

Tizen/Artik IoT Lecture Chapter 6. OCF Resource Type & IoTivity Simulator

Sungkyunkwan University

RESTful API Modeling Language (RAML)

2

8

- **RAML makes it easy to manage the whole API lifecycle from design to sharing**
 - <http://raml.org>
- **Machine readable API design that is *human friendly***
- **Support several languages**
 - node.js, Java, .NET, Python
- **The RAML definition are used to describe the payloads of the CRUDN operations**
 - Used in simulation
 - Validating of responses

```
1  #%RAML 1.0
2  title: World Music API
3  baseUrl: http://example.api.com/{version}
4  version: v1
5
6  uses:
7    Songs: !include libraries/songs.raml
8
9  annotationTypes:
10    monitoringInterval:
11      parameters:
12        value: integer
13
14  traits:
15    secured: !include secured/accessToken.raml
16
17  /songs:
18    is: secured
19    get:
20      (monitoringInterval): 30
21      queryParameters:
22        genre:
23          description: filter the songs by genre
24    post:
25      /{songId}:
26        get:
27          responses:
28            200:
29              body:
30                application/json:
31                  type: Songs.Song
32                application/xml:
33                  schema: !include schemas/songs.
34                  example: !include examples/song
```

CRUDN Operation Response Codes

3

8

- **A resource can be created or updated depending on the resource type's definition and allowed CRUDN operations**
- **The operation may have different response code with different meaning**

| Response Code | Meaning |
|---------------|---|
| 200 | Payload of the response will confirm the change |
| 201 | Payload is the URL of the Resource that was created by the server as a result of a CREATE operation |
| 204 | OK, everything went well, no payload provided |
| 403 | <i>CASE1:</i> On RETRIEVE, the server does not support the values provided <i>CASE2:</i> the server could not CREATE or UPDATE the Resource due to a problem with the provided payload |

OIC defined Resource Types

4

8

- **There are many resource types defined by OIC**
 - 62 resource types are defined
- **RAML and JSON set**
 - Acceleration.raml
 - oic.r.sensor.acceleration.json

Table 6-1 Alphabetical list of Resource Types

| Friendly (informative) | Name | Resource Type (rt) | Section |
|------------------------|----------|----------------------------------|-----------|
| Acceleration Sensor | | oic.r.sensor.acceleration | 6.52 |
| Activity Count | | oic.r.sensor.activity.count | 6.23 |
| Altimeter | | oic.r.altimeter | 6.57 |
| Atmospheric Pressure | | oic.r.sensor.atmosphericpressure | 6.24 |
| Air Flow | | oic.r.airflow | 6.1 |
| Air Flow Control | | oic.r.airflowcontrol | 6.2 |
| Audio Controls | | oic.r.audio | 6.25 |
| Auto Focus | | oic.r.autofocus | 6.26 |
| Automatic Feeder | Document | oic.r.automaticdocumentfeeder | 6.27 |
| Auto White Balance | | oic.r.colour.autowhitebalance | 6.31 |
| Basic Resource Schema | | Not Applicable | Annex A.1 |

Example: Acceleration

5

8

- **URI:** /AccelerationResURI
- **Rt:** oic.r.sensor.acceleration

JSON definition: oic.r.sensor.acceleration.json

```
1- {
2  "id": "http://openinterconnect.org/iotdatamodels/schemas/oic.r.sensor.acceleration.json#",
3  "$schema": "http://json-schema.org/draft-04/schema#",
4  "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
5  "title": "Acceleration Sensor",
6  "definitions": {
7    "oic.r.sensor.acceleration": {
8      "properties": {
9        "acceleration": {
10         "type": "number",
11         "description": "ReadOnly, sensed acceleration experienced in 'g'."
12       }
13     }
14   },
15   "type": "object",
16   "allOf": [
17     {"$ref": "oic.core.json#/definitions/oic.core"},
18     {"$ref": "oic.baseResource.json#/definitions/oic.r.baseresource"},
19     {"$ref": "#/definitions/oic.r.sensor.acceleration"}
20   ],
21   "required": ["acceleration"]
22 }
23
24
```

