Tizen/Artik IoT Lecture Chapter 10. IoTivity Resource Encapsulation

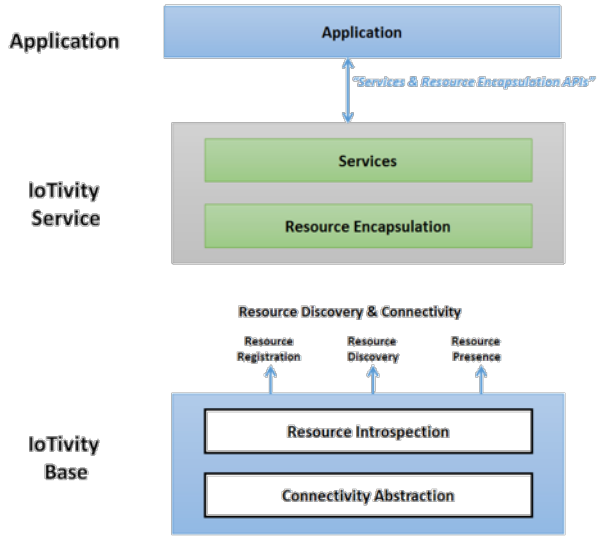
Sungkyunkwan University

Contents

- **Resource Encapsulation**
  - Components
  - Resource Client API: Key Objects
- **Resource Encapsulation Components**
  - Discovery Manager
  - Resource Broker
  - Resource Cache
  - Server Builder
- **RE Layer API**
- **Sample: Resource Client & Server**

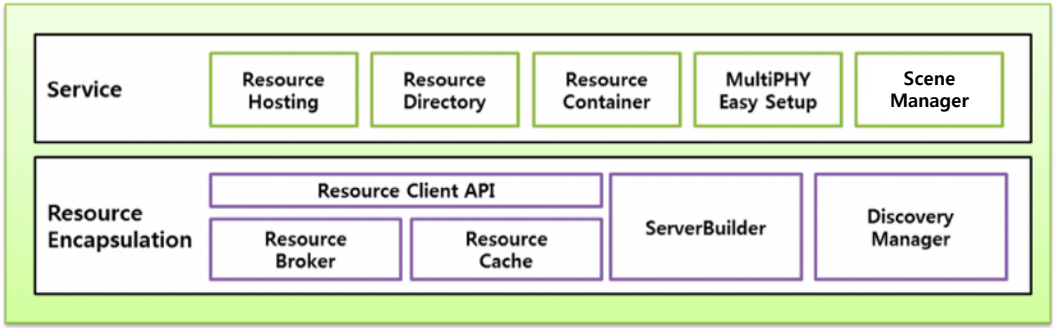
Resource Encapsulation

- **Abstract layer on common resource functions**
  - Library that encapsulates repeated client/server pattern
  - Base API is too difficult to use for IoTivity service developers
- **Service Layer**
  - Resources representing service features
  - Implemented with RE Layer functions
- **RE(Resource Encapsulation) Layer**
  - Encapsulate **operation patterns** on resources
- **Base Layer**
  - Resource Model (URI + CRUDN)



Resource Encapsulation: Components

- **Services are implemented by RE layer components**
- **Client-side**
  - Modules to ease the access & management on remote device's resources
  - Discovery Manager, Resource Broker, Resource Cache, Resource Client API
- **Server-side**
  - API to simplify to registration & management of resources
  - Server Builder



