LWM2M-Client

Introduction

This API provides functionality to use LWM2M features from mbed device client which are described in Lightweight Machine to Machine Technical Specification. Using this API mbed OS developers can develop their own application utilizing LWM2M features.

This API provides enabler which defines the application layer communication protocol between a LWM2M Server and a LWM2M Client, which is located in a LWM2M Device. The OMA Lightweight M2M enabler includes device management and service enablement for LWM2M Devices. The target LWM2M Devices for this enabler are mainly resource constrained devices. Therefore, this enabler makes use of a light and compact protocol as well as an efficient resource data model. A Client-Server architecture is introduced for the LWM2M Enabler, where the LWM2M Device acts as a LWNM2M Client and the M2M service, platform or application acts as the LWM2M Server. There are four interfaces between these two components as shown below: - Bootstrap - Client Registration - *Device management and service enablement* - *Information Reporting*

This API currently covers Bootstrap and Client Registration, the other two features will be added in future releases of this API.

LWM2M Features

The API provides interface to the application developer to define their applications's endpoint infomation which will be delivered to the bootstrap server and mbed Device Server during bootstrap and registration operations respectively, This is how you can create your interface for your endpoint.

```
#include "lwm2m-client/m2minterfacefactory.h"
#include "lwm2m-client/m2minterface.h"

M2MInterface* interface = M2MInterfaceFactory::create_interface(*this,
                                                    "lwm2m-endpoint",
                                                    "test",
                                                    3600,
                                                    8000,
                                                    "",
                                                    M2MInterface::UDP,
                                                    M2MInterface::LwIP_IPv4,
                                                    "");
```

Once you have created an interface, you can now proceed to execute operation using this interface. ####Bootstrap feature The Bootstrap Interface is used to

provision essential information into the LWM2M Client to enable the LWM2M Client to perform the operation “Register” with one or more LWM2M Servers.

In this release only one bootstrap mode is supported: #####Client Initiated Bootstrap Client Initiated Bootstrap mode provides a mechanism for the LWM2M Client to retrieve the Bootstrap Information from a LWM2M Bootstrap Server. The Client Initiated Bootstrap mode requires having a LWM2M Boostrap Server Account. This API enables client initiated bootstrap functionality. User can provide the bootstrap server information and issue bootstrap command in following way.

First you need to create your bootstrap server object which contains information about bootstrap server like server address, security mode used by server etc.

```
#include "lwm2m-client/m2msecurity.h"
 M2MSecurity *security = M2MInterfaceFactory::create_security(M2MSecurity::Bootstrap);
        if(security) {
            security->set_resource_value(M2MSecurity::M2MServerUri, BOOTSTRAP_SERVER_ADDRESS
            security->set_resource_value(M2MSecurity::BootstrapServer, 1);
            security->set_resource_value(M2MSecurity::SecurityMode, M2MSecurity::NoSecurity)
        }
```

- Please note, currently this API supports only non-secure mode operations. Security features will be added in upcoming releases.

Once you have the bootstrap object ready, all you need to do is to call bootstrap API by passing this object as parameter.

```
M2MInterface::bootstrap(M2MSecurity* bootstrap_object);
```

Since, this is an asynchronous operation, you will receive the result of this operation through callback defined in `m2minterfaceobserver.h` which you should be handling in your application. If the bootstrap operation is successful and client is able to fecth the mbed Device Server information from bootstrap server, your application will receive following callback

```
void bootstrap_done(M2MSecurity *server_object)
```

The `server_object` contains the data for mbed Device Server including serevr Uri, security mode etc. Using this object you can execute registration operation for this client.

In case, the bootstrap operation fails for some reason , then you will receive following callback

```
void error(M2MInterface::Error error)
```

You can get to know more about the error from the `error` parameter which is passed with the callback and then act accordingly.

Client Registration Interface The Client Registration Interface is used by a LWM2M Client to register with LWM2M Servers, maintain registration and de-register from a LWM2M Server. Currently, only one-to-one client server regisration is supported. But, in upcoming releases client API will support one-to many client-server registrations.

Client registration interface includes multiple sub-features and currently supported are: - Register - Update - De-register

Register When registering, the LWM2M Client performs the "Register" operation and provides the properties the LWM2M Server requires to contact the LWM2M Client (e.g., End Point Name); maintain the registration and session (e.g., Lifetime, Queue Mode) between the LWM2M Client and LWM2M Server as well as knowledge of the Objects the LWM2M Client supports and existing Object Instances in the LWM2M Client

This API enables clientregistration functionality. User can provide the mbed Device server server information and issue register command in following way.