| | | |
|------------|-----------|--|
| Read | oic.if.r | It allows only permission to read. |
| Read Write | oic.if.rw | It allows to both read and write. |
| Actuator | oic.if.a | It includes creating, updating and retrieving actuator values. |
| Sensor | oic.if.s | It allows only reading the sensor values. |

| Name | Interface | |
|-------------|-----------------|--|
| Baseline | oic.if.baseline | It includes all the information about the resource including meta-data and collection information about the resources. This is the default interface type. |
| Linked List | oic.if.ll | It includes only the collection information about the resources. This is the default interface type for oic/res. |
| Batch | oic.if.b | It allows aggregating interaction with all resources. Each resource will be interacted separately, and response from them will be aggregated. |

RAML definition: Acceleration.raml

```
1- ##RAML 0.8
2  title: OICAcceleration
3  version: v1.1.0-20160519
4  documentation:
5    - title: Copyright (c) 2016 Open Connectivity Foundation, Inc.
6    - content: |
7      Redistribution and use in source and binary forms, with o
8      1. Redistributions of source code must retain the above
9      2. Redistributions in binary form must reproduce the abo
10
11    THIS SOFTWARE IS PROVIDED BY THE Open Connectivity Founda
12  schemas:
13    - acceleration: !include oic.r.sensor.acceleration.json
14  traits:
15    - interface:
16      queryParameters:
17        if:
18          enum: ["oic.if.s", "oic.if.baseline"]
19
20  /AccelerationResURI:
21    description: |
22      This resource provides a measure of proper acceleration (g fo
23      (which is dependent on the co-ordinate system and the observe
24      The value is a float which describes the acceleration experie
25
26    displayName: Acceleration Sensor
27    is: [ interface ] # valid for all methods
28
29    get:
30      responses:
31        200:
32          body:
33            application/json:
34              schema: acceleration
35              example: |
36                {
37                  "rt": ["oic.r.sensor.acceleration"],
38                  "id": "unique_example_id",
39                  "acceleration": 0.5
40                }
41
```

IoTivity Simulator: Service Provider

6

8

- **IoTivity Simulator provides two functions**
 - Service provider
 - Client controller

Two ways to create resource

Create resources

Create Resource

Create a Simple resource

☒ Simple resource

☐ Simple resource(From RAML)

Details:

Create a simple resource by manually entering all the details given below.

1. Basic resource details: URI, Name, Resource Type, and Interface types.
2. Options such as Observable and Discoverable.
3. Adding simple attributes.

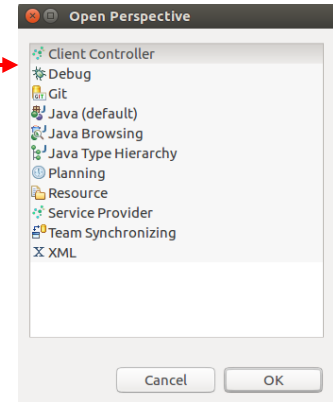
< Back Next > Cancel Finish

Create resource manually

Failed to create resource with raml

Test Automation

| Severity | Time | Message |
|----------|--------------|---|
| Info | 17:34:17.446 | Resource created [/testBulb] |
| Error | 17:34:38.186 | Failed to create resource! |
| Error | 17:35:48.871 | Failed to create resource! |
| Info | 17:36:05.946 | Resource automation started [URI: "/testBulb", id: 0] |



IoTivity Simulator: Client Controller

7

8

- Find resource at the client side
- You can use **CRUDN** operations
 - GET, PUT, POST, OBSERVE, AUTOMATION

