Name

libmosquitto — MQTT version 3.1 client library

Description

This is an overview of how to use libmosquitto to create MQTT aware client programs. There may be separate man pages on each of the functions described here in the future. There is also a binding for libmosquitto for C++ and a Python implementation. They are not documented here but operate in a similar way.

This is fairly incomplete, please see mosquitto.h for a better description of the functions.

libmosquitto symbol names

All public functions in libmosquitto have the prefix "mosquitto_". Any other functions defined in the source code are to be treated as private functions and may change between any release. Do not use these functions!

Functions

Library version

```
int mosquitto_lib_version( major,
                           minor,
                           revision);
int *major;
int *minor;
int *revision;
```

Obtain version information about the library. If any of major, minor or revision are not NULL they will return the corresponding version numbers. The return value is an integer representation of the complete version number (e.g. 1009001 for 1.9.1) that can be used for comparisons.

Library initialisation and cleanup

```
int mosquitto_lib_init( );

int mosquitto_lib_cleanup( );
```

Call mosquitto_lib_init() before using any of the other library functions and mosquitto_lib_cleanup() after finishing with the library.

Client constructor/destructor

```
struct mosquitto *mosquitto_new( id,
                                 clean_session,
                                 userdata);
const char *id;
bool clean_session;
void *userdata;
```

Create a new mosquitto client instance.

```
void mosquitto_destroy( mosq);
struct mosquitto *mosq;
```

Use to free memory associated with a mosquitto client instance.

```
int mosquitto_reinitialise( mosq,
                            id,
                            clean_session,
                            userdata);
struct mosquitto *mosq;
const char *id;
```

```
bool clean_session;
void *userdata;
```

Authentication and encryption

```
int mosquitto_username_pw_set( mosq,
                               username,
                               password);
struct mosquitto *mosq;
const char *username;
const char *password;

int mosquitto_tls_set( mosq,
                       cafile,
                       capath,
                       certfile,
                       keyfile,
                       (*pw_callback)(char *buf, int size, int rwflag, void *userdata));
struct mosquitto *mosq;
const char *cafile;
const char *capath;
const char *certfile;
const char *keyfile;
int (*pw_callback)(char *buf, int size, int rwflag, void *userdata);

int mosquitto_tls_opts_set( mosq,
                            cert_reqs,
                            tls_version,
                            ciphers);
struct mosquitto *mosq;
int cert_reqs;
const char *tls_version;
const char *ciphers;

int mosquitto_tls_insecure_set( mosq,
                                value);
struct mosquitto *mosq;
bool value;

int mosquitto_tls_psk_set( mosq,
                           psk,
                           identity,
                           ciphers);
struct mosquitto *mosq;
const char *psk;
const char *identity;
const char *ciphers;
```

Wills

```
int mosquitto_will_set( mosq,
                        topic,
                        payloadlen,
                        payload,
                        qos,
                        retain);
struct mosquitto *mosq;
const char *topic;
int payloadlen;
const void *payload;
int qos;
bool retain;

int mosquitto_will_clear( struct mosquitto *mosq);
struct mosquitto *mosq;
```

Connect/disconnect

```
int mosquitto_connect( mosq,
                       host,
                       port,
                       keepalive);
struct mosquitto *mosq;
const char *host;
int port;
int keepalive;
```

```
int mosquitto_connect_bind( mosq,
                            host,
                            port,
                            keepalive,
                            bind_address);
struct mosquitto *mosq;
const char *host;
int port;
int keepalive;
const char *bind_address;
```

```
int mosquitto_connect_async( mosq,
                             host,
                             port,
                             keepalive);
struct mosquitto *mosq;
const char *host;
int port;
int keepalive;
```

```
int mosquitto_connect_bind_async( mosq,
                                  host,
                                  port,
                                  keepalive,
                                  bind_address);
struct mosquitto *mosq;
const char *host;
int port;
int keepalive;
const char *bind_address;
```

```
int mosquitto_reconnect( mosq);
struct mosquitto *mosq;
```

```
int mosquitto_reconnect_async( mosq);
struct mosquitto *mosq;
```

```
int mosquitto_disconnect( mosq);
struct mosquitto *mosq;
```

Publish

```
int mosquitto_publish( mosq,
                       mid,
                       topic,
                       payloadlen,
                       payload,
                       qos,
                       retain);
struct mosquitto *mosq;
int *mid;
const char *topic;
```

```
int payloadlen;
const void *payload;
int qos;
bool retain;
```

Subscribe/unsubscribe

```
int mosquitto_subscribe( mosq,
                         mid,
                         sub,
                         qos);
struct mosquitto *mosq;
int *mid;
const char *sub;
int qos;
```

```
int mosquitto_unsubscribe( mosq,
                           mid,
                           sub);
struct mosquitto *mosq;
int *mid;
const char *sub;
```

Network loop

```
int mosquitto_loop( mosq,
                    timeout,
                    max_packets);
struct mosquitto *mosq;
int timeout;
int max_packets;
```

```
int mosquitto_loop_read( mosq,
                         max_packets);
struct mosquitto *mosq;
int max_packets;
```

```
int mosquitto_loop_write( mosq,
                          max_packets);
struct mosquitto *mosq;
int max_packets;
```

```
int mosquitto_loop_misc( mosq);
struct mosquitto *mosq;
```

```
int mosquitto_loop_forever( mosq,
                            timeout,
                            max_packets);
struct mosquitto *mosq;
int timeout;
int max_packets;
```

```
int mosquitto_socket( mosq);
struct mosquitto *mosq;
```

```
bool mosquitto_want_write( mosq);
struct mosquitto *mosq;
```

