Tizen IoTivity.Connectivity

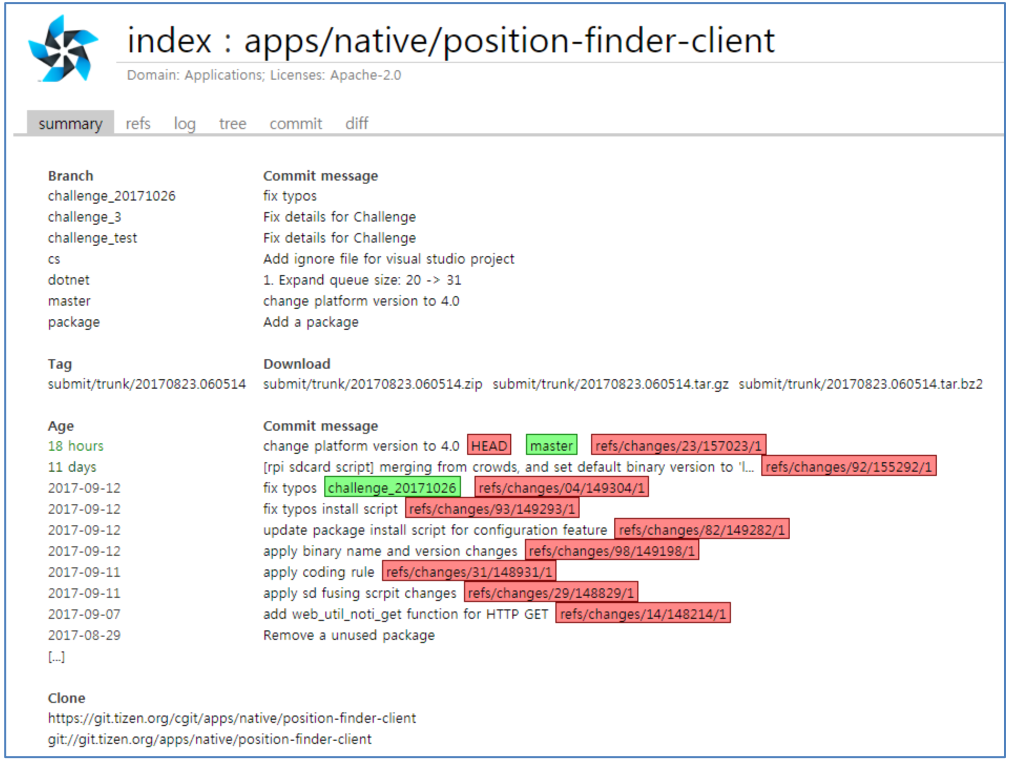
Client

SWC | Tizen Platform Lab | Geun Sun Lee

2017.10.26

SOSCON
SAMSUNG
OPEN SOURCE
CONFERENCE

position-finder-client



index : apps/native/position-finder-client

Domain: Applications; Licenses: Apache-2.0

Branch	Commit message
challenge_20171026	fix typos
challenge_3	Fix details for Challenge
challenge_test	Fix details for Challenge
cs	Add ignore file for visual studio project
dotnet	1. Expand queue size: 20 -> 31
master	change platform version to 4.0
package	Add a package

Tag	Download
submit/trunk/20170823.060514	submit/trunk/20170823.060514.zip submit/trunk/20170823.060514.tar.gz submit/trunk/20170823.060514.tar.bz2

Age	Commit message
18 hours	change platform version to 4.0 HEAD master refs/changes/23/157023/1
11 days	[rpi sdcard script] merging from crowds, and set default binary version to 'l... refs/changes/92/155292/1
2017-09-12	fix typos challenge_20171026 refs/changes/04/149304/1
2017-09-12	fix typos install script refs/changes/93/149293/1
2017-09-12	update package install script for configuration feature refs/changes/82/149282/1
2017-09-12	apply binary name and version changes refs/changes/98/149198/1
2017-09-11	apply coding rule refs/changes/31/148931/1
2017-09-11	apply sd fusing scrpit changes refs/changes/29/148829/1
2017-09-07	add web_util_noti_get function for HTTP GET refs/changes/14/148214/1
2017-08-29	Remove a unused package
[...]	