First you need to create your mbed Device server object which contains information about bootstrap server like server address, security mode used by server etc. If you have done bootstrap operation first, then this object is automatically created and available through `bootstrap_done()` callback which you can use directly.

```
#include "lwm2m-client/m2msecurity.h"
 M2MSecurity *security = M2MInterfaceFactory::create_security(M2MSecurity::LWM2M);
        if(security) {
            security->set_resource_value(M2MSecurity::M2MServerUri, LWM2M_SERVER_ADDRESS);
            security->set_resource_value(M2MSecurity::BootstrapServer, 0);
            security->set_resource_value(M2MSecurity::SecurityMode, M2MSecurity::NoSecurity)
        }
```

- Please note, currently this API supports only non-secure mode operations. Security features will be added in upcoming releases.

During registering your endpoint, you will also need to register all your resources that you would like to monitor or follow from mbed Deivce Server. This can be easily achieved by creating resource objects and passing them to register API for registration purposes.

For example, if you want to register your OMW M2M Device object , you need to simply create the device object and set the values for manadatory resources like this.

```
#include "lwm2m-client/m2mdevice.h"
```

```
M2MDevice *device = M2MInterfaceFactory::create_device();
if(device) {
    device->create_resource(M2MDevice::Manufacturer,MANUFACTURER);
    device->create_resource(M2MDevice::DeviceType,TYPE);
    device->create_resource(M2MDevice::ModelNumber,MODEL_NUMBER);
    device->create_resource(M2MDevice::SerialNumber,SERIAL_NUMBER);
    }
```

You can register other resources as well, please check detailed API documenttion for that.

Apart from manadatory device obejct, if you would like to register your own customized resources , you can create them and set their values accordingly. For doing that, please check the API documentation for M2MObject, M2MObjectInstance and M2MResource classes.

Once you have the registration server object and resources, which you want to register, ready, all you need to do is to call register API by passing these object as parameters.

```
M2MInterface::register_object(M2MSecurity* register_object, M2MObjectList object_list);
```

Since, this is an asynchronous operation, you will receive the result of this operation through callback defined in `m2minterfaceobserver.h` which you should be handling in your application. If the register operation is successful and client is able to register all your resources to the mbed Device Server information , your application will receive following callback

```
void object_registered(M2MSecurity *server_object, const M2MServer& server)
```

The `M2MSecurity *server_object` informs to which mbed Device Server the client has just registered and `M2MServer &server` contains the data related with mbed Device Server including Short ServerID, client registration period etc.

In case, the registration operation fails for some reason , then you will receive following callback

```
void error(M2MInterface::Error error)
```

You can get to know more about the error from the `error` parameter which is passed with the callback and then act accordingly.

Update Periodically or based on certain events within the LWM2M Client or initiated by the LWM2M Server, the LWM2M Client updates its registration information with a LWM2M Server by sending an "Update" operation to the LWM2M Server. This is how you can de-update your registration.

```
M2MInterface::update_registration(M2MSecurity* security_object, const uint32_t lifetime)
```

Normally, the enabler will take care of updating the registration automatically but if you want to renew the regisrtaion before that , you can use this API.

If the update operation is successful, your application will receive following callback

```
void registration_updated(M2MSecurity *const M2MServer& server)
```

The `M2MSecurity *server_object` informs to which mbed Device Server the client has just updated the registration and `M2MServer &server` contains the data related with mbed Device Server including Short ServerID, client registration period etc.

In case, the updation operation fails for some reason , then you will receive following callback

```
void error(M2MInterface::Error error)
```

De-register When a LWM2M Client determines that it no longer requires to be available to a LWM2M Server (e.g., LWM2M Device factory reset), the LWM2M Client de-register from the LWM2M Server. Upon receiving this message, the LWM2M Server removes the registration information from the LWM2M Server. This is how you can de-register your endpoint client. In case, endpoint has multiple server regsitration then you need to provide the server_object of the server where you would like to de-register your endpoint. Otherwise if there is only one registration then you can pass NULL and client will unregister the default registration from endpoint.

```
M2MInterface::unregister_object(M2MSecurity *object);
```

Since, this is an asynchronous operation, you will receive the result of this operation through callback defined in `m2minterfaceobserver.h` which you should be handling in your application.

If the de-register operation is successful and client is successfully unregistered from mbed Device Server information , your application will receive following callback

```
void object_unregistered(M2MSecurity *server_object)
```

The `M2MSecurity *server_object` informs to which mbed Device Server the client has just de-registered.

In case, the de-registration operation fails for some reason , then you will receive following callback

```
void error(M2MInterface::Error error)
```

You can get to know more about the error from the `error` parameter which is passed with the callback and then act accordingly.

Device Management and Service Enablement Interface *Not yet supported.* #####Information Reporting Interface *Not yet supported.*

More Information

This API is based on OMA LWM2M specification. You can get specification from here

API documentation

You can generate Doxygen API documentation for these APIs from doxy file which is present in `cd doxygen` folder . You need to run `doxygen` command from the `doxygen/` folder and it will generate `docs` folder at API source directory root level where you can find the detailed documentation for each API.

Example application

In order to use these API, there is an example application available in `lwm2m-client-example` directory.