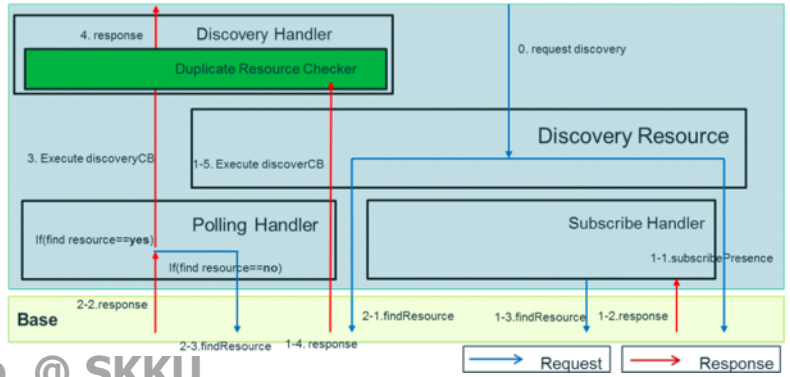
Resource Client API: Key Objects

- **RCSRemoteResourceObject (Client-side)**
  - Object representing a remote resource
  - Unit of presence monitoring, caching operations
- **RCSResourceObject (Server-side)**
  - Object representing present device's local resource
  - Unit of discoverable resource
  - Implementing getter/setter functions of RCSResourceObject results in making a server.

Discovery Manager

- **Module to discover other device's resources**
- **Support Advertising/Non-advertising Resources**
  - Contains 'subscribe handler' and 'polling handler'
  - In case of non-advertising resource: It polls in the period of 60 seconds until app issues cancel polling command.
- **APIs**
  - `discoverResource()`: discover any resources or resources with specific URI
  - `discoverResourceByType()`: discover resources with specific type

Type	Server-side	Client-side
Advertising	Advertise Resource	Subscribe
Non-advertising	Register Resource	Poll



Discovery Manager: API Code Path

service/resource-encapsulation/src/resourceClient/RCSDiscoveryManager.cpp

```
92      RCSDiscoveryManager::DiscoveryTask::Ptr RCSDiscoveryManager::discoverResourceByTypes(
93              const RCSAddress& address, const std::string& relativeUri,
94              const std::vector< std::string >& resourceTypes, ResourceDiscoveredCallback cb)
95      {
96          return RCSDiscoveryManagerImpl::getInstance()->startDiscovery(address,
97                  relativeUri.empty() ? OC_RSRVD_WELL_KNOWN_URI : relativeUri,
98                  resourceTypes.empty() ? std::vector< std::string >{
99                          RCSDiscoveryManagerImpl::ALL_RESOURCE_TYPE } : resourceTypes,
100                 std::move(cb));
101     }
```

```
196     void DiscoveryRequestInfo::discover() const
197     {
198         for (const auto& it : m_resourceTypes)
199         {
200             discoverResource(m_address, m_relativeUri + "?rt=" + it, m_discoverCb);
201         }
202     }
```

Make URI for the discovery on specific resource type

service/resource-encapsulation/src/common/primitiveResource/src/PrimitiveResource.cpp

```
41      void discoverResource(const std::string& host, const std::string& resourceURI,
42              OCConnectivityType connectivityType, DiscoverCallback callback)
43      {
47          invokeOCFunc(static_cast< FindResource >(OC::OCPlatform::findResource),
48                  host, resourceURI, connectivityType, static_cast < OC::FindCallback >(
49                      std::bind(std::move(callback),
50                              std::bind(&PrimitiveResource::create, std::placeholders::_1))));
51      }
```

resource/src/OCPlatform_impl.cpp

```
172     OCStackResult OCPlatform_impl::findResource(const std::string& host,
173                                                 const std::string& resourceName,
174                                                 OCConnectivityType connectivityType,
175                                                 FindCallback resourceHandler)
176     {
177         return findResource(host, resourceName, connectivityType, resourceHandler, m_cfg.QoS);
178     }
```

service/resource-encapsulation/src/resourceClient/RCSDiscoveryManagerImpl.cpp

```
89      RCSDiscoveryManager::DiscoveryTask::Ptr RCSDiscoveryManagerImpl::startDiscovery(
90              const RCSAddress& address, const std::string& relativeUri,
91              const std::vector< std::string >& resourceTypes,
92              RCSDiscoveryManager::ResourceDiscoveredCallback cb)
93      {
94          if (!cb)
95          {
96              throw RCSInvalidParameterException{ "Callback is empty" };
97          }
98
99          validateTypes(resourceTypes);
100
101         const ID discoveryId = createId();
102
103         DiscoveryRequestInfo discoveryInfo(address, relativeUri, resourceTypes,
104                 std::bind(&RCSDiscoveryManagerImpl::onResourceFound, this,
105                         std::placeholders::_1, discoveryId, std::move(cb)));
106
107         discoveryInfo.discover();
108
109         {
110             std::lock_guard < std::mutex > lock(m_mutex);
111             m_discoveryMap.insert(std::make_pair(discoveryId, std::move(discoveryInfo)));
112         }
113
114         return std::unique_ptr< RCSDiscoveryManager::DiscoveryTask >(
115                 new RCSDiscoveryManager::DiscoveryTask(discoveryId));
116     }
```

