OPEN CONNECTIVITY FOUNDATION®

# ToolChain

## Code explained

# The Generated C++ IOTivity code explained

IOTivity C++ API

- Main
- Supported functions
- Class implementation for each resource

Note IOTivity also has a C API

# IOTivity 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

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

# main

- Main
  - Starts the platform
  - Register the device, e.g. initializes oic/d
  - Register the platform, e.g. initializes oic/p
  - Create all application specific resources
  - Just loops until a resource is being called

# Register device

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

- Device type
- Device name
- Data model versions

# Register platform

setPlatformInfo function sets the platform (oic/p) information like:

- platformID
- Manufactorer: name, URL
- Hardware version
- Firmware version
- System time

This resource includes optional properties, hence this resource needs to be listed in the introspection file

# IoTServer class

The IoTServer is a wrapper that list all application specific resources.

- This class instantiates all application specific resources

- This class is (indirectly) responsible for populating all application specific resources in oic/res.

# Resource class

The Resource class is a virtual class being used by each specific resource class.

- Just to have less code

# Resource specific class

The resource specific class is a class representing a specific endpoint

- Each endpoint, even of the same type, has its own class
  - So that it can interact with specific hardware
- Each class has :
  - Set of member variables
    - E.g. the list of properties that can occur in the payloads
  - Constructor/Destructor
  - get function
    - E.g. the OCF Retrieve method
  - post function (if needed in the implementation)
    - E.g. the OCF Update method

# Resource specific class - constructor

- Initializes all member variables
  - From examples/defaults in the resource definitions
- Initializes the call backs (get/post)
- Register the resource
  - With the appropriate resource types (rt)
  - With the appropriate interface (if)
  - With the appropriate flags (secure, observable, discoverable)
  - With the specific URI to receive the call-backs for.

# Resource specific class – get function

- Function registered in the constructor
- Function that returns the payload when “coap-get” is called
- Function adds the member variables 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.

# Resource specific class – post function

- Function registered in the constructor
- 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 member variables.

Implementation specific:

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

# What to do next

- Define your own input file
- Run the tooling!
- Build it… and see if it works
  - For example: Check it against OTGC/CTT
  - To get an feeling what is there..
- Change the code, to what you want it to do

OPEN CONNECTIVITY
FOUNDATION®