Threaded network loop

```
int mosquitto_loop_start( mosq);
struct mosquitto *mosq;
```

```
int mosquitto_loop_stop( mosq,
                                    force);
struct mosquitto *mosq;
bool force;
```

Misc client functions

```
int mosquitto_max_inflight_messages_set( mosq,
                                         max_inflight_messages);
struct mosquitto *mosq;
unsigned int max_inflight_messages;
```

```
int mosquitto_message_retry_set( mosq,
                                 message_retry);
struct mosquitto *mosq;
unsigned int message_retry;
```

```
int mosquitto_reconnect_delay_set( mosq,
                                   reconnect_delay,
                                   reconnect_delay_max,
                                   reconnect_exponential_backoff);
struct mosquitto *mosq;
unsigned int reconnect_delay;
unsigned int reconnect_delay_max;
bool reconnect_exponential_backoff;
```

```
int mosquitto_user_data_set( mosq,
                             userdata);
struct mosquitto *mosq;
void *userdata;
```

Callbacks

```
int mosquitto_connect_callback_set( mosq,
                                    (*on_connect)(struct mosquitto *, void *, int));
struct mosquitto *mosq;
void (*on_connect)(struct mosquitto *, void *, int);
```

```
int mosquitto_disconnect_callback_set( mosq,
                                       (*on_disconnect)(struct mosquitto *, void *, int));
struct mosquitto *mosq;
void (*on_disconnect)(struct mosquitto *, void *, int);
```

```
int mosquitto_publish_callback_set( mosq,
                                    (*on_publish)(struct mosquitto *, void *, int));
struct mosquitto *mosq;
void (*on_publish)(struct mosquitto *, void *, int);
```

```
int mosquitto_message_callback_set( mosq,
                                    (*on_message)(struct mosquitto *, void *, const struct mosquitto_message *));
struct mosquitto *mosq;
void (*on_message)(struct mosquitto *, void *, const struct mosquitto_message *);
```

```
int mosquitto_subscribe_callback_set( mosq,
                                      (*on_subscribe)(struct mosquitto *, void *, int, int, const int *));
struct mosquitto *mosq;
void (*on_subscribe)(struct mosquitto *, void *, int, int, const int *);
```

```
int mosquitto_unsubscribe_callback_set( mosq,
                                        (*on_unsubscribe)(struct mosquitto *, void *, int));
struct mosquitto *mosq;
void (*on_unsubscribe)(struct mosquitto *, void *, int);
```

```
int mosquitto_log_callback_set( mosq,
                                (*on_unsubscribe)(struct mosquitto *, void *, int, const char *));
```

```
struct mosquitto *mosq;
void (*on_unsubscribe)(struct mosquitto *, void *, int, const char *);
```

Utility functions

```
const char *mosquitto_connack_string( connack_code);
int connack_code;

int mosquitto_message_copy( dst,
                            src);
struct mosquitto_message *dst;
const struct mosquitto_message *src;

int mosquitto_message_free( message);
struct mosquitto_message **message;

const char *mosquitto_strerror( mosq_errno);
int mosq_errno;

int mosquitto_sub_topic_tokenise( subtopic,
                                  topics,
                                  count);
const char *subtopic;
char ***topics;
int *count;

int mosquitto_sub_topic_tokens_free( topics,
                                     count);
char ***topics;
int count;

int mosquitto_topic_matches_sub( sub,
                                 topic,
                                 result);
const char *sub;
const char *topic;
bool *result;
```

Examples

```
#include <stdio.h>
#include <mosquitto.h>

void my_message_callback(struct mosquitto *mosq, void *userdata, const struct mosquitto_message *message)
{
        if(message->payloadlen){
                printf("%s %s\n", message->topic, message->payload);
        }else{
                printf("%s (null)\n", message->topic);
        }
        fflush(stdout);
}

void my_connect_callback(struct mosquitto *mosq, void *userdata, int result)
{
        int i;
        if(!result){
                /* Subscribe to broker information topics on successful connect. */
                mosquitto_subscribe(mosq, NULL, "$SYS/#", 2);
        }else{
                fprintf(stderr, "Connect failed\n");
        }
}

void my_subscribe_callback(struct mosquitto *mosq, void *userdata, int mid, int qos_count, const int *granted_qos)
{
        int i;

        printf("Subscribed (mid: %d): %d", mid, granted_qos[0]);
        for(i=1; i<qos_count; i++){
                printf(", %d", granted_qos[i]);
        }
        printf("\n");
}

void my_log_callback(struct mosquitto *mosq, void *userdata, int level, const char *str)
{
        /* Pring all log messages regardless of level. */
        printf("%s\n", str);
}

int main(int argc, char *argv[])
{
        int i;
        char *host = "localhost";
        int port = 1883;
        int keepalive = 60;
        bool clean_session = true;
        struct mosquitto *mosq = NULL;

        mosquitto_lib_init();
        mosq = mosquitto_new(NULL, clean_session, NULL);
        if(!mosq){
                fprintf(stderr, "Error: Out of memory.\n");
                return 1;
        }
        mosquitto_log_callback_set(mosq, my_log_callback);
        mosquitto_connect_callback_set(mosq, my_connect_callback);
        mosquitto_message_callback_set(mosq, my_message_callback);
        mosquitto_subscribe_callback_set(mosq, my_subscribe_callback);

        if(mosquitto_connect(mosq, host, port, keepalive)){
                fprintf(stderr, "Unable to connect.\n");
                return 1;
        }

        mosquitto_loop_forever(mosq, -1, 1);

        mosquitto_destroy(mosq);
        mosquitto_lib_cleanup();
        return 0;
}
```

See Also

mosquitto(8) mqtt(7)

Author

Roger Light <roger@atchoo.org>