2.3.1 Session configuration

A zenoh-net session is configured through a *zn_properties_t* properties map.

Multiple values are coma separated.

The following constants define the several property keys accepted for a zenoh-net session configuration and the associated accepted values.

const unsigned int **ZN_CONFIG_MODE_KEY**
The library mode.

- Accepted values : "peer", "client".
- Default value : "peer".

const unsigned int **ZN_CONFIG_PEER_KEY**

The locator of a peer to connect to.

- Accepted values : <locator> (ex: "tcp/10.10.10.10:7447").
- Default value : None.
- Multiple values accepted.

const unsigned int **ZN_CONFIG_LISTENER_KEY**
A locator to listen on.

- Accepted values : <locator> (ex: "tcp/10.10.10.10:7447").
- Default value : None.
- Multiple values accepted.

const unsigned int **ZN_CONFIG_USER_KEY**
The user name to use for authentication.

- Accepted values : <string>.
- Default value : None.

const unsigned int **ZN_CONFIG_PASSWORD_KEY**
The password to use for authentication.

- Accepted values : <string>.
- Default value : None.

const unsigned int **ZN_CONFIG_MULTICAST_SCOUTING_KEY**
Activates/Desactivates multicast scouting.

- Accepted values : "true", "false".
- Default value : "true".

const unsigned int **ZN_CONFIG_MULTICAST_INTERFACE_KEY**
The network interface to use for multicast scouting.

- Accepted values : "auto", <ip address>, <interface name>.
- Default value : "auto".

const unsigned int **ZN_CONFIG_MULTICAST_ADDRESS_KEY**
The multicast address and ports to use for multicast scouting.

- Accepted values : `<ip address>:<port>`.
- Default value : `"224.0.0.224:7447"`.

const unsigned int **ZN_CONFIG_SCOUTING_TIMEOUT_KEY**
In client mode, the period dedicated to scouting a router before failing.

- Accepted values : `<float in seconds>`.
- Default value : `"3.0"`.

const unsigned int **ZN_CONFIG_SCOUTING_DELAY_KEY**
In peer mode, the period dedicated to scouting first remote peers before doing anything else.

- Accepted values : `<float in seconds>`.
- Default value : `"0.2"`.

const unsigned int **ZN_CONFIG_ADD_TIMESTAMP_KEY**
Indicates if data messages should be timestamped.

- Accepted values : `"true"`, `"false"`.
- Default value : `"false"`.

const unsigned int **ZN_CONFIG_LOCAL_ROUTING_KEY**
Indicates if local writes/queries should reach local subscribers/queryables.

- Accepted values : `"true"`, `"false"`.
- Default value : `"true"`.

The following functions allow to create default *zn_properties_t* maps for zenoh-net session configuration. The returned configurations can be amended with extra options with *zn_properties_insert()*.

zn_properties_t ***zn_config_empty**()
Create an empty set of properties for zenoh-net session configuration.

zn_properties_t ***zn_config_default**()
Create a default set of properties for zenoh-net session configuration.

zn_properties_t ***zn_config_peer**()
Create a default set of properties for peer mode zenoh-net session configuration.

zn_properties_t ***zn_config_client**(char **peer*)
Create a default set of properties for client mode zenoh-net session configuration. If peer is not null, it is added to the configuration as remote peer.

Parameters

- **peer** – An optional peer locator.

1.2.3.2 Session management

zn_session_t ***zn_open**(*zn_properties_t* **config*)
Open a zenoh-net session

Parameters

- **config** – A set of properties.

Returns The created zenoh-net session or null if the creation did not succeed.

zn_properties_t ***zn_info**(zn_session_t **session*)
Get informations about an zenoh-net session.

Parameters

- **session** – A zenoh-net session.

Returns A *zn_properties_t* map containing informations on the given zenoh-net session.

void **zn_close**(zn_session_t **session*)
Close a zenoh-net session.

Parameters

- **session** – A zenoh-net session.

1.2.4 Resource

1.2.4.1 Resource key

struct zn_reskey_t
A resource key.

Resources are identified by URI like string names. Examples : `"/some/resource/key"`. Resource names can be mapped to numerical ids through *zn_declare_resource()* for wire and computation efficiency.

A resource key can be either:

- A plain string resource name.
- A pure numerical id.
- The combination of a numerical prefix and a string suffix.

unsigned long **id**
The id or prefix of this resource key. `0` if empty.

const char ***suffix**
The suffix of this resource key. `NULL` if pure numerical id.

zn_reskey_t **zn_rname**(**const** char **name*)
Create a resource key from a resource name.

