1 CONTENTS

2 Notification Manager (NM)........................................................................................................... 3
3 Enhancements.................................................................................................................................. 3
3.1 Notification Manager SDK.......................................................................................................... 3
3.2 Requesting Resource Hosting .................................................................................................... 3
3.3 Virtual Resource Synchronization ............................................................................................... 3
4 Terminology ..................................................................................................................................... 3
4.1 Unified / Lite Device..................................................................................................................... 3
4.2 Virtual Resource .......................................................................................................................... 3
4.3 Hosting Device ............................................................................................................................ 4
4.4 Resource Hosting ........................................................................................................................ 4
5 Notification Manager SDK.............................................................................................................. 4
5.1 Notification Manager APIs .......................................................................................................... 4
5.2 Resource Hosting Functional Diagram ....................................................................................... 5
6 Resource Hosting Operation .......................................................................................................... 5
6.1 Resource Hosting Flow Diagram ................................................................................................ 5
7 Example........................................................................................................................................... 6
7.1 Notification Manager Tizen Sample Application ........................................................................ 6
2 **Notification Manager (NM)**

This guide provides interaction details and the simple programming model between the Unified device and Lite device respectively. The purpose is to provide enough detail to a developer to understand the resource hosting features and operations.

The guide is divided into two parts. It would be described the API of resource hosting functions and functional block diagram of resource hosting in the first part. In the second part, resource hosting operation between Unified device and Lite device will be described according to the given sample source code.

3 **Enhancements**

In subsequent releases, the IoT Notification Manager Service functionalities will be enhanced to include some of other important features as follows:

3.1 **Notification Manager SDK**

APIs for starting and stop the resource hosting functionalities are exposed.

3.2 **Requesting Resource Hosting**

Registering resources on Lite device which is requesting “Resource Hosting” to Notification Manager by use of hosting identifier.

3.3 **Virtual Resource Synchronization**

An Iotivity Device can provide the state (create/read/update/delete) synchronization between the virtual resource and the original resource.

4 **Terminology**

4.1 **Unified / Lite Device**

- Unified Device : A device which could act as a hosting device. This device does not have any resource constraints.
- Lite Device : A device which has the resource constraints, e.g., small size of memory, limited energy sources.

4.2 **Virtual Resource**

A virtual resource is a resource that reflects the resource status of remote device. Hosting device creates and registers the virtual resources for other devices to consume via the hosting device while hiding the original resource.
4.3 Hosting Device

A device which creates and registers virtual resource. Hosting device observes the resource of origin server and provides virtual resource for IoTivity clients. The clients could access this resource as same as the origin resource.

4.4 Resource Hosting

A feature which stores(caches) only resource(s) data with new address:port info (as in Web Mirroring). The goal of this feature is to off-load the request handling works from the resource server where original resource is located.

5 Notification Manager SDK

5.1 Notification Manager APIs

These APIs provide methods for application to start and stop the coordinator and also to initialize and terminate resource hosting. The operations provided in the SDK are listed below:

- OICStartCoordinate
- OICStopCoordinate

OICStartCoordinate API can be used to start the resource coordinator. It will create mirrorResourceList and start to discover the coordinated candidates.

**Prototype:**

- OCStackResult OICStartCoordinate ();

**Return Value:**

- Returns OC_STACK_OK on Success, OC_STACK_ERROR on Failure.

OICStopCoordinate API can be used to stop the resource coordinator. This function will stop the resource hosting and delete mirrorResourceList used.

**Prototype:**

- OCStackResult OICStopCoordinate ();

**Return Value:**

- Returns OC_STACK_OK on Success, OC_STACK_ERROR on Failure.
5.2 RESOURCE HOSTING FUNCTIONAL DIAGRAM

Notification Manager has four functional blocks for resource hosting feature. Resource Virtualization creates and registers the virtual resources which reflects the status of the origin resource server. Presence Detection block checks the reachiability of origin resources and the remote requests from resource clients are handled by the Remote Request Handler. The resources between origin server and hosting device are synchronized via Resource Synchronization block.

![Resource Hosting Functional Block Diagram](image)

Figure 1. Resource Hosting Functional Block Diagram

6 RESOURCE HOSTING OPERATION

6.1 RESOURCE HOSTING FLOW DIAGRAM

![Resource Hosting Flow Diagram](image)

Figure 2. Resource Hosting Flow Diagram

Lite Device needs to register resource which want to be hosted, and starts presence to IoTivity Base. And then, Hosting Device finds the resource and receives the presence of Lite Device with resource information. After finding the hosting resources, Hosting Device creates and registers the virtual resource and reflects the status of the origin resource. With the flow, the client finds and consumes the virtual resources instead of observing the origin resource.
7 Example

7.1 Notification Manager Tizen Sample Application

- Pre-requisite
  - Run `sampleprovider`, Linux applications to test the Hosting and VirtualResource APIs.
  - After launching the Tizen application (NMSampleApp), run `sampleconsumer`, Linux applications for complete two way testing.

