PJSIP Snippets

Adrian Kosmaczewski

2018-04-24

I've been working in an IP telephony project with PJSIP, and had to implement a few features here and there. Here's a couple of snippets I used all over the place.

Extract header value

```
pj_str_t event_hdr_name = pj_str("Event");
pjsip_generic_string_hdr *event_hdr = (pjsip_generic_string_hdr*)pjsip_msg_find_hdr_by_name(
if (event_hdr == NULL)
    return NULL;
pj_str_t event_value = event_hdr->vhalue;
```

Source.

Add header to list

```
struct pjsip_generic_string_hdr header;
pj_str_t name = pj_str("Name");
pj_str_t value = pj_str("value");
pjsip_generic_string_hdr_init2(& header, &name, &value);

pj_list_push_back(&cfg.reg_hdr_list, &header);
```

Source.

Clone header

```
// Set the "Contact" header
pj_str_t contact_hdr_name = pj_str("");
pj_str_t contact_hdr_value = pj_str("");
const pj_str_t *name = pj_cstr(&contact_hdr_name, "Contact");
const pj_str_t *value = pj_cstr(&contact_hdr_value, "<sip:alice@192.168.7.12;transport=TCP;c
pjsip_generic_string_hdr *hdr = pjsip_generic_string_hdr_create(pool, name, value);

// Clone the "Contact" header
```

```
pjsip_contact_hdr *contact_hdr = (pjsip_contact_hdr*)pjsip_msg_find_hdr_by_name(last_data_re
pjsip_contact_hdr *clone_hdr = (pjsip_contact_hdr*)pjsip_hdr_clone(pool, contact_hdr);
```

Add User-Agent header

```
pj_str_t ua_hdr_name = pj_str("User-Agent");
pj_str_t ua_hdr_value = pj_str("CSTAEngine");
status = csta_tdata_add_header(pool, *tdata, &ua_hdr_name, &ua_hdr_value);
```

Get origin IP address

```
// Requires a memory pool
pjsip_response_addr res_addr;
pjsip_get_response_addr(pool, rdata, &res_addr);
```

Build response to a server request outside of dialog

Keep a reference to the data:

```
static pjsip_rx_data *last_data_received;

// ...later, on the module callback functions...
pjsip_rx_data_clone(rdata, 0, &last_data_received);
```

Use it to create a response outside of a dialog:

```
// Respond to the original message
pjsip_endpt_respond(endpoint, NULL, last_data_received, PJSIP_SC_OK, NULL, &hdr_list, body,
```

Minimal PJSIP Stack

Custom thread worker:

```
static pj_bool_t quit_flag = PJ_FALSE;
static int worker_thread(void *arg)
{
    PJ_UNUSED_ARG(arg);

    while (!quit_flag) {
        pj_time_val timeout = {0, 500};
        pjsip_endpt_handle_events(endpoint, &timeout);
    }

    return 0;
}
```

Custom stack setup:

```
pj_init();
pj_caching_pool_init(&cp, &pj_pool_factory_default_policy, 0);

pj_sockaddr addr;
pj_status_t status;

pj_log_set_level(3);

status = pjlib_util_init();
csta_log_assert(status);

status = pjsip_endpt_create(&cp.factory, NULL, &endpoint);
csta_log_assert(status);

pj_uint16_t sip_af = pj_AF_INET();

pj_sockaddr_init(sip_af, &addr, NULL, sip_port);
if (sip_af == pj_AF_INET()) {
    if (sip_tcp) {
        status = pjsip_tcp_transport_start(endpoint, &addr.ipv4, 1, NULL);
    }
    else {
        status = pjsip_udp_transport_start(endpoint, &addr.ipv4, NULL, 1, NULL);
    }
}
else if (sip_af == pj_AF_INET6()) {
    status = pjsip_udp_transport_start6(endpoint, &addr.ipv6, NULL, 1, NULL);
}
else {
    status = PJ_EAFNOTSUP;
}

csta_log_assert(status);

status = pjsip_tsx_layer_init_module(endpoint);
csta_log_assert(status);

status = pjsip_ua_init_module(endpoint, NULL);
csta_log_assert(status);

//        pj_bzero(&inv_cb, sizeof(inv_cb));
//        inv_cb.on_state_changed = &call_on_state_changed;
//        inv_cb.on_new_session = &call_on_forked;
//        inv_cb.on_media_update = &call_on_media_update;
//        inv_cb.on_rx_offer = &call_on_rx_offer;
```

```
//        status = pjsip_inv_usage_init(endpoint, &inv_cb);
//        csta_log_assert(status);

status = pjsip_100rel_init_module(endpoint);
csta_log_assert(status);

pj_pool_t *pool = pjsip_endpt_create_pool(endpoint, "", 1000, 1000);
status = pj_thread_create(pool, "", &worker_thread, NULL, 0, 0, &thread);
csta_log_assert(status);
```

Remember to add a module to be notified of events:

```
pj_status_t status = pjsip_endpt_register_module(endpoint, &csta_module);
```

Minimal PJSUA Stack

Much simpler than using bare bones PJSIP:

```
pj_status_t status;

pj_caching_pool_init(&cp, &pj_pool_factory_default_policy, 0);

status = pjsua_create();
csta_log_assert(status);

pjsua_config cfg;
pjsua_config_default (&cfg);
NSString *userAgentString = [PJSUALibraryManager userAgentString];
cfg.user_agent = pj_str((char*)[userAgentString cStringUsingEncoding:NSUTF8StringEncoding]);

pjsua_logging_config log_cfg;
pjsua_logging_config_default(&log_cfg);
log_cfg.console_level = 4;
log_cfg.decor &= ~(PJ_LOG_HAS_TIME | PJ_LOG_HAS_MICRO_SEC);

// Init the pjsua
status = pjsua_init(&cfg, &log_cfg, NULL);
csta_log_assert(status);

pjsua_transport_config cfg2;
pjsua_transport_config_default(&cfg2);
cfg2.port = 0;

// Add UDP transport.
status = pjsua_transport_create(PJSIP_TRANSPORT_UDP, &cfg2, NULL);
csta_log_assert(status);
```

```
pjsua_transport_config cfg3;
pjsua_transport_config_default(&cfg3);

// Add TCP transport.
status = pjsua_transport_create(PJSIP_TRANSPORT_TCP, &cfg3, NULL);
csta_log_assert(status);

// Initialization is done, now start pjsua
status = pjsua_start();
csta_log_assert(status);
```