Clone

https://git.tizen.org/cgit/apps/native/position-finder-client

git://git.tizen.org/apps/native/position-finder-client

Git path : apps/native/position-finder-client
Branch : master

- Find a server
- Observe remote resources
- Get a data from the server
- Control a remote resource(CRUD)

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb  cb,
        void  *user_data);
```

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb  cb,
        void *user_data);
```

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb cb,
        void *user_data);
```

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb  cb,
        void  *user_data);
```

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb cb,
        void *user_data);
```

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb cb,
        void *user_data);
```

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb cb,
        void *user_data);
```

```
typedef void (*connectivity_observe_resource_cb)(connectivity_resource_s *resource_info,
                                                  const char *path,
                                                  void *user_data);
```

```
struct _connectivity_resource_s {
    char *device_id;
    char *host_address;
    char *device_name;
    char *type;
    char *uri_path;
    …
    iotcon_remote_resource_h resource;
    iotcon_attributes_h attributes;
    …
};
```

```
typedef void (*connectivity_observe_resource_cb)(connectivity_resource_s *resource_info,
                                                 const char *path,
                                                 void *user_data);
```

```
struct _connectivity_resource_s {
    char *device_id;
    char *host_address;
    char *device_name;
    char *type;
    char *uri_path;
    ...
    iotcon_remote_resource_h resource;
    iotcon_attributes_h attributes;
    ...
};
```

```
typedef void (*connectivity_observe_resource_cb)(connectivity_resource_s  *resource_info,
                                                 const char  *path,
                                                 void  *user_data);
```

```
struct _connectivity_resource_s {
    char *device_id;
    char *host_address;
    char *device_name;
    char *type;
    char *uri_path;
    …
    iotcon_remote_resource_h resource;
    iotcon_attributes_h attributes;
    …
};
```

```
typedef void (*connectivity_observe_resource_cb)(connectivity_resource_s  *resource_info,
                                                 const char *path,
                                                 void *user_data);
```

position-finder-client

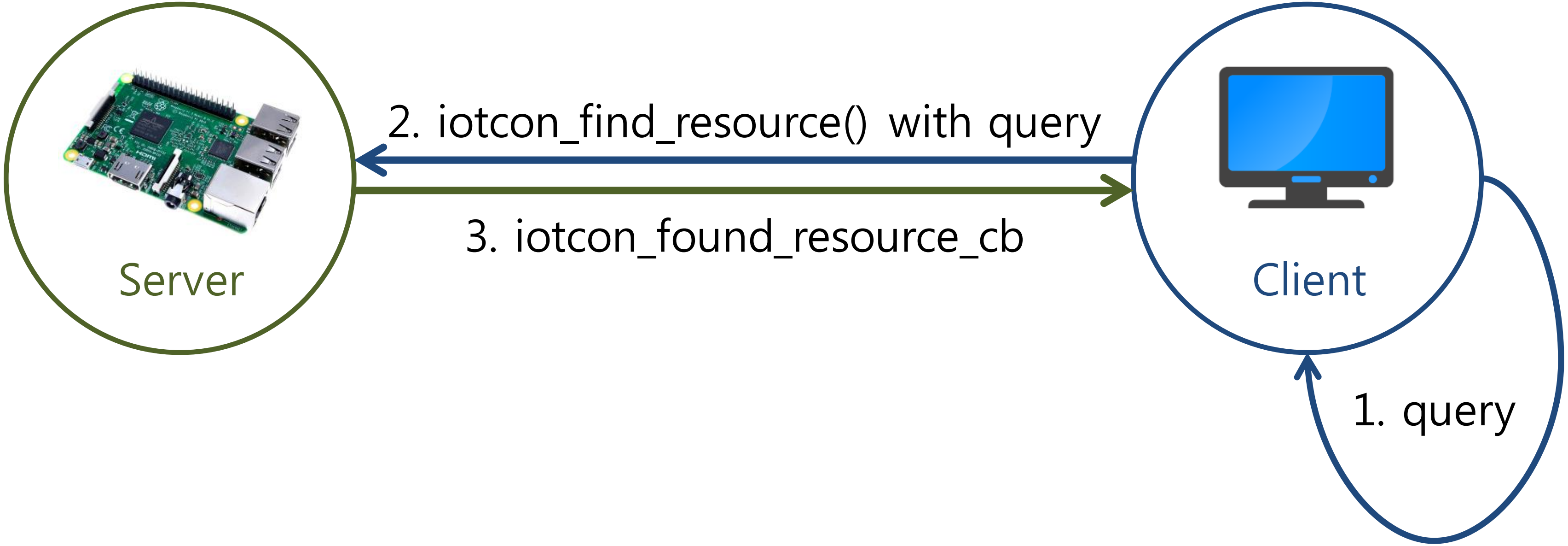
SOSCON

src/connectivity.c

