



OPEN CONNECTIVITY
FOUNDATION®

ToolChain

IoTivity-Lite Code explained





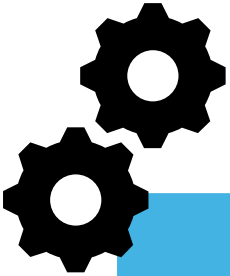
The Generated IoTivity-Lite code explained

IoTivity-Lite API

- Main
- Supported functions
- Global variables for each resource
- Handler implementation for each resource-method




Typical stages, setup and running of the stack



initialize

- stack
- build in resources
- application specific resources



Run (wait on callback)

- GET: create response
- POST:
 - Check input
 - Process (assign) input
 - Create response



IOivity-lite build in resources

- Set of APIs to create an OCF device (or client)
- Has a set of “build-in” resources:
 - oic/res
 - oic/p
 - oic/d
 - Security resources
 - introspection

The entity handler of these resources will be handled by the stack.



Main

- Main
 - Starts the platform
 - Register the device and platform, e.g. initializes oic/d and oic/p
 - Create all application specific resources
 - Message pump
 - A loop that handles the incoming messages, e.g. GET and POST
 - Calls the installed callbacks for each resource.
 - This loop makes sure that all access to the functions/global variables are not concurrent.



Register device (app_init)

app_init function sets the device (oic/d) information like:

- Device type
- Device name
- Data model versions

No specific things that is being set the platform

- Only name



Register resources

register_resources function sets for each endpoint:

- Resource type
- Interface, including the default interface
- Discoverable
- Observable
- And the GET/POST request handlers



Resource global variables

Each end point has a set of variables:

- The property name
 - naming convention:
g_<path>_RESOURCE_PROPERTY_NAME_<propertyname>
- The actual value of the property, which is typed from the json data type
 - naming convention: g_<path>_<propertyname>
- The path in a variable:
 - naming convention: g_<path>_RESOURCE_ENDPOINT
- Array of interfaces, where by the first will be set as default interface
 - naming convention g_<path>_RESOURCE_INTERFACE



Handling the Retrieve (GET) operation

Each endpoint can have an Retrieve (GET) operation:

- Naming convention: `get_<path>`
- Function returns the payload when “coap-get” is called
- Function adds the global variables of that specific endpoint to the payload that will be returned

Implementation specific:

before assigning the member variables to the payload, one can update the member variables from the HW.



Handling the Update (POST) operation

Each endpoint can have an UPDATE function:

- Naming convention: `post_<path>`
- Function that interprets the payload when “coap-post” is called
- Function checks if the properties are correct and within the limits.
- If all properties are correct, then the values of the post are assigned to the global variables for that specific endpoint

Implementation specific:

after assigning the value to the member variables, one can interact with the HW, using the newly assigned member variables.



OPEN CONNECTIVITY
FOUNDATION®