IoTivity Programmer's Guide – Notification Manager for Linux

1 CONTENTS

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 resouces on Lite device which is reqeusting “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 reflecting the resource status of remote lite device or unified 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

Notification Manager API provides resource hosting start and stop functions for application which wants to host the Lite device's resource. There are two APIs for resource hosting functions in Notification Manager.

OICStartCoordinate () starts the resource hosting function. With use of this API, Notification Manager creates mirrorResourceList and discovers resource candidates for hosting.

- Int OICStartCoordinate( )
- Returns :
  OC_STACK_OK : no errors
  OC_STACK_ERROR : Otherwise error (initialized value)

OICStopCoordinate() stops the resource hosting function. This API deletes mirrorResourceList used during resource hosting operation.

- Int OICStopCoordinate( )
- Returns :
  OC_STACK_OK : no errors
  OC_STACK_ERROR : Otherwise error (initialized value)

5.2 RESOURCE HOSTING FUNCTIONAL DIAGRAM

Notification Manager has four functional blocks for resource hosting feature. Resource Virtualizaton 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.

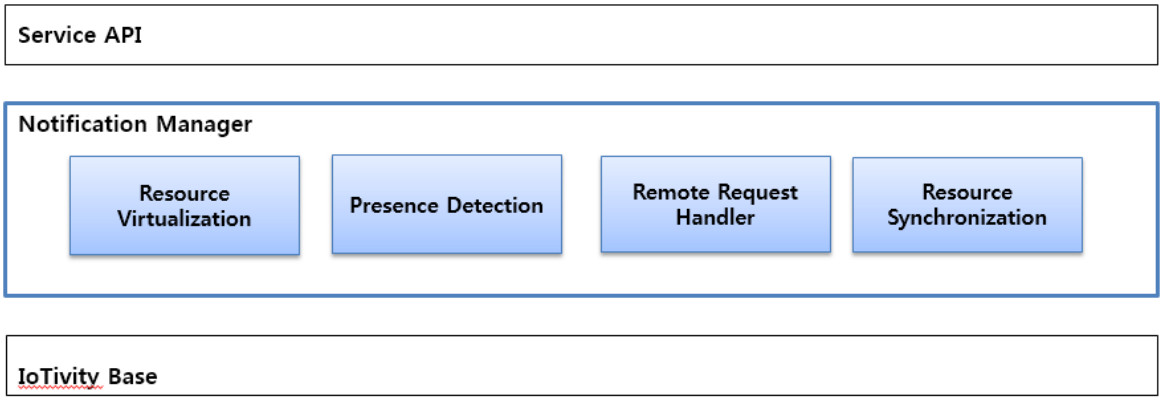

Figure 1. Resource Hosting Functional Block Diagram

6 RESOURCE HOSTING OPERATION

6.1 RESOURCE HOSTING FLOW DIAGRAM

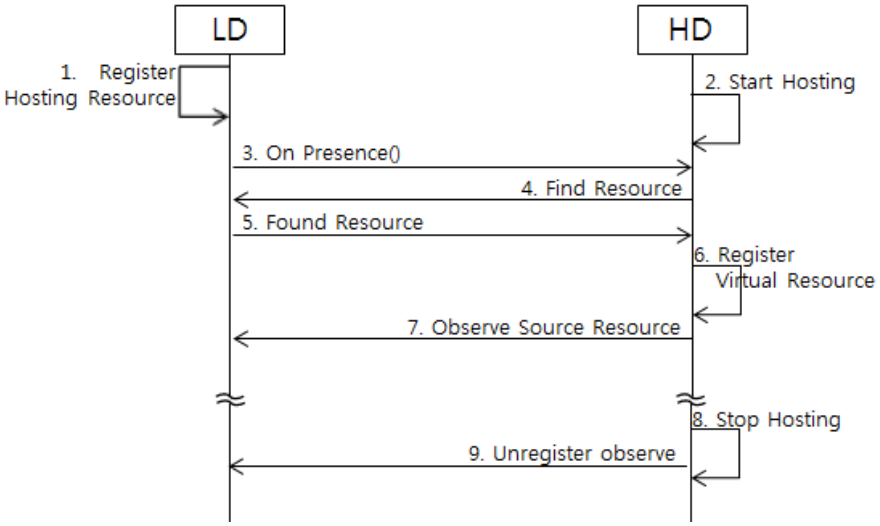

Figure 2. Resource Hosting Flow Diagram

Lite Device needs to register resource 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.

6.2 REQUEST RESOURCE HOSTING IN LITE DEVICE

To request resource hosting in lite device, the hosting flag should be added during resource registration.

```
int main(void)
{
            … Initialize IoTivity Base …
    startPresence(30);
    OCResourceHandle resourceHandle;
    std::string resourceURI = "/a/TempHumSensor";
    std::string resourceTypeName = "Resource.Hosting";
    std::string resourceInterface = DEFAULT_INTERFACE;
    uint8_t resourceProperty = OC_DISCOVERABLE | OC_OBSERVABLE;

    std::string requestHostingURI = resourceURI + "/hosting";
    OCStackResult result = OCPlatform::registerResource(resourceHandle , requestHostingURI ,
             resourceTypeName , resourceInterface , &entityHandler , resourceProperty);
          … Something To Do …
}
```

6.3 STARTING RESOURCE HOSTING IN HOSTING DEVICE

If you want to run resource hosting, IoTivity Base needs to be configured as "OC_CLIENT_SERVER" mode.

Hosting device starts to check presence from resource server and to create, register virtualized resource which is reflecting original resource.

```
int main(void)
{
    OCInit(<ipaddress> , <portnumber> , OC_CLIENT_SERVER);
    OCStackResult result = OICStartCoordinate();
    If(result != OC_STACK_OK)
    {
        Printf("Try to start resource hosting : Fail\n");
    }

          … Something To Do …
}
```

6.4 STOP RESOURCE HOSTING

Hosting device releases the whole virtual resource data and states when it stops resource hosting.

```
int main(void)
{
            … Something To Do …

    OCStackResult result = OICStopCoordinate();
    If(result != OC_STACK_OK)
    {
        Printf("Try to stop resource hosting : Fail\n");
    }

            … Something To Do …
}
```