```
int connectivity_observe_resource(const char *type, connectivity_observe_resource_cb cb, void *user_data)
{
    iotcon_query_h query = NULL;

    ...

    ret = iotcon_query_create(&query);

    ret = iotcon_query_set_resource_type(query, type);

    ret = iotcon_find_resource(IOTCON_MULTICAST_ADDRESS,
            IOTCON_CONNECTIVITY_IP | IOTCON_CONNECTIVITY_PREFER_UDP,
            query,
            _found_resource_cb,
            cb_info);

    iotcon_query_destroy(query);

    return 0;
}
```

Find a server

Find a server



Server
2. iotcon_find_resource() with query
3. iotcon_found_resource_cb
Client
1. query

QUERY

```
int  iotcon_query_create (iotcon_query_h *query)
int  iotcon_query_destroy (iotcon_query_h query)

int  iotcon_query_get_resource_type (iotcon_query_h query, char **resource_type)
int  iotcon_query_get_interface (iotcon_query_h query, char **resource_iface)
int  iotcon_query_set_resource_type (iotcon_query_h query, const char *resource_type)
int  iotcon_query_set_interface (iotcon_query_h query, const char *resource_iface)

int  iotcon_query_add (iotcon_query_h query, const char *key, const char *value)
int  iotcon_query_remove (iotcon_query_h query, const char *key)
int  iotcon_query_lookup (iotcon_query_h query, const char *key, char **data)

int  iotcon_query_foreach (iotcon_query_h query, iotcon_query_foreach_cb cb, void *user_data)
```

https://goo.gl/h59kWH

Find a resource

SOSCON

```
int  iotcon_find_resource (const char *host_address,
                           int connectivity_type,
                           iotcon_query_h query,
                           iotcon_found_resource_cb cb,
                           void  *user_data)


typedef bool(*  iotcon_found_resource_cb )(iotcon_remote_resource_h resource,
                                           iotcon_error_e result,
                                           void  *user_data)
```

IOTCON_CONNECTIVITY_ALL	Indicates all connectivities
IOTCON_CONNECTIVITY_IP	Indicates Internet Protocol connectivity
IOTCON_CONNECTIVITY_PREFER_UDP	It is related to IOTCON_CONNECTIVITY_IP, and it indicates UDP is preferred
IOTCON_CONNECTIVITY_PREFER_TCP	It is related to IOTCON_CONNECTIVITY_IP, and it indicates TCP is preferred
IOTCON_CONNECTIVITY_IPV4_ONLY	When this bit is set with IOTCON_CONNECTIVITY_IP, resources are discovered for IPv4
IOTCON_CONNECTIVITY_IPV6_ONLY	When this bit is set with IOTCON_CONNECTIVITY_IP, resources are discovered for IPv6

https://goo.gl/4dYfbY

server

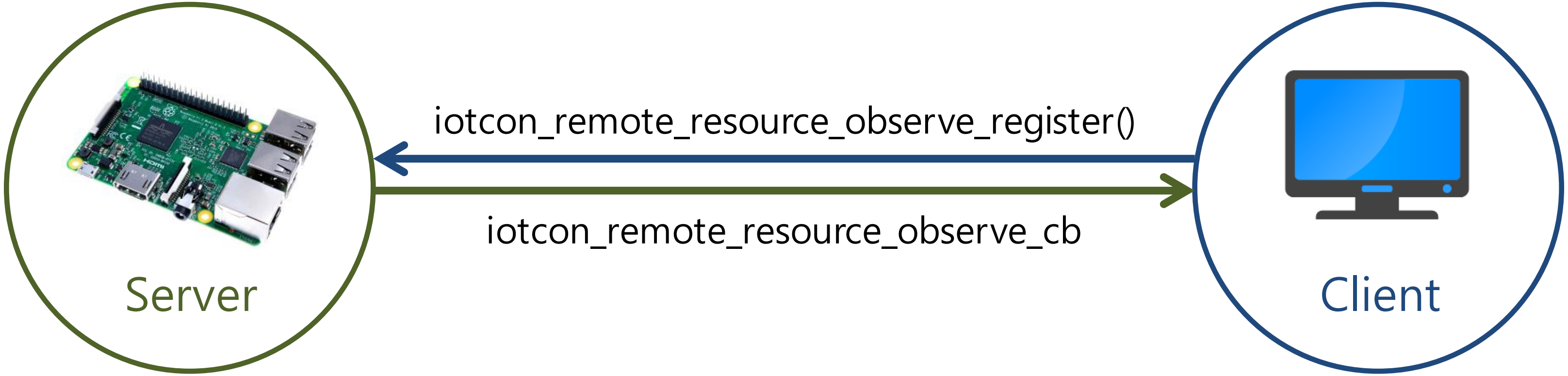
```
int connectivity_set_resource(const  char *uri_path, const  char *type,
connectivity_resource_s  **out_resource_info)
{
    ...
    iotcon_resource_interfaces_h ifaces  =  NULL;
    ...