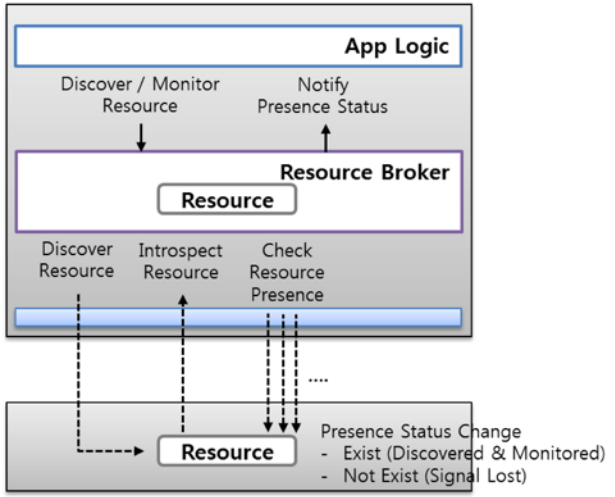
Check if the type is empty string

Assign discovery ID
(Used for handling ACK)

discoverResourceByTypes() in RE Layer API is forwarded to **findResource()** in Base API.

Resource Broker

- **Module to monitor the presence status of the specific resource**
  - Run on client-side
  - Used for managing the resources on remote devices
- **APIs**
  - `startMonitoring()`: start monitoring resource's presence status
  - `stopMonitoring()`: stop monitoring resource's presence status



Resource State (BROKER_STATE)	Descriptions
REQUESTED	Presence monitoring is started by startMonitoring(), but it is waiting for the response still.
ALIVE	Resource's presence is perceived **in 15 seconds**.
LOST_SIGNAL	Resource's presence is **NOT** perceived **in 15 seconds**. (Regarded as disconnection)
DESTROYED	State from calling stopMonitoring() to the object's destruction.

Resource Broker: API Code Path

service/resource-encapsulation/src/resourceClient/RCSRemoteResourceObject.cpp

```
237     void RCSRemoteResourceObject::startMonitoring(StateChangedCallback cb)
238     {
239         SCOPE_LOG_F(DEBUG, TAG);
240
241         if (!cb)
242         {
243             throw RCSInvalidParameterException{ "startMonitoring : Callback is NULL" };
244         }
245
246         if (isMonitoring())
247         {
248             OIC_LOG(DEBUG, TAG, "startMonitoring : already started");
249             throw RCSBadRequestException{ "Monitoring already started." };
250         }
251
252         m_brokerId = ResourceBroker::getInstance()->hostResource(m_primitiveResource,
253                 std::bind(hostingCallback, std::placeholders::_1, std::move(cb)));
254     }
```

service/resource-encapsulation/src/resourceBroker/src/ResourceBroker.cpp

```
64      BrokerID ResourceBroker::hostResource(PrimitiveResourcePtr pResource, BrokerCB cb)
65      {
66          OIC_LOG_V(DEBUG, BROKER_TAG, "hostResource().");
67          if(pResource == nullptr || cb == nullptr || cb == NULL)
68          {
69              throw InvalidParameterException("[hostResource] input parameter(PrimitiveResource or BrokerCB)
    is Invalid");
70          }
71
72          BrokerID retID = generateBrokerID();
80              try
81              {
82                  OIC_LOG_V(DEBUG, BROKER_TAG, "create the ResourcePresence.");
83                  presenceItem.reset(new ResourcePresence());
84                  presenceItem->initializeResourcePresence(pResource);
85              }catch(RCSPlatformException &e)
86              {
87                  throw FailedSubscribePresenceException(e.getReasonCode());
88              }
89              if(s_presenceList != nullptr)
90              {
91                  OIC_LOG_V(DEBUG, BROKER_TAG, "push the ResourcePresence in presenceList.");
92                  s_presenceList->push_back(presenceItem);
93              }
94          }
95          OIC_LOG_V(DEBUG, BROKER_TAG, "add the BrokerRequester in ResourcePresence.");
96          presenceItem->addBrokerRequester(retID, cb);
97
98          BrokerCBResourcePair pair(presenceItem, cb);
99          s_brokerIDMap->insert(std::pair<BrokerID, BrokerCBResourcePair>
100             (retID, BrokerCBResourcePair(presenceItem, cb)));
101
102         return retID;
```