Parameters

- **id** – The resource name.

Returns A new resource key.

zn_reskey_t **zn_rid**(unsigned long *id*)
Create a resource key from a resource id.

Parameters

- **id** – The resource id.

Returns A new resource key.

zn_reskey_t **zn_rid_with_suffix**(unsigned long *id*, **const** char **suffix*)
Create a resource key from a resource id and a suffix.

Parameters

- **id** – The resource id.
- **suffix** – The suffix.

Returns A new resource key.

1.2.4.2 Sample

struct zn_sample_t
A zenoh-net data sample.

A sample is the value associated to a given resource at a given point in time.

z_string_t **key**
The resource key of this data sample.

z_bytes_t **value**
The value of this data sample.

1.2.4.3 Resource declaration

unsigned long **zn_declare_resource**(zn_session_t *session, *zn_reskey_t reskey*)
Associate a numerical id with the given resource key.

This numerical id will be used on the network to save bandwidth and ease the retrieval of the concerned resource in the routing tables.

Parameters

- **session** – The zenoh-net session.
- **resource** – The resource key to map to a numerical id.

Returns A numerical id.

1.2.5 Publication

1.2.5.1 Types

type zn_publisher_tr
A zenoh-net Publisher.

1.2.5.2 Functions

zn_publisher_t ***zn_declare_publisher**(zn_session_t *session, *zn_reskey_t reskey*)
Declare a `zn_publisher_t` for the given resource key.

Written resources that match the given key will only be sent on the network if matching subscribers exist in the system.

Parameters

- **session** – The zenoh-net session.
- **resource** – The resource key to publish.

Returns The created `zn_publisher_t` or null if the declaration failed.

void **zn_undeclare_publisher**(zn_publisher_t *publ)
Undeclare a `zn_publisher_t`.

Parameters

- **sub** – The `zn_publisher_t` to undeclare.

int **zn_write**(zn_session_t *_session_, _zn_reskey_t_ _reskey_, **const** char *_payload_, unsigned int _len_)
Write data.

Parameters

- **session** – The zenoh-net session.
- **resource** – The resource key to write.
- **payload** – The value to write.
- **len** – The length of the value to write.

Returns `0` in case of success, `1` in case of failure.

1.2.6 Subscription

1.2.6.1 Types

type zn_subscriber_t
A zenoh-net subscriber.

enum zn_reliability_t
The subscription reliability.

- **zn_reliability_t_BEST_EFFORT**
- **zn_reliability_t_RELIABLE**

enum zn_submode_t
The subscription mode.

- **zn_submode_t_PUSH**
- **zn_submode_t_PULL**

struct zn_period_t
The subscription period.

unsigned int **origin**

unsigned int **period**

unsigned int **duration**

struct zn_subinfo_t
Informations to be passed to _zn_declare_subscriber()_ to configure the created _zn_subscriber_t_.

_zn_reliability_t_ **reliability**
The subscription reliability.

_zn_submode_t_ **mode**
The subscription mode.

_zn_period_t_ ***period**
The subscription period.

_zn_subinfo_t_ **zn_subinfo_default**()
Create a default subscription info.

1.2.6.2 Functions

zn_subscriber_t ***zn_declare_subscriber**(zn_session_t **session*, *zn_reskey_t reskey*, *zn_subinfo_t sub_info*, void (**callback*))(**const** *zn_sample_t**, **const** void*, void **arg*)

Declare a *zn_subscriber_t* for the given resource key.

Parameters

- **session** – The zenoh-net session.
- **resource** – The resource key to subscribe.
- **sub_info** – The *zn_subinfo_t* to configure the *zn_subscriber_t*.
- **callback** – The callback function that will be called each time a data matching the subscribed resource is received.
- **arg** – A pointer that will be passed to the **callback** on each call.

Returns The created *zn_subscriber_t* or null if the declaration failed.

void **zn_pull**(*zn_subscriber_t *sub*)

Pull data for a pull mode *zn_subscriber_t*. The pulled data will be provided by calling the **callback** function provided to the *zn_declare_subscriber()* function.

Parameters

- **sub** – The *zn_subscriber_t* to pull from.

void **zn_undeclare_subscriber**(*zn_subscriber_t *sub*)

Undeclare a *zn_subscriber_t*.

Parameters

- **sub** – The *zn_subscriber_t* to undeclare.