    ret  =  iotcon_resource_interfaces_add(ifaces,
                    IOTCON_INTERFACE_DEFAULT);
    goto_if(IOTCON_ERROR_NONE  != ret, error);

    ret  =  iotcon_resource_interfaces_add(ifaces,
                    IOTCON_INTERFACE_BATCH);
    goto_if(IOTCON_ERROR_NONE  != ret, error);


    ...

    ret  =  iotcon_resource_create(uri_path,
            resource_types,
            ifaces,
            policies,
            _request_resource_handler,
            resource_info,
            &resource_info->res);
    goto_if(IOTCON_ERROR_NONE  != ret, error);
    ...
}
```

client

```
int connectivity_observe_resource(connectivity_observe_resource_cb   cb,
void *user_data)
{
    ...
     ret  =  iotcon_find_resource(IOTCON_MULTICAST_ADDRESS,
            IOTCON_CONNECTIVITY_IP  |
                IOTCON_CONNECTIVITY_PREFER_UDP,
            query,
            _found_resource_cb,
            cb_info);
    goto_if(IOTCON_ERROR_NONE  != ret, error);
    ...
}

static  bool  _found_resource_cb(iotcon_remote_resource_h resource,
iotcon_error_e result, void  *user_data)
{
    ...
    iotcon_resource_interfaces_h resource_interfaces;
    ...
    ret  =  iotcon_remote_resource_get_interfaces(resource,
&resource_interfaces);
    retv_if(IOTCON_ERROR_NONE  != ret,  -1);

    ret  =  iotcon_resource_interfaces_foreach(resource_interfaces,
_get_res_iface_cb, uri_path);
    retv_if(IOTCON_ERROR_NONE  != ret,  -1);
}
```

Observe remote resources

Observe remote resources



Server

iotcon_remote_resource_observe_register()

iotcon_remote_resource_observe_cb

Client

Observe remote resources

```
int  iotcon_remote_resource_observe_register (iotcon_remote_resource_h resource,
                                              iotcon_observe_policy_e observe_policy,
                                              iotcon_query_h query,
                                              iotcon_remote_resource_observe_cb cb,
                                              void  *user_data)


int  iotcon_remote_resource_observe_deregister (iotcon_remote_resource_h resource)
```

IOTCON_OBSERVE_IGNORE_OUT_OF_ORDER	Indicates observation request for most up-to-date notifications only
IOTCON_OBSERVE_ACCEPT_OUT_OF_ORDER	Indicates observation request for all notifications including state notifications

https://goo.gl/8HjgKf

server

```
int connectivity_set_resource(const  char *uri_path,  const char *type,
connectivity_resource_s  **out_resource_info)
{
    …
    iotcon_resource_create(uri_path,
            resource_types,  ifaces,  policies,  _request_resource_handler,
            resource_info,  &resource_info->res);
    …
}

static void  _request_resource_handler(iotcon_resource_h   resource,  iotcon_request_h
request,  void *user_data)
{
    iotcon_request_get_observe_type(request,   &observe_type);
    if (IOTCON_OBSERVE_REGISTER  ==  observe_type)  {
        iotcon_request_get_observe_id(request,   &observe_id);
        iotcon_observers_add(observers,   observe_id);
    } else  if (IOTCON_OBSERVE_DEREGISTER  ==  observe_type)  {
        iotcon_request_get_observe_id(request,   &observe_id);
        iotcon_observers_remove(observers,   observe_id);
    }