Make Presence Resource object and add it to Presence List

service/resource-encapsulation/src/resourceBroker/src/ResourcePresence.cpp

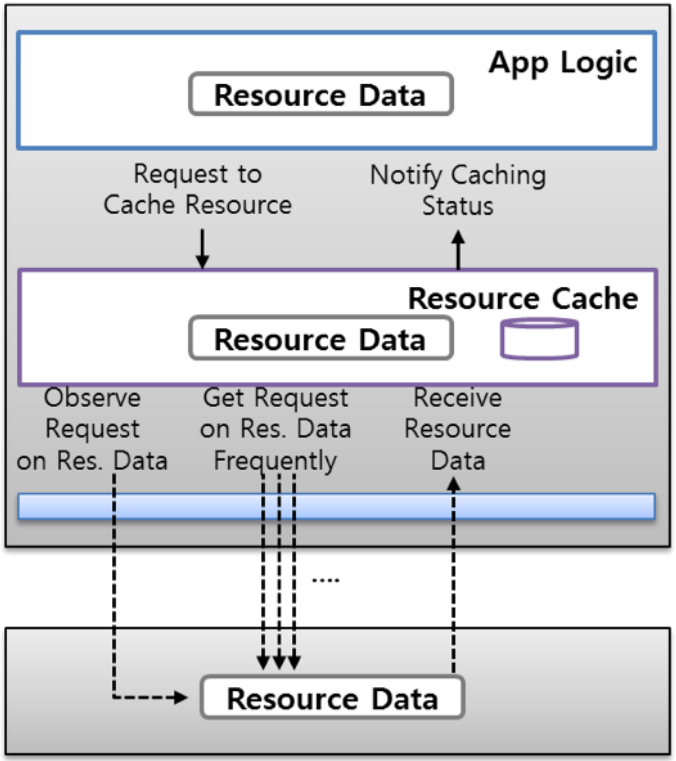
```
72      void ResourcePresence::initializeResourcePresence(PrimitiveResourcePtr pResource)
73      {
74          OIC_LOG_V(DEBUG,BROKER_TAG,"initializeResourcePresence().\n");
75          pGetCB = std::bind(getCallback, std::placeholders::_1, std::placeholders::_2,
76                  std::placeholders::_3, std::weak_ptr<ResourcePresence>(shared_from_this()));
77          pTimeoutCB = std::bind(timeOutCallback, std::placeholders::_1,
78                  std::weak_ptr<ResourcePresence>(shared_from_this()));
79          pPollingCB = std::bind(&ResourcePresence::pollingCB, this, std::placeholders::_1);
80
81          primitiveResource = pResource;
82          requesterList
83          = std::unique_ptr<std::list<BrokerRequesterInfoPtr>>
84          (new std::list<BrokerRequesterInfoPtr>);
85
86          timeoutHandle = expiryTimer.post(BROKER_STATE_TIMEOUT, pTimeoutCB);
87          OIC_LOG_V(DEBUG,BROKER_TAG,"initializeResourcePresence() timeoutHandle : %d",timeoutHandle);
88          primitiveResource->requestGet(pGetCB);
89
90          registerDevicePresence();
91      }
```

On making presence resource object, it binds presence event-related callback

- **pPollingCB**: Called in the period of 5 seconds → Send GET request to each resource
- **pGetCB**: Called when the resource is perceived in 15 seconds
- **pTimeoutCB**: Called when the resource is NOT perceived in 15 seconds