1.2.7 Query

1.2.7.1 Types

struct zn_target_t

Which amongst the matching queryables should be target of a *zn_query()*.

zn_target_t_Tag **tag**;

zn_target_t_COMPLETE_Body **complete**;

Members of zn_target_t when *zn_target_t.tag* is set to `zn_target_t_COMPLETE`.

unsigned int **n**

The number of complete queryables that should be target of a *zn_query()*.

enum zn_target_t_Tag

The possible values of *zn_target_t.tag*.

- **zn_target_t_BEST_MATCHING**: The nearest complete queryable if any else all matching queryables.
- **zn_target_t_COMPLETE**: A set of complete queryables.
- **zn_target_t_ALL**: All matching queryables.
- **zn_target_t_NONE**: No queryables.

zn_target_t **zn_target_default**()
Create a default *zn_target_t*.

The network interface to use for multicast scouting.

struct zn_query_target_t
The zenoh-net queryables that should be target of a *zn_query()*.

unsigned int **kind**
A mask of queryable kinds.

zn_target_t **target**
The query target.

Predefined values for *zn_query_target_t.kind*:

const unsigned int **ZN_QUERYABLE_ALL_KINDS**

const unsigned int **ZN_QUERYABLE_EVAL**

const unsigned int **ZN_QUERYABLE_STORAGE**

zn_query_target_t **zn_query_target_default**()
Create a default *zn_query_target_t*.

enum zn_consolidation_mode_t
The kind of consolidation that should be applied on replies to a *zn_query()*.

- **zn_consolidation_mode_t_FULL**: Guaranties unicity of replies. Optimizes bandwidth.
- **zn_consolidation_mode_t_LAZY**: Does not garanty unicity. Optimizes latency.
- **zn_consolidation_mode_t_NONE**: No consolidation.

struct zn_query_consolidation_t
The kind of consolidation that should be applied on replies to a *zn_query()* at the different stages of the reply process.

zn_consolidation_mode_t **first_routers**
The consolidation mode to apply on first routers of the replies routing path.

zn_consolidation_mode_t **last_router**
The consolidation mode to apply on last router of the replies routing path.

zn_consolidation_mode_t **reception**
The consolidation mode to apply at reception of the replies.

zn_query_consolidation_t **zn_query_consolidation_default**()
Create a default *zn_query_consolidation_t*.

struct zn_source_info_t
Information on the source of a reply.

unsigned int **kind**
The kind of source.

z_bytes_t **id**
The unique id of the source.

1.2.7.2 Functions

void **zn_query**(zn_session_t *_session_, _zn_reskey_t_ _reskey_, **const** char *_predicate_, _zn_query_target_t_ _target_, _zn_query_consolidation_t_ _consolidation_, void (*_callback_))(**const** _zn_source_info_t_*, **const** _zn_sample_t_*, **const** void*), void *_arg_

Query data from the matching queryables in the system.

Parameters

- **session** – The zenoh-net session.
- **resource** – The resource key to query.
- **predicate** – An indication to matching queryables about the queried data.
- **target** – The kind of queryables that should be target of this query.
- **consolidation** – The kind of consolidation that should be applied on replies.
- **callback** – The callback function that will be called on reception of replies for this query.
- **arg** – A pointer that will be passed to the **callback** on each call.

1.2.8 Queryable

1.2.8.1 Types

type zn_queryable_t

The zenoh-net Queryable.

1.2.8.2 Functions

_zn_queryable_t_ ***zn_declare_queryable**(zn_session_t *_session_, _zn_reskey_t_ _reskey_, unsigned int _kind_, void (*_callback_))(zn_query_t*, **const** void*), void *_arg_

Declare a _zn_queryable_t_ for the given resource key.

Parameters

- **session** – The zenoh-net session.
- **resource** – The resource key the _zn_queryable_t_ will reply to.
- **kind** – The kind of _zn_queryable_t_.
- **callback** – The callback function that will be called each time a matching query is received.
- **arg** – A pointer that will be passed to the **callback** on each call.

Returns The created _zn_queryable_t_ or null if the declaration failed.

Predefined values for `kind`:

const unsigned int **ZN_QUERYABLE_EVAL**

const unsigned int **ZN_QUERYABLE_STORAGE**

void **zn_undeclare_queryable**(_zn_queryable_t_ *_qable_)

Undeclare a _zn_queryable_t_.

Parameters

- **qable** – The _zn_queryable_t_ to undeclare.

INDEX

Z