    return  0;
}
```

client

```
int connectivity_observe_resource(…  ,  connectivity_observe_re
{
    …
    ret  =  iotcon_find_resource(IOTCON_MULTICAST_ADDRESS,
            IOTCON_CONNECTIVITY_IP  |  IOTCON_CONNECTIVITY_PREFER_UDP,
            query,
            _found_resource_cb,
            cb_info);
    …
}
static bool  _found_resource_cb(…)
{
    …
    iotcon_remote_resource_observe_register(info->resource,
            IOTCON_OBSERVE_IGNORE_OUT_OF_ORDER,
            NULL,
            _observe_cb,
            info);
    …
}
static void  _observe_cb(iotcon_remote_resource_h   resource,  iotcon_error_e   err,  int
sequence_number,  iotcon_response_h  response,  void *user_data)
{
    …
    iotcon_response_get_result(response,   &response_result);
    if (IOTCON_RESPONSE_OK  !=  response_result)  {
        _E("_on_response_observe   Response  error(%d)",  response_result);
        return;
    }

    iotcon_response_get_representation(response,   &repr);
    iotcon_representation_get_attributes(repr,   &attributes);
    iotcon_attributes_get_bool(attributes,   "opened",  &opened);

    resource_info->cb_info->cb(resource_info,  (void *)(int) opened,  resource_info-
>cb_info->user_data);
}
```

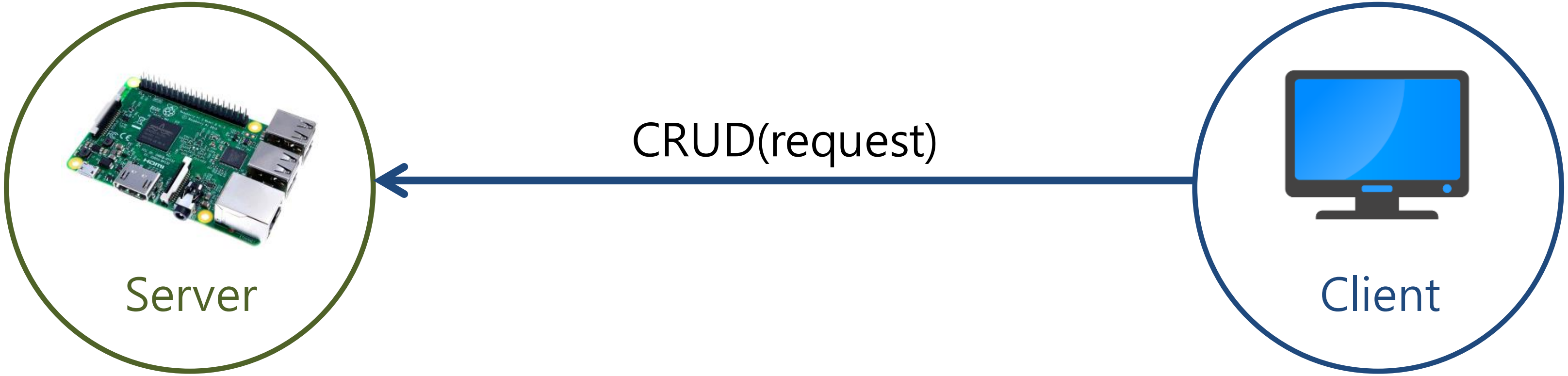
Get a data from the server

```
int connectivity_observe_resource(
        const char *type,
        connectivity_observe_resource_cb cb,
        void *user_data);
```

```
int connectivity_resource_pop_int(
        connectivity_resource_s *resource_info,
        const char *key,
        int *value)

int connectivity_resource_pop_double(
        connectivity_resource_s *resource_info,
        const char *key,
        double *value)

int connectivity_resource_pop_string(
        connectivity_resource_s *resource_info,
        const char *key,
        char **value)
```

Control a remote resouce

CRUD remote resources



CRUD(request)

Server

Client

CRUD remote resources

```
int iotcon_remote_resource_post (iotcon_remote_resource_h resource,
                                 iotcon_representation_h repr, iotcon_query_h query,
                                 iotcon_remote_resource_response_cb cb,
                                 void *user_data)
```

Create

```
int iotcon_remote_resource_get (iotcon_remote_resource_h resource,
                                iotcon_query_h query, void *user_data)
```

Read