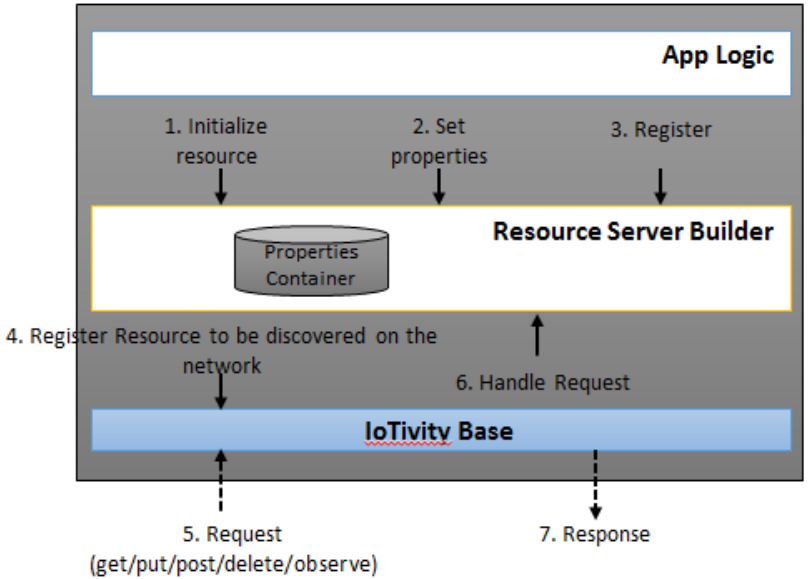
Resource Cache

- **Module caching the data of remote resource**
  - Intermittent connection to remote resource
- **APIs**
  - `startCaching()`: start caching resource
  - `stopCaching()`: stop caching resource
  - `getCachedAttribute()`: read the data of cached attribute
- **Two Cache Modes Configurable**
  - FREQUENCY: polling (default)
  - OBSERVE : push



Server Builder

- **Easier API that provides resource initialization, resource registration and property setting**
- **API: RCSResourceObject**
  - Server-side Resource object
  - Getter/setter functions
    - Easier handling of request and response than CRUDN
    - `getAttributeValue()`
    - `setAttribute()`



RE Layer API

Components	Functions	Description
Resource Broker	`startMonitoring()`	Start monitoring resource's presence status
	`stopMonitoring()`	Stop monitoring resource's presence status
Resource Cache	`startCaching()`	Start caching of requested resource
	`stopCaching()`	Stop caching of requested resource
	`getCachedAttribute()`	Read the data of cached attribute
Discovery Manager	`discoverResource()`	Discover any resources or resources with specific URI
	`discoverResourceByType()`	Discover resources with specific type
Server Builder	`getAttributeValue()`	Getter of resource's attribute
	`setAttribute()`	Setter of resource's attribute

Resource Encapsulation: Source Code

- **`android`**
  - RE layer API for Android
- **`examples`**
  - RE layer examples for Android, Linux, and Tizen
- **`include`**
- **`src`**
  - `common`: PrimitiveResource
  - `resourceBroker`: Resource Broker
  - `resourceCache`: Resource Cache
  - `resourceClient`: Discovery Manager
  - `serverBuilder`: RCSResourceObject
- **`unittests`**

Sample: Resource Client & Server

- **Run on two shells**
  - Directory: `out/linux/x86_64/release/service/resource-encapsulation/examples/linux`

```
$ ./sampleResourceServer
============================
1. Presence On
2. Presence Off
3. Quit
============================
1
============================
1. Creation of Simple Resource Without Handlers
2. Creation of Resource With Set and Get Handlers
3. Quit
============================
2
============================
Select Resource Type
1. Temperature
2. Light
3. Quit
1
```

```
$ ./sampleResourceClient
============================
1. discoverResource
2. quit
1
discoverResource start..
============================
1. Temperature Resource Discovery
2. Light Resource Discovery
============================
1
============================
Please input address (empty for multicast)
============================

Discovery in progress, press '1' to stop.
onResourceDiscovered callback ::
uri : /a/TempSensor
host address: coap://[fe80::a62:66ff:fe7f:9282%em1]:49836
```