Running all the Sample Apps:

- Steps to run `sampleprovider`(Linux)
  - Export `LD_LIBRARY_PATH` to “<iotivity>/out/linux/<arch>/release”
  - Run the `sampleprovider` application as shown below

```bash
<iotivity>/out/linux/<arch>/release/service/notification-manager/SampleApp/linux $./sampleprovider
```

The following logs will be shown

```
1. Temp is up
2. Temp is down
3. This Program will be ended.
```

- Run the NMSampleApp(Tizen)

![Notification Manager](image)

- App description
  - Above given is the home screen of the application. Platform configurations will be done on the launch of the application by default.
  - On clicking the **Find and Host** button, the notification manager service will start completely.
  - **Stop** button will terminate the notification manager service completely. Once the service is terminated it can be restarted using the Find and Host button on the same screen.
Once user presses “Find and Host” button the corresponding logs can be seen on linux and tizen Applications.

**Sampleprovider (Linux)**

```
0:
In entity handler wrapper:
Sample Provider entityHandler flag : request
/a/TempHumSensor/hosting

Receive ObserverFlag : Start Observe
requestFlag : Observer
pthrerad_create
CheckLightRepresentation Enter
pthread_cond_wait
```

**NMSampleApp (Tizen)**

```
OCResourceHosting started
Interface Name : wlan0
Starting OIC resource hosting on 192.168.1.11
OICStartCoordinate done successfully
```

- **Steps to run sampleconsumer (Linux)**
  - Export `LD_LIBRARY_PATH` to “<iotivity>/out/linux/<arch>/release”
  - Run the `sampleconsumer` application as shown below

```
<iotivity>/out/linux/<arch>/release/service/notification-manager/SampleApp/linux$./sampleconsumer
```

**NOTE**: `<iotivity>` is the path to 'iotivity' project.

The following logs will be shown in `sampleconsumer`
The following logs will be shown in **sampleprovider**
Now, if user enter 1 in `sampleconsumer` Application, it will send a observer request to `NMSampleApp` (as it is hosting the resource);

Following logs will be updated on `sampleconsumer` Application:

```
1
request for new observation
onObserve

Receive OBSERVE RESULT:
Url: /a/TempHumSensor
SequenceNumber: 0
Temperature: 0
Humidity: 0
```

If user enter 2 in `sampleconsumer` Application, it will send a get request to `NMSampleApp` (as it is hosting the resource).

Following logs will be updated on `sampleconsumer` Application:

```
2
URI : /a/TempHumSensor
GET request was successful
GET request was successful
Resource URI: /a/TempHumSensor
  Temperature: 0
  Humidity: 0

*****************************************************************************
  * method Type : 1 - Observe    *
  * method Type : 2 - Get        *
  * method Type : 3 - Put        *
  * method Type : 4 - Delete     *
*****************************************************************************
```

If user enter 3 in `sampleconsumer` Application, it will send a put request to `NMSampleApp` (as it is hosting the resource).

Following logs will be updated on `sampleProvider` Application:

```
8:
In entity handler wrapper:
Sample Provider entityHandler
flag : request

requestFlag : Request
  requestType : PUT
  temperature: 25
  humidity: 10
```
If user enter 4 in `sampleconsumer` Application, it will send a delete request to `NMSampleApp` (as it is hosting the resource).

Following logs will be updated on `sampleProvider Application`:

```plaintext
0: In entity handler wrapper:
Sample Provider entityHandler
flag : request
   requestFlag : Request
   requestType : DELETE
DeviceResource Delete Request
0: In entity handler wrapper:
Sample Provider entityHandler
flag : request
   requestFlag : Request
   requestType : GET
0: In entity handler wrapper:
Sample Provider entityHandler
flag : request
   requestFlag : Request
   requestType : GET
0: In entity handler wrapper:
Sample Provider entityHandler
flag : request
   requestFlag : Request
   requestType : GET
0: In entity handler wrapper:
Sample Provider entityHandler
flag : request
   requestFlag : Request
   requestType : GET
Success DELETE
```

*Note*: No direct communication will be there between `sampleprovider` and `sampleconsumer` as `NMSampleApp` is hosting the resource.