```
int iotcon_remote_resource_put (iotcon_remote_resource_h resource,
                                iotcon_representation_h repr, iotcon_query_h query,
                                iotcon_remote_resource_response_cb cb, void *user_data)
```

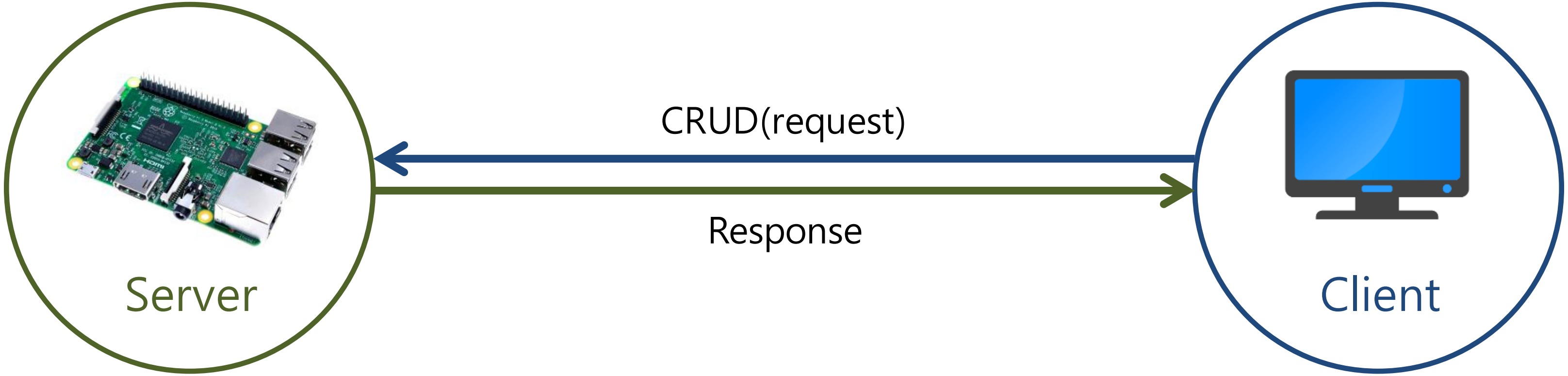
Update

```
int iotcon_remote_resource_delete (iotcon_remote_resource_h resource,
                                   iotcon_remote_resource_response_cb cb, void *user_data)
```

Delete

https://goo.gl/8HjgKf

Response



Response

```
int  iotcon_response_create (iotcon_request_h request, iotcon_response_h *response)
int  iotcon_response_destroy (iotcon_response_h resp)

int  iotcon_response_get_options (iotcon_response_h resp,  iotcon_options_h *options)
int  iotcon_response_get_representation (iotcon_response_h resp,  iotcon_representation_h *repr)
int  iotcon_response_get_result (iotcon_response_h resp,  iotcon_response_result_e *result)

int  iotcon_response_set_result (iotcon_response_h resp,  iotcon_response_result_e result)
int  iotcon_response_set_representation (iotcon_response_h resp, iotcon_representation_h repr)
int  iotcon_response_set_options (iotcon_response_h resp,  iotcon_options_h options)

int  iotcon_response_send (iotcon_response_h resp)
```

https://goo.gl/jc8SVV

server

```
static iotcon_representation_h _request_handler_get(door_resource_s
*door, iotcon_request_h request)
{
    iotcon_attributes_create(&attributes);
    iotcon_attributes_add_bool(attributes, "opened", door->attributes);

    iotcon_representation_create(&repr);
    iotcon_representation_set_uri_path(repr, door->uri_path);
    iotcon_representation_set_attributes(repr, attributes);

    iotcon_attributes_destroy(attributes);

    iotcon_response_create(request, &response);
    iotcon_response_set_result(response, IOTCON_RESPONSE_OK);
    iotcon_response_set_representation(response, repr);

    iotcon_response_send(response);

    iotcon_response_destroy(response);
    iotcon_representation_destroy(resp_repr);
}
```

client

```
static void _response_get_query(iotcon_remote_resource_h resource,
iotcon_response_h response, void *user_data)
{
    ...
    ret = iotcon_response_get_result(response, &response_result);
    ret_if(IOTCON_ERROR_NONE != ret);

    if (IOTCON_RESPONSE_OK != response_result) {
        _E("_response_get_query response error(%d)", response_result);
        return;
    }

    iotcon_remote_resource_get_host_address(resource, &resource_host);
    _I("Resource host : %s", resource_host);

    ret = iotcon_response_get_representation(response, &recv_repr);
    ret_if(IOTCON_ERROR_NONE != ret);

    ret = iotcon_representation_get_attributes(recv_repr, &recv_attributes);
    ret_if(IOTCON_ERROR_NONE != ret);

    ret = iotcon_attributes_get_bool(recv_attributes, "opened", &opened);
    ret_if(IOTCON_ERROR_NONE != ret);
    ...
}
```

THANK YOU

SOSCON
SAMSUNG
OPEN SOURCE
CONFERENCE