Sample: Resource Client

1. Select resource type to discover

– runDiscovery() → discoverResource()
→ RCSDiscoveryManager::discoverResourceByType

service/resource-encapsulation/examples/linux/SampleResourceClient.cpp

```
void discoverResource()
{
    auto onResourceDiscovered = [](
            const RCSRemoteResourceObject::Ptr& discoveredResource)
    {
        std::cout << "onResourceDiscovered callback :: " << std::endl;

        std::cout << "uri : " << discoveredResource->getUri() << std::endl;
        std::cout << "host address : " << discoveredResource->getAddress() << std::endl;

        g_discoveredResources.push_back(discoveredResource);

        printDiscoveryInProgress();
    };

    auto resourceType = selectResourceType();
    auto address = inputAddress();

    printDiscoveryInProgress();

    auto discoveryTask = RCSDiscoveryManager::getInstance()->discoverResourceByType(address,
            resourceType, onResourceDiscovered);

    while(processUserInput() != 1);

    discoveryTask->cancel();
}
```

- **onResourceDiscovered():** Discover Event Callback (C++11 rambda function)

- **RCSDiscoveryManager::discoverResourceByType()**
: Discover resources with specific type

Sample: Resource Client

2. After discovery, menu will be printed

– discoverResource() → runResourceSeletion()

service/resource-encapsulation/examples/linux/SampleResourceClient.cpp

```
504 void runDiscovery()
505 {
506     static std::vector<MenuItem> discoveryMenuItems {
507         DECLARE_MENU(discoverResource),
508     };
509
510     handleItems(discoveryMenuItems);
511
512     if (g_discoveredResources.empty()) throw std::runtime_error("No resource found!");
513
514     g_currentRun = runResourceSelection;
515 }
```

```
497 void runResourceSelection()
498 {
499     handleItems(g_discoveredResources);
500
501     g_currentRun = runResourceControl;
502 }
```

```
449 void discoverResource()
450 {
451     auto onResourceDiscovered = [](
452             const RCSRemoteResourceObject::Ptr& discoveredResource)
453     {
454         std::cout << "onResourceDiscovered callback :: " << std::endl;
455
456         std::cout << "uri : " << discoveredResource->getUri() << std::endl;
457         std::cout << "host address : " << discoveredResource->getAddress() << std::endl;
458
459         g_discoveredResources.push_back(discoveredResource);
460
461         printDiscoveryInProgress();
462     };
463
464     auto resourceType = selectResourceType();
465     auto address = inputAddress();
466
467     printDiscoveryInProgress();
468
469     auto discoveryTask = RCSDiscoveryManager::getInstance()->discoverResourceByType(address,
470             resourceType, onResourceDiscovered);
471
472     while(processUserInput() != 1);
473
474     discoveryTask->cancel();
475 }
```

Sample: Resource Client

3. Menu to use resource control functions

- ex. runResourceControl() → startMonitoring()
  → RCSRemoteResourceObject::startMonitorg()

service/resource-encapsulation/examples/linux/SampleResourceClient.cpp

```
477 void runResourceControl()
478 {
479     static std::vector<MenuItem> resourceMenuItems {
480         DECLARE_MENU(startMonitoring),
481         DECLARE_MENU(stopMonitoring),
482         DECLARE_MENU(getRemoteAttributes),
483         DECLARE_MENU(setRemoteAttributes),
484         DECLARE_MENU(getWithInterface),
485         DECLARE_MENU(setWithInterface),
486         DECLARE_MENU(startCachingWithoutCallback),
487         DECLARE_MENU(startCachingWithCallback),
488         DECLARE_MENU(getResourceCacheState),
489         DECLARE_MENU(getCachedAttributes),
490         DECLARE_MENU(getCachedAttribute),
491         DECLARE_MENU(stopCaching),
492     };
493
494     handleItems(resourceMenuItems);
495 }
```

```
295 void startMonitoring()
296 {
297     if (g_selectedResource->isMonitoring())
298     {
299         std::cout << "\tAlready Started..." << std::endl;
300         return;
301     }
302
303     g_selectedResource->startMonitoring(&onResourceStateChanged);
304     std::cout << "\tMonitoring Started..." << std::endl;
305 }
```

```
239 void onResourceStateChanged(ResourceState resourceState)
240 {
241     std::cout << "onResourceStateChanged callback" << std::endl;
242
243     switch(resourceState)
244     {
245         case ResourceState::NONE:
246             std::cout << "\tState changed to : NOT_MONITORING" << std::endl;
247             break;
248
249         case ResourceState::ALIVE:
250             std::cout << "\tState changed to : ALIVE" << std::endl;
251             break;
252
253         case ResourceState::REQUESTED:
254             std::cout << "\tState changed to : REQUESTED" << std::endl;
255             break;
256
257         case ResourceState::LOST_SIGNAL:
258             std::cout << "\tState changed to : LOST_SIGNAL" << std::endl;
259             break;
260
261         case ResourceState::DESTROYED:
262             std::cout << "\tState changed to : DESTROYED" << std::endl;
263             break;
264     }
265 }
```