The screenshot displays the 'workspace - Client Controller - Eclipse' interface. On the left, the 'Find resources' dialog is open, showing 'Resource Type' set to 'All' and 'Resource Types' as 'Ex: sample.light, hall.fridge'. A red arrow points from the 'Find Resources' button in this dialog to the 'Find Resources' button in the 'Resource Manager' pane. The 'Resource Manager' pane shows a table of 'Found Resources' with one entry: '/testBulb'. The 'Attribute Manager' pane shows a table of 'Attributes' with one entry: 'temp' with value '75'. The 'Properties' pane shows a table of 'Properties' with entries for 'Resource URI', 'Address', 'Connectivity Type', 'Observable', 'Resource Types', and 'Resource Interfaces'. The 'Simulator Log' pane shows a list of log messages with columns for 'Severity', 'Time', and 'Message'. On the right, two dialog boxes are shown: 'Generate PUT Request' and 'Generate POST Request'. The 'Generate PUT Request' dialog shows 'Interface Type' as 'Baseline (oic.if.baseline)', 'Name' as 'temp', and 'Value' as '50'. The 'Generate POST Request' dialog shows 'Interface Type' as 'Baseline (oic.if.baseline)', 'Name' as 'temp', 'Value' as '75', and 'Select' as 'POST'.

Find resources
Find Resources
Select the resource type of the resources to be discovered

Resource Type
☒ All
☐ Specific Resource types (seperated by commas)
Resource Types: Ex: sample.light, hall.fridge

Cancel OK

Resource Manager
Find Resources Refresh
Found Resources Favorite Resources
type filter text
/testBulb

Attribute Manager
Attributes
Attribute Name Attribute Value
temp 75

GET PUT POST Observe Automation

Properties
Default Device Platform
Property Value
Resource URI /testBulb
Address coop://[fe80::804f51...]
Connectivity Type SIMULATOR_CT_DEF
Observable Yes
Resource Types eslab.test.bulb
Resource Interfaces oic.if.baseline

Simulator Log
type filter text
Severity Time Message
Info 17:51:43.453 Response received for GET.
Info 18:23:29.478 Sending GET request.
Info 18:23:29.967 Response received for GET.
Info 18:24:18.598 Sending PUT request.
Info 18:24:18.656 Response received for PUT.
Info 18:24:47.705 Sending POST request.
Info 18:24:47.738 Response received for POST.

Generate PUT Request
Dialog which takes input and generates a put request.
Interface Type Baseline (oic.if.baseline)
Name Value
temp 50
Cancel PUT

Generate POST Request
Dialog which takes input and generates a post request.
Interface Type Baseline (oic.if.baseline)
Name Value Select
temp 75 POST
Cancel POST

IoTivity Simulator: Observe

8

8

- During OBSERVE, provider can notify to client

The screenshot shows the Eclipse IDE workspace for the Client Controller. The Attribute Manager displays a single attribute named 'temp' with a value of 75. The Properties view shows details for the resource URI, address, connectivity type, and observable status. The Simulator Log shows a sequence of messages including GET, PUT, POST, and OBSERVE requests and responses. A red box highlights the 'Stop Observe' button in the Simulator Log.

| Severity | Time | Message |
|----------|--------------|--|
| Info | 18:23:29.967 | Response received for GET. |
| Info | 18:24:18.598 | Sending PUT request. |
| Info | 18:24:18.656 | Response received for PUT. |
| Info | 18:24:47.705 | Sending POST request. |
| Info | 18:24:47.738 | Response received for POST. |
| Info | 18:27:18.944 | [URI: /testBulb] Sent OBSERVE request. |
| Info | 18:27:19.084 | Response received for OBSERVE request. |

The screenshot shows the Eclipse IDE workspace for the Service Provider. The Resource Manager displays a single resource named 'testBulb'. The Properties view shows details for the resource URI, address, connectivity type, and observable status. The Simulator Log shows a sequence of messages including PUT, POST, and GET requests and responses. A red box highlights the 'Notify' button in the Resource Manager.

| Severity | Time | Message |
|----------|--------------|---|
| Info | 18:24:18.616 | [/testBulb] PUT request received. |
| Info | 18:24:18.636 | [/testBulb] Sent response for PUT request |
| Info | 18:24:47.719 | [/testBulb] POST request received. |
| Info | 18:24:47.719 | [/testBulb] Sent response for POST request |
| Info | 18:27:18.946 | [/testBulb] GET request received. |
| Info | 18:27:18.961 | [/testBulb] Sent response for GET request |
| Info | 18:27:19.064 | [/testBulb] Observer added [id: 102, address: fe80::804:f511:8705:8066, port: 49704]. |