```
Userspace communication protocol over connector [1].


Message types.
=============

There are three types of messages between w1 core and userspace:
1. Events. They are generated each time new master or slave device
        found either due to automatic or requested search.
2. Userspace commands.
3. Replies to userspace commands.


Protocol.
========

[struct cn_msg] - connector header.
        Its length field is equal to size of the attached data
[struct w1_netlink_msg] - w1 netlink header.
        __u8 type       - message type.
                        W1_LIST_MASTERS
                                list current bus masters
                        W1_SLAVE_ADD/W1_SLAVE_REMOVE
                                slave add/remove events
                        W1_MASTER_ADD/W1_MASTER_REMOVE
                                master add/remove events
                        W1_MASTER_CMD
                                userspace command for bus master
                                device (search/alarm search)
                        W1_SLAVE_CMD
                                userspace command for slave device
                                (read/write/touch)
        __u8 res        - reserved
        __u16 len       - size of data attached to this header data
        union {
                __u8 id[8];                     - slave unique device id
                struct w1_mst {
                        __u32           id;     - master's id
                        __u32           res;    - reserved
                } mst;
        } id;

[struct w1_netlink_cmd] - command for given master or slave device.
        __u8 cmd        - command opcode.
                        W1_CMD_READ     - read command
                        W1_CMD_WRITE    - write command
                        W1_CMD_TOUCH    - touch command
                                (write and sample data back to userspace)
                        W1_CMD_SEARCH   - search command
                        W1_CMD_ALARM_SEARCH - alarm search command
        __u8 res        - reserved
        __u16 len       - length of data for this command
                For read command data must be allocated like for write command
        __u8 data[0]    - data for this command
```

```
Each connector message can include one or more w1_netlink_msg with
zero or more attached w1_netlink_cmd messages.

For event messages there are no w1_netlink_cmd embedded structures,
only connector header and w1_netlink_msg strucutre with "len" field
being zero and filled type (one of event types) and id:
either 8 bytes of slave unique id in host order,
or master's id, which is assigned to bus master device
when it is added to w1 core.

Currently replies to userspace commands are only generated for read
command request. One reply is generated exactly for one w1_netlink_cmd
read request. Replies are not combined when sent - i.e. typical reply
messages looks like the following:

[cn_msg][w1_netlink_msg][w1_netlink_cmd]
cn_msg.len = sizeof(struct w1_netlink_msg) +
             sizeof(struct w1_netlink_cmd) +
             cmd->len;
w1_netlink_msg.len = sizeof(struct w1_netlink_cmd) + cmd->len;
w1_netlink_cmd.len = cmd->len;

Replies to W1_LIST_MASTERS should send a message back to the userspace
which will contain list of all registered master ids in the following
format:

        cn_msg (CN_W1_IDX.CN_W1_VAL as id, len is equal to sizeof(struct
        w1_netlink_msg) plus number of masters multipled by 4)
        w1_netlink_msg (type: W1_LIST_MASTERS, len is equal to
                number of masters multiplied by 4 (u32 size))
        id0 ... idN

        Each message is at most 4k in size, so if number of master devices
        exceeds this, it will be split into several messages,
        cn.seq will be increased for each one.

W1 search and alarm search commands.
request:
[cn_msg]
  [w1_netlink_msg type = W1_MASTER_CMD
        id is equal to the bus master id to use for searching]
  [w1_netlink_cmd cmd = W1_CMD_SEARCH or W1_CMD_ALARM_SEARCH]

reply:
  [cn_msg, ack = 1 and increasing, 0 means the last message,
        seq is equal to the request seq]
  [w1_netlink_msg type = W1_MASTER_CMD]
  [w1_netlink_cmd cmd = W1_CMD_SEARCH or W1_CMD_ALARM_SEARCH
        len is equal to number of IDs multiplied by 8]
  [64bit-id0 ... 64bit-idN]
Length in each header corresponds to the size of the data behind it, so
w1_netlink_cmd->len = N * 8; where N is number of IDs in this message.
        Can be zero.
w1_netlink_msg->len = sizeof(struct w1_netlink_cmd) + N * 8;
cn_msg->len = sizeof(struct w1_netlink_msg) +
              sizeof(struct w1_netlink_cmd) +
		N*8;

W1 reset command.
[cn_msg]
  [w1_netlink_msg type = W1_MASTER_CMD
  	id is equal to the bus master id to use for searching]
  [w1_netlink_cmd cmd = W1_CMD_RESET]
```

Command status replies.
```
======================
```

Each command (either root, master or slave with or without w1_netlink_cmd
structure) will be 'acked' by the w1 core. Format of the reply is the same
as request message except that length parameters do not account for data
requested by the user, i.e. read/write/touch IO requests will not contain
data, so w1_netlink_cmd.len will be 0, w1_netlink_msg.len will be size
of the w1_netlink_cmd structure and cn_msg.len will be equal to the sum
of the sizeof(struct w1_netlink_msg) and sizeof(struct w1_netlink_cmd).
If reply is generated for master or root command (which do not have
w1_netlink_cmd attached), reply will contain only cn_msg and w1_netlink_msg
structires.

w1_netlink_msg.status field will carry positive error value
(EINVAL for example) or zero in case of success.

All other fields in every structure will mirror the same parameters in the
request message (except lengths as described above).

Status reply is generated for every w1_netlink_cmd embedded in the
w1_netlink_msg, if there are no w1_netlink_cmd structures,
reply will be generated for the w1_netlink_msg.

All w1_netlink_cmd command structures are handled in every w1_netlink_msg,
even if there were errors, only length mismatch interrupts message processing.

Operation steps in w1 core when new command is received.
```
=======================================================
```

When new message (w1_netlink_msg) is received w1 core detects if it is
master or slave request, according to w1_netlink_msg.type field.
Then master or slave device is searched for.
When found, master device (requested or those one on where slave device
is found) is locked. If slave command is requested, then reset/select
procedure is started to select given device.

Then all requested in w1_netlink_msg operations are performed one by one.
If command requires reply (like read command) it is sent on command completion.

When all commands (w1_netlink_cmd) are processed muster device is unlocked
and next w1_netlink_msg header processing started.

Connector [1] specific documentation.
```
====================================
```

Each connector message includes two u32 fields as "address".
w1 uses CN_W1_IDX and CN_W1_VAL defined in include/linux/connector.h header.
Each message also includes sequence and acknowledge numbers.
Sequence number for event messages is appropriate bus master sequence number
increased with each event message sent "through" this master.
Sequence number for userspace requests is set by userspace application.
Sequence number for reply is the same as was in request, and
acknowledge number is set to seq+1.

Additional documantion, source code examples.

1. Documentation/connector
2. http://www.ioremap.net/archive/w1
This archive includes userspace application w1d.c which uses
read/write/search commands for all master/slave devices found on the bus.