Sample: Resource Server

1. Resource Presence ON/OFF

- Set Presence TTL → Create Advertising Resource
- runPresenceSelection() → OC::startPresence()

service/resource-encapsulation/examples/linux/SampleResourceServer.cpp

```
216 void runPresenceSelection()
217 {
218     constexpr int PRESENCE_ON = 1;
219     constexpr int PRESENCE_OFF = 2;
220
221     std::cout << "=============================================
222     std::cout << PRESENCE_ON << ". Presence On
223     std::cout << PRESENCE_OFF << ". Presence Off
224     std::cout << PRESENCE_OFF + 1 << ". Quit
225     std::cout << "=============================================
226
227     if (processUserInput(PRESENCE_ON, PRESENCE_OFF) == PRESENCE_ON)
228     {
229         g_isPresenceStarted = true;
230         startPresence(3);
231     }
```

resource/csdk/stack/src/ocstack.c

```
3083 #ifdef WITH_PRESENCE
3084 OCStackResult OCStartPresence(const uint32_t ttl)
3085 {
3086     uint8_t tokenLength = CA_MAX_TOKEN_LEN;
3087     OCChangeResourceProperty(
3088             &(((OCResource *)presenceResource.handle)->resourceProperties),
3089             OC_ACTIVE, 1);
3090
3091     if (OC_MAX_PRESENCE_TTL_SECONDS < ttl)
3092     {
3093         presenceResource.presenceTTL = OC_MAX_PRESENCE_TTL_SECONDS;
3094         OIC_LOG(INFO, TAG, "Setting Presence TTL to max value");
3095     }
3096     else if (0 == ttl)
3097     {
3098         presenceResource.presenceTTL = OC_DEFAULT_PRESENCE_TTL_SECONDS;
3099         OIC_LOG(INFO, TAG, "Setting Presence TTL to default value");
3100     }
3101     else
3102     {
3103         presenceResource.presenceTTL = ttl;
```

resource/src/OCPlatform_impl.cpp

```
348     OCStackResult OCPlatform_impl::startPresence(const unsigned int announceDurationSeconds)
500     OCStackResult InProcServerWrapper::startPresence(const unsigned int seconds)
```

resource/src/InProcServerWrapper.cpp

Sample: Resource Server

2. Create Resource

- Create two resources (temperature, brightness) through RCSResourceObject(Server Builder API)

service/resource-encapsulation/examples/linux/SampleResourceServer.cpp

```
165 void runResourceTypeSelection(int resourceMode)
178     switch (resourceType)
179     {
180         case RESOURCE_TEMP:
181             attrKey = "Temperature";
182             initServer("/a/TempSensor", "oic.r.temperaturesensor", attrKey);
183
184             displayMenuFunc = displayControlTemperatureMenu;
185             break;
186
187         case RESOURCE_LIGHT:
188             attrKey = "Brightness";
189             initServer("/a/light", "oic.r.light", attrKey)
190
191             displayMenuFunc = displayControlLightMenu;
192             break;
193     }
194
195     if (resourceMode == CUSTOM_SERVER)
196     {
197         g_resource->setGetRequestHandler(requestHandlerForGet);
198         g_resource->setSetRequestHandler(requestHandlerForSet);
199     }
200
201     g_currentRun = std::bind(runResourceControl, displayMenuFunc, std::move(attrKey));
202 }
```

```
121 void initServer(const std::string& resourceUri, const std::string& resourceType,
122         const std::string& attrKey)
123 {
124     g_resource = RCSResourceObject::Builder(resourceUri, resourceType, ACTUATOR_INTERFACE)
125             .addInterface(CUSTOM_INTERFACE)
126             .addInterface(SENSOR_INTERFACE)
127             .setDefaultInterface(BASELINE_INTERFACE)
128             .setDiscoverable(true)
129             .setObservable(true)
130             .build();
131
132     g_resource->setAutoNotifyPolicy(RCSResourceObject::AutoNotifyPolicy::UPDATED);
133     g_resource->setSetRequestHandlerPolicy(RCSResourceObject::SetRequestHandlerPolicy::NEVER);
134     g_resource->setAttribute(attrKey, 0);
135 }
```

Sample: Resource Server

3. Resource Getter & Setter ON/OFF

- Getter & Setter function for handling GET/PUT requests

service/resource-encapsulation/examples/linux/SampleResourceServer.cpp

```
165 void runResourceTypeSelection(int resourceMode)
178     switch (resourceType)
179     {
180         case RESOURCE_TEMP:
181             attrKey = "Temperature";
182             initServer("/a/TempSensor", "oic.r.temperaturesensor", attrKey);
183
184             displayMenuFunc = displayControlTemperatureMenu;
185             break;
186
187         case RESOURCE_LIGHT:
188             attrKey = "Brightness";
189             initServer("/a/light", "oic.r.light", attrKey);
190
191             displayMenuFunc = displayControlLightMenu;
192             break;
193     }
194
195     if (resourceMode == CUSTOM_SERVER)
196     {
197         g_resource->setGetRequestHandler(requestHandlerForGet);
198         g_resource->setSetRequestHandler(requestHandlerForSet);
199     }
200
201     g_currentRun = std::bind(runResourceControl, displayMenuFunc, std::move(attrKey));
202 }
```

```
 99 RCSGetResponse requestHandlerForGet(const RCSRequest&, RCSResourceAttributes& attrs)
100 {
101     std::cout << "Received a Get request from Client" << std::endl;
102     printAttributes(attrs);
103
104     {
105         RCSResourceObject::LockGuard lock(g_resource);
106         std::cout << "\nSending response to Client : " << std::endl;
107         printAttributes(g_resource->getAttributes());
108     }
109
110     return RCSGetResponse::defaultAction();
111 }
```

```
113 RCSSetResponse requestHandlerForSet(const RCSRequest&, RCSResourceAttributes& attrs)
114 {
115     std::cout << "Received a Set request from Client" << std::endl;
116     printAttributes(attrs);
117
118     return RCSSetResponse::defaultAction();
119 }
```

Sample: Resource Server

4. Handling Resource Attribute Value

– runResourceControl() → updateAttribute()
→ RCSResourceObject::getAttributes() / getAttributeValue()

service/resource-encapsulation/examples/linux/SampleResourceServer.cpp

```
159 void runResourceControl(DisplayControlMenuFunc displayMenuFunc, const std::string& attrKey)
160 {
161     displayMenuFunc();
162     updateAttribute(attrKey, processUserInput(INCREASE, DECREASE));
163 }
```

```
137 void updateAttribute(const std::string& attrKey, int control)
138 {
139     const int diff = control == INCREASE ? 1 : - 1;
140
141     {
142         RCSResourceObject::LockGuard lock(g_resource);
143         auto& attrs = g_resource->getAttributes();
144         attrs[attrKey] = attrs[attrKey].get<int>() + diff;
145     }
146
147     if(control == INCREASE)
148     {
149         std::cout << attrKey << " increased." << std::endl;
150     }
151     else
152     {
153         std::cout << attrKey << " decreased." << std::endl;
154     }
155     std::cout << "\nCurrent " << attrKey << ": "
156             << g_resource->getAttributeValue(attrKey).get<int>() << std::endl;
157 }
```